European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Production, Manufacturing, Transportation and Logistics

# The maintenance scheduling and location choice problem for railway rolling stock

Jordi Zomer [a,1], Nikola Bešinović [a,2], Mathijs M. de Weerdt [b], Rob M.P. Goverde [a,*]

[a] Department of Transport and Planning, Delft University of Technology, Delft, The Netherlands
[b] Department of Software and Computer Technology, Delft University of Technology, Delft, The Netherlands

ARTICLE INFO

*Keywords:*
Transportation
Maintenance scheduling
Railway rolling stock
Location choice
Optimization

ABSTRACT

The increasing train traffic over railway networks stretches the demand for capacity of railway yards and rolling stock maintenance locations, which increasingly limits performance and further growth. Therefore, the scheduling of rolling stock maintenance and the choice regarding optimal locations to perform maintenance is increasingly complicated. This research introduces a Maintenance Scheduling and Location Choice Problem (MSLCP). It simultaneously determines maintenance locations and maintenance schedules of rolling stock, while considering the available capacity of maintenance locations. Solving the MSLCP using one large Mixed Integer Programming appears not to perform well enough. Therefore, to solve the MSLCP, an optimization framework based on Logic-Based Benders' Decomposition (LBBD) is proposed by combining two models, the Maintenance Location Choice Problem (MLCP) and the Activity Planning Problem (APP), to assess the capacity of an MLCP solution. Within the LBBD, four variants of cut generation procedures are introduced to improve the computational performance: a naive procedure, two heuristic procedures and the so-called min-cut procedure that aims to exploit the specific characteristics of the problem at hand. The framework is demonstrated on realistic scenarios from the Dutch railways. It is shown that the best choice for the cut generation procedure depends on the objective: when aiming to find a good but not necessarily optimal solution, the min-cut procedure performs best, whereas when aiming for the optimal solution, one of the heuristic procedures is the preferred option. The techniques used in the current research are new to the current field and offer interesting next research opportunities.

## 1. Introduction

In many countries, rail transport is increasingly important. In order for a railway network to function properly, the rolling stock (RS) that operates on the railway network (i.e. locomotives, passenger wagons and freight wagons, multiple units) needs to receive maintenance on a regular basis. The aim of maintenance is to ensure that the rolling stock that operates on the network remains available to ensure a reliable train service, safe and comfortable for passengers (Dinmohammadi et al., 2016). To this end, maintenance activities can be divided into two categories: regular maintenance, corresponding to the maintenance activities with higher frequencies (every 1 to 14 days) and shorter duration (1–3 h) which can be performed whenever the unit has a planned standstill, and heavy maintenance, corresponding to maintenance types with lower frequencies (every several months or less) and longer duration (up to several days) during which the unit is taken out of service (see e.g. Andrés et al., 2015). The current work focuses on regular maintenance in particular. This type of maintenance is similar for all train types so teams and locations are interchangeable.

For regular maintenance, the maximum interval between consecutive maintenance activities is governed by strict rules that are imposed by railway authorities. These maintenance activities are carried out at so-called maintenance locations, which are railway yards with maintenance facilities, spread over the network. The number of maintenance teams stationed at a location is the operator's decision and an important determinant of the capacity of a maintenance location. In particular, a distinction can be made between daytime operations (i.e. a location is opened during daytime) and nighttime operations (i.e. a location is opened during nighttime). For example, in The Netherlands, maintenance is usually carried out during nighttime.

* Corresponding author.
*E-mail addresses:* jordizomer@gmail.com (J. Zomer), nikola.besinovic@tu-dresden.de (N. Bešinović), M.M.deWeerdt@tudelft.nl (M.M. de Weerdt), r.m.p.goverde@tudelft.nl (R.M.P. Goverde).
[1] Present address: Centre for Quantitative Methods (CQM), Eindhoven, The Netherlands.
[2] Present address: Chair of Railway Operations, "Friedrich List" Faculty of Transport and Traffic Sciences, Technical University of Dresden, Dresden, Germany.

https://doi.org/10.1016/j.ejor.2025.12.005
Received 18 November 2024; Accepted 1 December 2025
Available online 6 December 2025

The increasing use of the capacity of the railway network leads to two issues for rolling stock maintenance. First, this process is currently performed manually and on an ad-hoc basis during operations (for a large part during nighttime). The planners attempt to keep an overview in a spreadsheet with the rolling stock units that need to be planned. They manually change the order of rolling stock units to make them fit and assign the maintenance teams to rolling stock units, often in an unsystematic way. This planning process makes it hard to make a good planning, and in addition, as a result of the increasing number trains, the complexity of this scheduling process is increasing. This raises the need for tools that automate the maintenance scheduling process. Second, the use of the capacity of maintenance locations during nighttime is under pressure and reaching its capacity. As a result, a railway operator can consider to perform more maintenance activities during daytime, as is the case in The Netherlands for example. These issues were originally addressed by Zomer et al. (2021), providing a model for the *Maintenance Location Choice Problem* (MLCP). However, they assumed an unlimited capacity of maintenance locations and do not provide a maintenance schedule. The capacity of maintenance locations is a challenging factor to incorporate, as it typically depends on the optimal planning of all maintenance activities (which is not readily available).

This research introduces a new mathematical problem formulation, the Maintenance Scheduling and Location Choice Problem (MSLCP), which extends the MLCP. For a given rolling stock circulation, it determines an optimal maintenance schedule and an optimal maintenance location choice while including capacity constraints of maintenance locations. In this schedule, as much activities as possible are moved from nighttime to daytime, which is in line with the goal of the Dutch railways, for whom as much daytime maintenance as possible is desirable as it reduces severe capacity issues, while providing financial benefits due to lower daytime labour costs. First exploratory experimental results demonstrate that the time necessary to solve the problem using a straightforward MILP formulation is too long for practical applications where a solution is required within minutes. In order to solve this problem more quickly, the current paper introduces a subproblem to assess the capacity of a MLCP solution, called the *Activity Planning Problem* (APP). To solve the MSLCP, an optimization framework based on Logic-Based Benders' Decomposition (LBBD) is proposed by combining two models, i.e., the MLCP and APP. Within the LBBD, three variants of cut generation procedures are introduced: a naive procedure, a heuristic procedure, and the so-called min-cut procedure which uses the specific structure of the problem at hand. The performance of MSLCP is demonstrated on a realistic case from the Dutch railways.

The contribution of this research is fourfold. First, it extends the model initially proposed by Zomer (2020) and further established by Zomer et al. (2021), by introducing maintenance location capacity constraints and thereby making the model capable of delivering a complete maintenance schedule taking into account available maintenance teams. Second, it provides an efficient method to assess the required capacity of a maintenance schedule which has as an additional benefit that it can be used to quickly provide rolling stock dispatchers with a maintenance activity planning during operations. Third, it proposes an advanced solution strategy for the inclusion of the capacity constraints based on Logic-Based Benders' Decomposition. Fourth, it demonstrates the applicability of the proposed solution on real-life instances at the Dutch railways.

The remainder of this paper is structured as follows. Section 2 summarizes the most important existing literature. Section 3 restates the MLCP based on Zomer et al. (2021). Section 4 formulates the MSLCP, extending the MLCP to include capacity. Section 5 provides a solution methodology to solve the MSLCP, in which the MLCP and APP are the main building blocks. Section 6 reports computational results for the MSLCP and Section 7 gives the main conclusions.

## 2. Literature review

The current work considers rolling stock maintenance scheduling as well as rolling stock maintenance location choice. This section aims to identify the contributions of the current work to the literature and to obtain insights in the methodologies and techniques used in related research.

### 2.1. Maintenance scheduling

Maróti and Kroon (2007) considered a problem regarding assigning rolling stock to heavy maintenance. They proposed a model to make modifications to the regular rolling stock circulation to route rolling stock units to maintenance locations and formulated it as an integer programming problem. In situations where only one rolling stock unit needs to be rerouted, this formulation provides the optimal solution; in situations where multiple rolling stock units need to be rerouted the formulation is used within a heuristic framework. Wagenaar et al. (2017) proposed a model that reschedules the rolling stock circulation after disruptions taking into account the current maintenance planning. They based their models on the composition model, which assigns rolling stock units to train trips. They came up with three models that have comparable performance, dependent on the problem size.

Herr et al. (2017) considered a problem in which rolling stock units need to be assigned to train trips such that maintenance constraints are satisfied. They proposed a MIP model and the objective that they used is to schedule maintenance as late as possible, thereby making optimal use of the total allowable interval between maintenance activities. Andrés et al. (2015), similarly to Herr et al. (2017), considered the problem of assigning rolling stock units to train trips. They used an aggregated space-time network in which the nodes are trip arrival times or trip departure times with the corresponding location. A MIP model that minimizes total operating costs was designed and a column generation approach was used to solve the problem in reasonable time.

Van Hövell et al. (2022) proposed a new rolling stock exchange concept for servicing, which provides possibilities of exchanging rolling stock during daytime to increase the effectiveness of the capacity usage at a single service location. A new MILP model is introduced to model Rolling Stock Servicing Scheduling Problem (RS-SSP). Xu and Dessouky (2022) addressed train shunting with service scheduling in a single depot where daily maintenance, cleaning operation, and safety operational requirements are considered. They constructed a two-layer time-space network in which each layer is used by trains traveling in the same direction. They formulated the considered problem as a minimum-cost multi-commodity network flow model with incompatible arc sets and operational constraints. To solve the network flow problem, they presented a Lagrangian relaxation heuristic.

Zhong et al. (2019) developed a two-stage heuristic approach to rolling stock scheduling that incorporates maintenance requirements. In the first stage, maintenance constraints were temporarily ignored to generate candidate schedules using a conventional Mixed Integer Programming model; in the second stage, these schedules were evaluated for feasibility with respect to maintenance needs through an assignment problem. A computational study using real-world data from the Chinese High-Speed Railway demonstrated the effectiveness of the method. Amorosi et al. (2024) addressed the interdependence between high-speed train maintenance and rolling stock planning, which is often overlooked in existing models. The authors proposed an integrated mixed-integer programming approach that jointly optimizes both aspects and tested it on instances derived from a real-world case study. Compared to the commonly used two-stage method, the integrated model produced more feasible solutions and significantly reduced the optimality gap, highlighting the inefficiencies of separating planning and scheduling. Lin et al. (2023) proposed a 0–1 program-

**Table 1**
Overview of the literature discussed in Section 2.

| | RS unit allocation | Maintenance scheduling | Location choice |
|---|---|---|---|
| Maróti and Kroon (2007) | x | x | |
| Andrés et al. (2015) | x | x | |
| Wagenaar et al. (2017) | x | x | |
| Herr et al. (2017) | x | x | |
| Tönissen and Arts (2018) | | | x |
| Canca and Barrena (2018) | x | | x |
| Tönissen et al. (2019) | | | x |
| Zhong et al. (2019) | x | x | |
| Zomer et al. (2021) | | | x |
| Van Hövell et al. (2022) | | x | |
| Xu and Dessouky (2022) | | x | |
| Lin et al. (2023) | | x | |
| Amorosi et al. (2024) | x | x | |
| Folco et al. (2024) | | x | |
| *Current paper* | | x | x |

ming model that was formulated to address high-level maintenance scheduling within these variable windows. The authors proposed an algorithm to adjust both maintenance dates and planned daily mileages dynamically. A real-world case study and sensitivity analysis were conducted to validate the model and assess its robustness under varying conditions. Folco et al. (2024) introduced the Rolling Stock Maintenance Scheduling Problem (RSMSP), incorporating cyclical maintenance activities into the planning framework. To manage complexity, the authors applied a rolling horizon approach that decomposed the problem into sequential subproblems focused on upcoming maintenance cycles. These subproblems were solved using integer linear programming to ensure efficient and feasible scheduling. Related problems were addressed in the area of aviation, for example by Clarke et al. (1997) and Gopalan and Talluri (1998), who aimed to assign specific aircraft to each flight from a given set of flights, and Sarac et al. (2006), who developed a model that solves the aircraft maintenance scheduling problem including maintenance constraints in an operational context.

### 2.2. Maintenance location choice

Tönissen et al. (2019) aimed at locating the maintenance facilities in the railway network. They came up with models that determine optimal maintenance locations under line and fleet planning that are subject to uncertainty or change. They proposed two-stage stochastic mixed integer programming models, in which the first stage is to open a facility, and in the second stage to minimize the routing cost for the first-stage location decision for each line plan scenario. Tönissen and Arts (2018) built on Tönissen et al. (2019) by including recovery costs of maintenance location decisions, unplanned maintenance, multiple facility sizes and economies of scale (providing that a location twice as big is not twice as expensive). Since, as a result, the second-stage problem becomes NP-hard, an algorithm was provided with the aim to avoid having to solve the second stage for every scenario.

Canca and Barrena (2018) considered the simultaneous rolling stock allocation to lines and choice for depot locations in a rail-rapid transit context. They proposed a MILP formulation which appeared hard to solve. Therefore they proposed a three-step heuristic approach determining first the minimum number of vehicles needed for each line, subsequently the routes of rolling stock on each line, and lastly the circulation of rolling stock on lines over multiple days together with the depot choice.

Zomer et al. (2021) introduced the Maintenance Location Choice Problem (MLCP). To solve it, the authors developed a MILP model taking a rolling stock circulation as input, and provided for this rolling stock circulation an optimal maintenance location choice that minimize the total number of maintenance activities during nighttime, thereby reducing the pressure on maintenance locations during nighttime. However, they did not include the capacity of maintenance locations, nor determine exact maintenance schedules (i.e. maintenance activities are assigned to maintenance opportunities, which are longer time windows in which maintenance has to take place at some moment), and do not consider actual moments when maintenance has to be performed.

Some related research can be found in the area of aviation. Examples are the works by Feo and Bard (1989) and Gopalan (2014), who consider the problem of assigning aircraft to flights and simultaneously determining maintenance locations and introduce various heuristics to solve the problem.

### 2.3. Current work

In Table 1, the discussed literature is classified in several categories. It shows whether it considered the allocation of rolling stock (RS) units to trips, whether it considered maintenance constraints, whether it created an explicit maintenance schedule for every (relevant) RS unit and whether it considered maintenance location choice optimization.

Research on railway maintenance scheduling began with Maróti and Kroon (2007), who modeled rolling stock rerouting for maintenance, and later expanded to include disruption recovery (Wagenaar et al., 2017), and servicing during daytime optimization (Van Hövell et al., 2022). More recent studies have focused on integrating maintenance with rolling stock planning through mixed-integer programming models (Amorosi et al., 2024; Zhong et al., 2019) and advanced scheduling methods that address operational complexities (Folco et al., 2024; Lin et al., 2023), resulting in improved efficiency and cost-effectiveness in rolling stock operations. Meanwhile, early research on maintenance location choice, such as Canca and Barrena (2018), largely overlooked scheduling aspects. Subsequent work has incorporated additional considerations like recovery costs (Tönissen & Arts, 2018) and daytime maintenance planning (Zomer et al., 2021). Only recently, with growing capacity constraints at maintenance facilities due to expanding rolling stock fleets, has the need to jointly address maintenance location and scheduling become increasingly urgent.

This literature review indicates that several aspects have not been addressed in the currently existing literature. First, although variants of problems relating to rolling stock maintenance location and maintenance scheduling have been investigated independently, this joint problem has not been tackled. That is, no research has aimed to determine optimal opening of maintenance locations and simultaneously find an optimal maintenance schedule for a given rolling stock circulation. Second, although some papers do consider some type of a constraint for the available capacity at maintenance locations, such constraints are typically rather general and ignore many practical aspects. Third, the existing research does not consider available maintenance teams to perform maintenance tasks.

The current research attempts to fill the aforementioned gaps in the existing literature by finding the optimal location choice and optimal maintenance schedule simultaneously, while considering capacity restrictions measured as the number of available maintenance teams. In addition, it delivers a complete maintenance schedule, which provides operators at maintenance locations with exact moments when each rolling stock unit needs to be maintained and by which maintenance team.

## 3. Maintenance location choice problem (MLCP)

This section summarizes the mathematical model of the *Maintenance Location Choice Problem* (MLCP). For more detailed explanations of the model and the computational experiments, the reader may resort to Zomer et al. (2021).

The following notation is used for the parameters of the model. Let $I$ be the set of rolling stock units, $T \in \mathbb{R}$ the length of the planning horizon and $L$ the set of potential maintenance locations. The rolling stock circulation is assumed to be given. A *maintenance opportunity* (MO) occurs when a rolling stock unit is standing still at a potential maintenance location. Let $J_i \equiv \{1, \dots, \overline{J_i}\}$ denote the MOs for rolling stock unit $i \in I$. The location of a rolling stock unit $i$ at MO $j \in J_i$ is denoted by $l_{ij} \in L$. The start time of MO $j \in J_i$ for rolling stock unit $i \in I$ is denoted by $s_{ij} \in \mathbb{R}$ and the end time by $e_{ij} \in \mathbb{R}$. For any problem instance, these times are dictated by the rolling stock plan under consideration.

Let $d_{ij} = 1$ indicate the decision that an MO occurs during daytime. An MO is considered to be during daytime if and only if both the start time $s_{ij}$ and the end time $e_{ij}$ are during daytime of the same day, i.e.:

$$d_{ij} = \begin{cases} 1 & \text{if } (\delta^D \le s_{ij} \bmod 24 < \delta^N) \text{ and } (\delta^D \le e_{ij} \bmod 24 < \delta^N) \\ 0 & \text{else} \end{cases},$$

where $\delta^D$ is the time daytime maintenance starts and $\delta^N$ the time nighttime maintenance starts. Unless stated otherwise, $\delta^D = 7.00$ and $\delta^N = 19.00$. Let $K$ be the set of *maintenance types*, $K \equiv \{1, \dots, \overline{K}\}$. For each maintenance type $k \in K$, let $v_k \in \mathbb{R}^+$ be its duration and let $o_k \in \mathbb{R}^+$ be the maximum interval between two consecutive maintenance activities of maintenance type $k$.

The decision variable $y_l^D \in \{0, 1\}$ to *open a potential maintenance location during daytime* is 1 if location $l \in L$ is available for daytime maintenance. Similarly $y_l^N \in \{0, 1\}$ is equal to 1 if location $l \in L$ is available for nighttime maintenance. The number of potential maintenance locations that can be opened during daytime is restricted by $L^D_{\max}$:

$$\sum_{l \in L} y_l^D \le L^D_{max}. \tag{1}$$

This parameter reflects that opening an unlimited number of daytime maintenance locations is not possible in practice, as opening a new location for daytime maintenance comes with significant start-up costs and organizational challenges.

The *assignment decisions* of maintenance activities to maintenance opportunities are encoded by $x_{ijk} \in \{0, 1\}$, which is 1 if maintenance of type $k$ is performed to rolling stock unit $i \in I$ at MO $j \in J_i$, and 0 otherwise. It is required that the total time available at MO $j$ is not exceeded:

$$\sum_{k \in K} x_{ijk} v_k \le e_{ij} - s_{ij} \qquad \forall i \in I, j \in J_i. \tag{2}$$

Furthermore, an MO $j$ can only be used if the corresponding location is open at the moment of the MO. Therefore, if $d_{ij} = 0$ and $y^N_{l_{ij}} = 0$ then $x_{ijk} = 0 \; \forall k \in K$, and similarly if $d_{ij} = 1$ and $y^D_{l_{ij}} = 0$ then $x_{ijk} = 0 \; \forall k \in K$, which is encoded in a single linear constraint as follows:

$$x_{ijk} \le y^D_{l_{ij}} \cdot d_{ij} + y^N_{l_{ij}} \cdot (1 - d_{ij}) \qquad \forall i \in I, j \in J_i, k \in K. \tag{3}$$

Finally, the intervals between two successive maintenance activities $j$ and $j'$ of the same type $k$ should be at most $o_k$ apart. This is modeled as follows: if $x_{ijk} = 1$ (and $e_{ij} + o_k \le T$, meaning the next maintenance activity is within the current horizon) then $\exists j' \in V_{ijk} : x_{ij'k} = 1$, where $V_{ijk} = \{p \in J_i : e_{ij} < s_{ip} \le e_{ij} + o_k\}$. For a correct start, let $b_{ik}$ be the time since the last maintenance activity of type $k$ for rolling stock unit $i$ at the start of the planning horizon, let $V_{i0k} = \{p \in J_i : s_{ip} \le o_k + b_{ik}\}$ and

$$1 \le \sum_{p \in V_{i0k}} x_{ipk} \qquad \forall i \in I, k \in K \tag{4}$$

$$x_{ijk} \le \sum_{p \in V_{ijk}} x_{ipk} \qquad \forall i \in I, j \in J_i, k \in K : e_{ij} + o_k \le T. \tag{5}$$

The model aims to find $x_{ijk}$ and $y_l$ satisfying the constraints (1) to (5) that minimize number of maintenance activities during the night:

$$\min \sum_{i \in I} \sum_{j \in J_i} \sum_{k \in K} x_{ijk}(1 - d_{ij}) + \varepsilon \sum_{i \in I} \sum_{j \in J_i} \sum_{k \in K} x_{ijk}. \tag{6}$$

The second term penalizes every maintenance activity with an arbitrarily small penalty cost $\varepsilon$ in order to avoid unnecessary maintenance activities being performed.

An overview of all mathematical formulation used throughout this paper is provided in Table A.1 in A.

## 4. Maintenance scheduling and location choice problem (MSLCP)

The MLCP delivers an assignment of maintenance activities to maintenance opportunities. Maintenance activities are not explicitly scheduled accurate to the minute, but may be performed anytime within the MO. This alone does not allow to directly determine the required number of maintenance teams to effectuate the maintenance schedule. Therefore, the MLCP is extended in such a way that maintenance activities are scheduled at specific moments in time and assigned to a specific maintenance team. The resulting problem is called the *Maintenance Scheduling and Location Choice Problem* (MSLCP). This problem considers jobs. A job represents the activities that need to be performed on one rolling stock unit during a specified maintenance opportunity. A job may contain one maintenance activity of a specific maintenance type, but can also contain multiple maintenance activities of different maintenance types.

An important reason why different maintenance activities on the same rolling stock unit are grouped into jobs is that, in practice, maintenance activities of different types on the same rolling stock unit often cannot be performed simultaneously, e.g. external cleaning and wheels inspection. To ensure this, the slightly stricter assumption is made that maintenance activities on one rolling stock unit need to be performed subsequently and uninterruptedly. This assumption is deemed acceptable in practice. It also simplifies the model as it is not necessary to include separate, complicating constraints to prohibit that maintenance activities of different types on the same rolling stock units are performed simultaneously.

A key additional feature of the MSLCP is that it tracks the number of teams necessary to perform the given set of jobs, which is a measure of the required capacity of a MLCP solution, enabling also to limit this capacity to a predefined number of teams. Additionally, it gives the corresponding optimal activity planning, defining the start and end times of each job, which has useful practical applications as well.

The MSLCP determines the required number of maintenance teams for each maintenance shift. A maintenance shift is a period of time for a specific location for which a planning is made. The current research assumes each day contains two maintenance shifts: a daytime shift and a nighttime shift. Therefore, the MSLCP now not only determines the optimal maintenance location choice, but also a detailed shift plan and the required capacity. These additional model features show similarities with the class of Parallel Machine Scheduling Problems, as addressed by for example Kravchenko and Werner (2009).

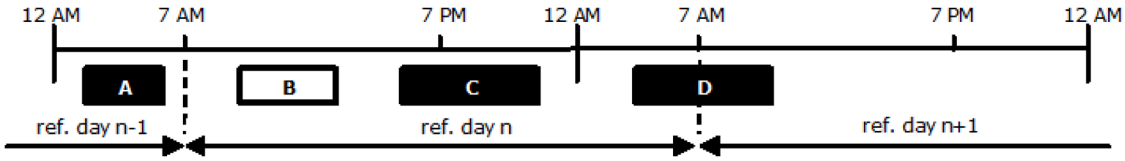



**Fig. 1.** Assignment of MOs to shifts. This figure presents four MOs (A, B, C and D), of which B is classified as daytime MO and A, C and D are classified as nighttime MOs.

### 4.1. Additional mathematical notation for MSLCP

Let $Q$ be a given set of *jobs* that need to be scheduled, and let $q_{ij} \in Q$ represent the job that consists of the maintenance activities assigned to MO $j$ for rolling stock unit $i$. Let $r_{ij} \in \mathbb{R}$ be the release time and $t_{ij} \in \mathbb{R}$ be the deadline time of job $q_{ij}$, for MO $J$ of unit $i$. The duration of the job is the sum of the duration of these main activities. The release time and deadline time are based on the start and end time of the respective maintenance opportunity, explained in more detail later in this section.

Maintenance is performed in a *maintenance shift*. Let $U$ be the set of maintenance shifts. The current research assumes maintenance shifts occur in two time windows: the daytime maintenance shift between 7.00 and 19.00 and the nighttime maintenance shift between 19.00 and 7.00. Unique maintenance shifts are characterised by a combination of maintenance location, time window (i.e. daytime or nighttime) and reference day (i.e. the day when the maintenance shift starts). An example of a unique maintenance shift would be *the night shift in Amsterdam on day 3*, meaning the shift that starts in Amsterdam at 19.00 on day 3 and ends in Amsterdam at 07.00 on day 4.

The following procedure is used to determine to what maintenance shift an MO belongs.

- Suppose an MO is classified as a daytime MO. Then, by the definition of daytime MOs, it is clear that the entire MO is contained within the daytime time window. The reference day is therefore equal to the end time of the MO and it belongs to the daytime maintenance shift of that particular day.
- Suppose an MO is classified as a nighttime MO. Note that this does not necessarily mean that the end time is during nighttime (for example, an MO starting during nighttime and ending during daytime is still classified as a nighttime MO). For nighttime MOs, we assign these MOs to the last nighttime maintenance shift that it was in. In other words, if the end time is between 0.00 and 19.00, it is classified as an MO during the nighttime shift with a reference day at the *previous day*; if, on the other hand, the end time is between 19.00 and 0.00, this last maintenance shift is the nighttime maintenance shift with reference day on the *current day*.

An example is found in Fig. 1. This figure presents four MOs with their start and end time. Based on the above described procedure, these MOs can be assigned to maintenance shifts: MO $A$ is assigned to the nighttime shift of reference day $n-1$, MO $B$ is assigned to the daytime shift of reference day $n$ and MOs $C$ and $D$ are assigned to the nighttime shift of reference day $n$.

Each job needs to be performed by one and only one maintenance *team*. The team works on this job uninterruptedly, i.e. the job cannot be split into multiple separate parts (meaning preemption is not allowed). Let $\overline{N}_u$ be the maximum number of available maintenance teams in shift $u$ and define $N_u = \{1, \dots, \overline{N}_u\}$ to be the set of maintenance teams in shift $u$. The maintenance jobs are assigned to maintenance teams, and the start time of each maintenance job is determined. The end time of the job is then automatically determined by adding the job duration to the start of the maintenance job. The start time should be such that it is after the release time of a job, and such that the end time is before the deadline time of a job.

The current formulation of the MSLCP uses so-called *moments*. A moment represents the opportunity of a maintenance team to start a job. This is a construct used to retain linearity. Each team has a set of moments available, corresponding to the maximum number of jobs that they can perform. A job can be assigned to any moment. If a job is assigned to a moment, the start time of this particular moment is associated to the start time of the corresponding maintenance job. The introduction of the concept of moments allows to model a sequential planning, by requiring that if a job is assigned to moment $m$, moment $m+1$ can start only after the job assigned to moment $m$ is finished.

Let $\overline{M}$ be the number of moments available per team and define $M = \{1, \dots, \overline{M}\}$ to be the set of moments. Note that the maximum number of moments used by a team occurs when a team is continually occupied with maintenance activities of the shortest duration for the entire length of the maintenance shift. A sufficiently large $\overline{M}$ can thus be obtained by dividing the total time available in a maintenance shift over the minimum time required for each maintenance job. Eq. (7) gives an appropriate value for $\overline{M}$ that is used in the current research.

$$\overline{M} = \left\lceil \frac{\delta^N - \delta^D}{\min_{k \in K} v_k} \right\rceil \tag{7}$$

For each of the maintenance jobs, a *release time* and a *deadline time* need to be specified. It must be noted that in most cases, MOs are contained in either the daytime shift or the nighttime shift. In these cases, the release time is equal to the start of the MO and the deadline time is equal to the end of the MO. However, there are also MOs that are not fully contained in the corresponding maintenance shift, such as MOs $C$ and $D$ in Fig. 1. Still, they are assigned to nighttime maintenance shift $n$ and therefore need to be performed in this shift.

In order to make sure that maintenance activities are performed as much as possible in the maintenance shift that they were assigned to, the following rules are used to determine the release times.

- If a maintenance activity takes place in a daytime MO, then its release time is equal to the start of the corresponding MO.
- If a maintenance activity takes place in a nighttime MO and the start of the MO is after the start of the maintenance shift, then the release time of the maintenance job is equal to the start of the corresponding MO.
- If a maintenance activity takes place in a nighttime MO and the start of the MO is before the start of the maintenance shift, then the release time of the maintenance job is set to the start of the maintenance shift (usually 19.00). There is one exception to this rule: when, by setting the release time to 19.00, the time available for maintenance (i.e. between the end of the MO and 19.00) is less than the duration of the maintenance, then the release time is set to end time minus the *reference duration*. Since the actual duration of the maintenance is not known at this point, we set the reference duration is the maximum duration of maintenance, i.e. $\sum_{k \in K} v_k$.

A symmetric set of rules prevails for the determination of the deadline moment.

The objective is adjusted to also minimize the number of maintenance teams required. The main goal remains to find a solution minimizing the number of maintenance activities during nighttime. Secondarily, we minimize the number of required maintenance teams. Note that the former objective is more important; hence, finding a solution with a lower number of maintenance activities during nighttime is preferred over finding a solution with a lower number of required maintenance teams.

### 4.2. MILP formulation for MSLCP

The MSLCP can now be formulated as a MILP model. In addition to the notation already introduced for the MLCP, we define the following decision variables.

- Let $z_{nmuij}$ be a binary variable equal to 1 if maintenance team $n$ on moment $m$ in shift $u$ performs the maintenance that was assigned to MO $j$ for RS unit $i$, and 0 otherwise.
- Let $\mu_{nmuij}$ be equal the duration of maintenance in MO $j$ of RS unit $i$ if that maintenance is assigned to team $n$ on moment $m$ in shift $u$; and 0 otherwise.
- Let $h_{nmu}$ be the start time of the moment $m$ for team $n$ in shift $u$.
- Let $w_{ij}$ be a binary variable equal to 1 if MO $j$ for RS unit $i$ is used for any maintenance.
- Let $\gamma_n \in \{0,1\}$ be a binary variable equal to 1 if team $n$ in shift $u$ is active and 0 otherwise.

Also, we use $\sigma_{uij}$ as a binary parameter equal to 1 if MO $j$ for RS unit $i$ is part of shift $u$. Note that each MO is part of exactly one shift, determined during preprocessing. The model can then be formulated as follows.

$$\min \left( \sum_{i\in I} \sum_{j\in J_i} \sum_{k\in K} x_{ijk}(1-d_{ij}) + \varepsilon \sum_{i\in I} \sum_{j\in J_i} \sum_{k\in K} x_{ijk} + \varepsilon_2 \sum_{n} \sum_{u} \gamma_{nu} \right) \tag{8}$$

subject to

$$1 \le \sum_{p\in V_{i0k}} x_{ipk} \qquad i\in I, k\in K \tag{9}$$

$$x_{ijk} \le \sum_{p\in V_{ijk}} x_{ipk} \qquad i\in I, j\in J_i, k\in K : e_{ij}+o_k \le T \tag{10}$$

$$x_{ijk} \le y^D_{l_{ij}} \cdot d_{ij} + y^N_{l_{ij}} \cdot (1-d_{ij}) \qquad i\in I, j\in J_i, k\in K \tag{11}$$

$$\sum_{k\in K} x_{ijk} v_k \le e_{ij} - s_{ij} \qquad i\in I, j\in J_i \tag{12}$$

$$\sum_{l\in L} y^D_l \le L^D_{max} \tag{13}$$

$$\sum_{k\in K} v_k \left( x_{ijk} - (1-z_{nmuij}) \right) \le \mu_{nmuij} \qquad n\in N_u, m\in M, u\in U, i\in I, j\in J_i \mid \sigma_{uij}=1 \tag{14}$$

$$\mu_{nmuij} \le \sum_{k\in K} v_k \left( x_{ijk} + (1-z_{nmuij}) \right) \qquad n\in N_u, m\in M, u\in U, i\in I, j\in J_i \mid \sigma_{uij}=1 \tag{15}$$

$$\sum_{i\in I} \sum_{j\in J_i} z_{nmuij} r_{ij} \sigma_{uij} \le h_{nmu} \qquad u\in U, n\in N_u, m\in \{1,\dots,\overline{M}-1\} \tag{16}$$

$$h_{nmu} \le \sum_{i\in I} \sum_{j\in J_i} (z_{nmuij} t_{ij} \sigma_{uij} - \mu_{nmuij}) \qquad u\in U, n\in N_u, m\in \{1,\dots,\overline{M}-1\} \tag{17}$$

$$h_{n,m+1,u} \ge h_{nmu} + \sum_{i\in I} \sum_{j\in J_i} \mu_{nmuij} \sigma_{uij} \qquad u\in U, n\in N_u, m\in \{1,\dots,\overline{M}-1\} \tag{18}$$

$$w_{ij} \ge x_{ijk} \qquad i\in I, j\in J_i, k\in K \tag{19}$$

$$w_{ij} \le \sum_{k\in K} x_{ijk} \qquad i\in I, j\in J_i \tag{20}$$

$$\sum_{n\in N_u} \sum_{m\in M} \sum_{u\in U} z_{nmuij} \sigma_{uij} = w_{ij} \qquad i\in I, j\in J_i \tag{21}$$

$$\sum_{i\in I} \sum_{j\in J_i} z_{nmuij} \sigma_{uij} \le 1 \qquad n\in N_u, m\in M, u\in U \tag{22}$$

$$\sum_{m\in M} \sum_{i\in I} \sum_{j\in J_i} (\gamma_{nu} - z_{nmuij}) \ge 0 \qquad n\in N_u, u\in U \tag{23}$$

$$\sum_{n} \gamma_{nu} \le \overline{N}_u \qquad u\in U \tag{24}$$

$$\sum_{l\in L} y^D_l \le L^D_{max} \tag{25}$$

and

$$x_{ijk} \in \{0,1\},\ y^D_l \in \{0,1\},\ g_{ij} \ge 0,\ w_{ij} \in \{0,1\} \quad i\in I, j\in J_i, k\in K, l\in L \tag{26}$$

$$z_{nmuij} \in \{0,1\},\ \mu_{nmuij} \ge 0,\ h_{nmu} \ge 0,\ \gamma_{nu} \in \{0,1\} \quad n\in N_u, m\in M, u\in U, i\in I, j\in J_i. \tag{27}$$

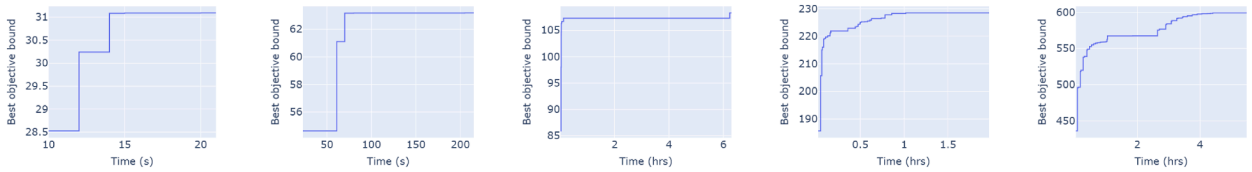
In the objective function (8), the first set of terms minimizes the number of nighttime maintenance activities. The second set of terms uses an arbitrarily small (in any case smaller than 1), but strictly positive penalty cost $\varepsilon$ to penalize maintenance activities in order to avoid unnecessary maintenance activities being performed. The third set of terms uses an even smaller, but strictly positive penalty cost $\varepsilon_2 \ll \varepsilon$ to penalize the number of maintenance teams used in order to avoid the use of unnecessary maintenance teams.

The first set of constraints, (9)–(13), corresponds to the part of the model that is responsible for the maintenance location choice and the assignment of maintenance activities to MOs (MLCP model). Constraints (9) and (10) enforce that intervals between successive maintenance activities are satisfied, the first assuring that the first maintenance activity is scheduled in time and the latter assuring that all subsequent maintenance activity are scheduled in time. Constraints (11) ensure that maintenance can only be executed at a location that is opened. Constraints (12) take account of the requirement that the duration of maintenance may not exceed the total time of an MO. The number of locations for daytime maintenance is restricted by constraint (13). The second set of constraints, (14)–(22), corresponds to the scheduling part of the model in which activities per locations are assigned to maintenance teams. Constraints (14)-(15) set the variables that represent the duration of assigned maintenance. Constraints (16)–(17) guarantee that the start moment is after the release time of the corresponding maintenance activities and before the latest start moment for the corresponding maintenance activities (i.e. the deadline minus the duration). Constraints (18) enforce that the start moments for one team are sufficiently far apart so that maintenance activities do not overlap. Constraints (19)–(20) determine whether an MO is used for maintenance, and if it is, Constraints (21) ensure that the maintenance job is assigned to exactly one moment. Constraints (22) make sure that each moment is used for at most one job. Constraints (23) establish that a team can only be used if it is 'active'. Constraints (24) restrict the number of maintenance teams that can be deployed. Constraints (26)-(27) define the ranges of the variables.

### 4.3. Exploratory results

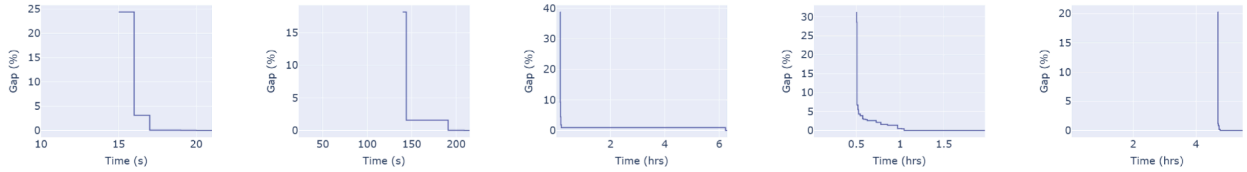
A first exploratory run of the MSLCP using the above MILP formulation demonstrates that solution times rapidly increase to several h for realistic problem instances. In this exploratory run we set the number of available maintenance teams equal to $N_u = 1$ for all shifts $u \in U$. The choices for the maximum number of maintenance locations for daytime maintenance $L^D_{\max}$, the planning horizon $T$, the values for the maintenance duration $v_k$ and the technical parameter $\varepsilon$ are as accounted for in Section 6.1. We use RS plan 1. We set $\varepsilon = 10^{-3}$ and $\varepsilon_2 = 10^{-6}$, which are, for the considered rolling stock plans, small enough to ensure that the first component of the objective function (8) is always larger than second part of the objective function and the second part is always larger than the third part of the objective function. The model is implemented using Python version 3.12 and solved using Gurobi version 10.0.0. The results were generated using a computer with Windows 10 Pro, 16.00 GB of RAM and with an Intel(R) Core(TM) i7-8750H CPU 2.20 GHz processor.

Fig. 2 shows the development of the MSLCP objective trajectory for various problem sizes (obtained by varying the number of trains $|I|$). For 10 and 20 trains (Fig. 2(a) and (b)), a solution is found within 30 s and 250 s, respectively, but for 30, 50 and 100 rolling stock units (Fig. 2(c)–(e)) we observe solve times of several h. These solve times




(a) 10 trains (b) 20 trains (c) 30 trains (d) 50 trains (e) 100 trains

**Fig. 2.** Development of the best bound in the MILP formulation of the MSLCP, for various problem instance sizes. The best objective bound is chosen as relevant metric as it provides a practical meaning, i.e. it represents the number of maintenance activities performed during nighttime.




(a) 10 RS units (b) 20 RS units (c) 30 RS units (d) 50 RS units (e) 100 RS units

**Fig. 3.** Development of the MILP gap in the MILP formulation of the MSLCP, for various problem instance sizes.

are too long considering the nature of the practical applications, where a solution is required within at most a few minutes, especially taking into account that realistic problem sizes usually include well over 100 rolling stock units. In the remainder of this work we use real rolling stock circulations that indeed contain well over 100 rolling stock units (i.e. even larger than the largest one presented here) and we use the same parameter settings if applicable.

Furthermore, we present the development of the MILP gap in Fig. 3. We see that, especially for the realistic scenario in Fig. 3(e), the MILP gap is high during a large part of the total solve time, indicating that stopping the optimization earlier would not provide a good solution. This further establishes the conclusion that the MILP formulation is not appropriate in practice. Hence, an alternative solution methodology is necessary, which we develop in the remainder of this work.

## 5. Solution methodology for MSLCP

The goal of the MSLCP is to find a solution to the MLCP that satisfies predetermined constraints regarding the available number of maintenance teams. In the previous section, we showed that a corresponding MILP formulation performs poorly (Section 4.3). To reach better performance, we develop an alternative solution approach based on Logic Based Benders' Decomposition (LBBD), that we discuss in this Section.

### 5.1. Activity planning problem (APP)

We decompose the MSLCP into a master problem and a sub problem. The master problem is the MLCP. The sub problem is the *Activity Planning Problem* (APP), which determines an optimal maintenance schedule minimizing the number of required teams, given a solution of the MLCP.

To define the APP we mostly reuse the notation and definitions given in Section 4, with a few simplifications, made possible because in this context the assignment of maintenance activities to maintenance opportunities and shifts is fixed. Hence, the set of jobs $Q$ to be scheduled is given.

We simplify the representation of $z$, $h$, and $\gamma$ by indexing this only with $nmq$, $nm$ and $n$, respectively: let $z_{nmq} \in \{0,1\}$ be a binary variable which equals 1 if and only if team $n$ at moment $m$ processes job $q$; let $h_{nm} \in \mathbb{R}$ be the start time of the moment $m$ for team $n$; and let $\gamma_n \in \{0,1\}$ be a binary variable which equals 1 if team $n$ is used for this schedule. Also the release time, deadline time and maintenance duration are only dependent on the job $q$: let $r_q \in \mathbb{R}$ be the release time of job $q$; let $t_q \in \mathbb{R}$ be the deadline time of job $q$ and let $v_q \in \mathbb{R}$ be the duration of job $q$. Since in the APP the maintenance activities in each job are known, the *reference duration* is in this case set to the more accurate *actual* duration of the maintenance activities assigned to this MO, leading to slightly different release and deadline times in corner cases. Moreover, the maximum number of maintenance teams is simplified to $\overline{N}$ and the associated set is $N = \{1, \dots, \overline{N}\}$.

The APP model is then formulated as follows.

$$\min \sum_{n \in N} \gamma_n \tag{28}$$

subject to

$$\sum_{q \in Q} z_{nmq} r_q \leq h_{nm} \leq \sum_{q \in Q} z_{nmq}(t_q - v_q) \qquad \forall n \in N, m \in M \tag{29}$$

$$h_{n,m+1} \geq h_{nm} + \sum_{q \in Q} z_{nmq} v_q \qquad \forall n \in N, m \in \{1, \dots, \overline{M} - 1\} \tag{30}$$

$$\sum_{n \in N} \sum_{m \in M} z_{nmq} = 1 \qquad \forall q \in Q \tag{31}$$

$$\sum_{q \in Q} z_{nmq} \leq 1 \qquad \forall n \in N, m \in M \tag{32}$$

$$\sum_{m \in M} \sum_{q \in Q} (\gamma_n - z_{nmq}) \geq 0 \qquad \forall n \in N \tag{33}$$

$$z_{nmq} \in \{0,1\}, \gamma_n \in \{0,1\}, \ h_{nm} \in \mathbb{R} \qquad \forall n \in N, m \in M, q \in Q \tag{34}$$

The objective (28) minimizes the number of teams necessary. Constraints (29) guarantee that the start moment is after the release time of the corresponding job and before the latest start moment for the corresponding job (i.e. the deadline minus the duration). Constraints (30) enforce that the start moments for one team are sufficiently far apart so that maintenance activities do not overlap. Constraints (31) ensure that every job is assigned to exactly one moment. Constraints (32) make sure that each moment is used for at most one job. Constraints (33) establish that a team can only be used if it is 'active'. Constraints (34) ensure that the integer decision variables are also binary.

### 5.2. MSLCP solution approach

To solve the MSLCP with better performance, our solution approach integrates the MLCP and APP in one framework using an approach called Logic-Based Benders' Decomposition (LBBD), which is

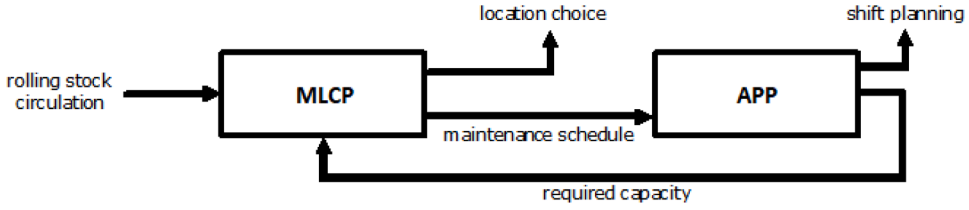



**Fig. 4.** Graphical representation of the MSLCP, demonstrating how it integrates the MLCP and the APP.

a generalization of the recognized method called Benders' Decomposition (Hooker, 2011). In railways, Benders' Decomposition-based approaches have been used in railway timetabling (Lamorgese et al., 2017; Trepat Borecka et al., 2022) and railway rescheduling problems (Keita et al., 2020; Lamorgese & Mannino, 2015; Lamorgese et al., 2016), but not yet for maintenance planning.

*Benders' decomposition* (BD) is a method proposed by Benders (1962) and aims to efficiently solve large-scale linear optimization problems by decomposing the complete problem into a master problem and a sub problem. First, the master problem is solved. Based on the solution of the master problem, a sub problem is identified and solved. Based on the solution of the sub problem, constraints (also called *cuts*) are added to the master problem, which is then solved again. This process continues in an iterative manner. Optimality is reached when the objective value of the master problem is equal to the objective value of the sub problem and the algorithm terminates. In classical BD, cuts are generated via a standard procedure using duality theory. However, in order to do so it requires a specific form for the sub problem (i.e. the complete problem needs to be formulated as one mixed integer program) and it requires that the sub problem be linear and continuous. LBBD does not require that the sub problem take a specific form, at the cost of the fact that it does not have a standard procedure to generate cuts.

In the current implementation of LBBD, the MLCP is defined as the master problem and the APP as the subproblem.

Fig. 4 visualises the cooperation between the MLCP and the APP to include capacity constraints. The maintenance schedule of the MLCP is used to determine the required capacity in the APP model. If the required capacity exceeds the available capacity, the information from the APP is used to add constraints to the MLCP and the MLCP is run again.

The MSLCP model repeatedly executes the following steps. First, an empty set of *cuts* is initialized. Second, the MLCP subject to the current set of all generated cuts is solved. Third, by solving APP, a candidate solution for a maintenance schedule, i.e. an assignment of maintenance activities to MOs, is generated. The algorithm terminates when the APP results in a feasible solution for all time shifts. In that case all constraints in the MLCP and all additional constraints handled by the APP are satisfied and an optimal solution has been determined. Also, it terminates when the current running time exceeds the predetermined maximum running time. Otherwise, it returns to solving MLCP again with newly generated cuts.

### 5.3. MSLCP algorithm

Let $U$ be the set of unique maintenance shifts and $\overline{N}$ the number of teams available at any location. The MSLCP iteratively solves the master and subproblem. Let $\rho^\kappa$ be the solution of the MLCP after the $\kappa$th iteration of the MSLCP (i.e. this corresponds to a maintenance schedule, which is an assignment of maintenance activities to MOs), $Q_{\rho^\kappa}(u)$ the set of jobs for shift $u$, given the solution of the MLCP $\rho^\kappa$, and $APP(Q)$ the objective value obtained after running the APP for the set of jobs $Q$. Use the notation $APP(Q) = \infty$ if the APP for the set of jobs $Q$ results in an infeasible solution, meaning the required capacity exceeds $\overline{N}$ maintenance teams. To describe the capacity required for a shift $u$, given a solution $\rho^\kappa$ of the master problem, the notation $APP\left(Q_{\rho^\kappa}(u)\right)$ is used.

If $APP(Q) = \infty$ for a given set $Q$, it can be concluded that the combination of jobs in the set $Q$ results in a violation of the maintenance location capacity. In this case, based on the set $Q$, cuts can be generated according to one of the procedures that are described in Section 5.4. A cut indicates a combination of jobs that results in an infeasible solution of the APP. Let $C(Q)$ be the set of cuts based on set $Q$. For any cut $A \in C(Q)$ it holds that $A \subseteq Q$ and $APP(A) = \infty$.

Each cut can be translated into a constraint of the MLCP in the following way. Consider a cut $A$. Since $A \subseteq Q$, every element in $A$ signifies a maintenance job which is notated as a tuple $(i, j, K)$ where $i$ is the rolling stock unit, $j$ is the corresponding MO and $K$ is the set of assigned maintenance activities. To include a cut $A$ in the MLCP, the constraint in Eq. (35) needs to be added to prevent the combination of jobs in the cut to show up in a next iteration of the MSLCP.

$$\sum_{(i,j,K)\in A} \sum_{k\in K} (1 - x_{ijk}) \geq 1 \tag{35}$$

Multiple cuts, for example the set of cuts $C(Q)$, can be added by adding the constraint from Eq. (35) to the MLCP for every cut $A \in C(Q)$.

Pseudo-code for the iterative procedure of the MSLCP is given in Algorithm 1. Here, $\kappa$ is index that tracks the current iteration, $C^*_\kappa$ is the set of cuts generated up to and including the $\kappa$th iteration, $C^*_0 = \emptyset$, $\ell_0$ is the start time of the algorithm, and $\ell$ be a parameter restricting the total computation time until the process terminates (if no optimal solution is found earlier).

**Algorithm 1** MSLCP iterative approach.

```
function MSLCP(ℓ)
    C*_0 ← ∅
    ℓ_0 ← current time
    κ ← 1
    while current time − ℓ_0 < ℓ do
        compute MLCP solution ρ^κ, subject to cuts in C*_{κ−1}
        C*_κ ← C*_{κ−1}
        for u ∈ U do
            if APP(Q_{ρ^κ}(u)) = ∞ then
                C*_κ ← C*_κ ∪ C(J_{ρ^κ}(u))
            end if
        end for
        if |C*_{κ−1}| = |C*_κ| then
            return ρ^κ as the optimal MLCP solution
        end if
        κ ← κ + 1
    end while
    return ρ^κ as the best found sub-optimal MLCP solution
end function
```

The algorithm starts by initializing $C^*_0$, $\ell_0$ and $\kappa$, after which the iterative loop starts. This loop first computes a solution to the MLCP subject to all cuts generated so far. Then, for each shift $u$ in which the required capacity exceeds the available capacity, cuts are generated. The process terminates if either an optimal MLCP solution is found, satisfying all constraints, or if the user-defined maximum running time is exceeded.

### 5.4. Cut generation

If a solution to the MLCP is found that violates the maintenance location capacity constraints, cuts are added to the MSLCP in order to constrain the solution space and prevent such a solution from showing up again. A cut is a set of jobs that cannot occur together since it would result in a violation of available capacity. Cuts result in a restriction of the solution space of the master problem. For a quick convergence of the algorithm, it is desirable to add cuts that are as restrictive as possible. In general, cuts with a smaller amounts of jobs are more restrictive than cuts with larger amounts of jobs. As an example, suppose that the set of maintenance jobs $\{A, B, C\}$ results in an infeasible solution but that the set of maintenance jobs $\{A, B\}$ results in an infeasible solution as well. Both sets of jobs would constitute a valid cut, but the latter set of jobs is smaller, hence more restrictive and as a result more efficient to add.

The remainder of this section proposes three different cut generation procedures: the naive method, the heuristic method and lastly the min-cut method, which is a more complex method that uses the structure of the problem.

#### 5.4.1. Naive cut generation

Let $Q$ be a set of jobs that results in a capacity violation, i.e. $APP(Q) = \infty$. Let $C(Q)$ be the set of cuts generated for this set of jobs. Since $Q$ results in an infeasible solution to the APP, this set itself can be added as a cut. Hence, $C(Q) = \{Q\}$.

#### 5.4.2. Heuristic cut generation

To generate smaller cuts compared to the naive procedure, the *heuristic cut generation procedure* is proposed. This method applies a procedure that is inspired by the principle of binary search (see for example Cormen et al., 2009, p.799).

Let $A$ be an initially empty set such that at any moment in the procedure, the jobs in $A$ result in a feasible solution, i.e. $APP(A) < \infty$. Let $B$ be a set of candidate jobs that, when added to the jobs in $A$, at any moment in the procedure results in an infeasible solution: $APP(A \cup B) = \infty$. The algorithm repeatedly splits $B$ into two halves, a left half $B_L$ and a right half $B_R$, and it computes $APP(A \cup B_L)$. If this results in an infeasible solution, i.e. $APP(A \cup B_L) = \infty$, then the set $B_R$ is discarded. In the subsequent iteration of the algorithm the set B of candidate jobs is reduced to $B_L$. If this results in a feasible solution, i.e. $APP(A \cup B_L) < \infty$, some jobs from $B_R$ still need to be added to achieve a 'just infeasible' solution. In this case, the jobs in $B_L$ are all included in the set $A$, and the remaining candidate jobs $B$ to decide on are the jobs $B_R$. The algorithm terminates when $|B| = 1$. Pseudo-code for the described procedure is given in Algorithm 2.

The following loop invariants hold (i.e. those expressions are true at the start and end of each iteration): $APP(A) < \infty$, meaning that the set of jobs in $A$ is feasible; and $APP(A \cup B) = \infty$, meaning that when the set of jobs in $B$ is added to the set of $A$, the resulting set of jobs is infeasible.

#### 5.4.3. Min-cut cut generation

In order to find more efficient cuts, the current section designs a procedure that aims to find cuts with a small amount of jobs, by making use of the specific structure of the problem. To this end, the *Relaxed Activity Planning Problem* (RAPP) is defined, which is a relaxation of the APP. In this research, the RAPP is developed for one maintenance team only, although it is expected that the approach can be generalized to multiple teams.

The benefit of the definition of the RAPP lies in the fact that any infeasible solution to the RAPP is also an infeasible solution to the APP. Recall that cuts need to be generated if the APP is infeasible (see Algorithm 1). To generate cuts according to the min-cut cut generation procedure, the RAPP is solved. If the RAPP turns out to be infeasible, the min-cut cut generation procedure described in this section can be used. If the RAPP turns out to be feasible, the min-cut cut generation procedure cannot be used and one needs to resort to other cut generation procedures.

**Algorithm 2** Heuristic cut generation.

```
function HEURISTIC CUT GENERATION(Q)
    A ← ∅
    B ← Q
    while |B| > 1 do
        B_L ← ∅
        h ← ⌈½|B|⌉
        for i ← 1 to h do
            pick random j ∈ B
            B_L ← B_L ∪ {j}
            B ← B \ {j}
        end for
        B_R ← B
        if APP(A ∪ B_L) = ∞ then
            B ← B_L
        else
            A ← A ∪ B_L
            B ← B_R
        end if
    end while
    return A ∪ B
end function
```

The RAPP is a relaxation of the APP in two ways. First, the RAPP discretizes the planning horizon to a set of *instants*, which are integer minutes, meaning that jobs can only start and end on integer minutes and job durations should be specified as integers. In the practical context of the railway industry, this is not expected to be problematic since rolling stock units are usually planned per minute. Second, the RAPP allows for preemption of jobs. This means that, unlike in the APP, the work on a job does not need to be performed uninterruptedly.

The RAPP attempts to assign jobs to as many distinct instants as its duration. This problem can be viewed as a variant of the bipartite matching problem (Cormen et al., 2009, p. 732), where jobs need to be matched to instants, with this difference that jobs in the current problem usually need to be matched to multiple instants instead of only one. The bipartite matching problem is often modeled as a maximum flow problem (Ford & Fulkerson, 1956), for which efficient solution algorithms exist (Cormen et al., 2009, p. 732–735). Following this approach, the current research defines the RAPP as a maximum flow problem.

Let $Q$ be the set of jobs, and let $r_q, t_q, v_q$ be the release time, deadline time and duration for job $q \in Q$, respectively, defined in minutes. It is assumed that the duration $v_q$ is integer. Let $P_q$ be the set of instants at which job $q$ is available. This comprises all minutes between $r_q$ and $t_q$ and can be expressed as follows: $P_q = \{x \in \mathbb{N} : \lfloor r_q \rfloor \leq x \leq \lceil t_q \rceil\}$. Let $P$ be the set of all time instants at which at least one job is available, $P = \cup_{j \in J} P_j$. Observe that the RAPP uses discrete time moments (in full minutes) instead of real-valued time moments. Since, in the railway industry, the release and deadline times are usually given in minutes, this is not restrictive.

*Step 1: Find the maximum flow..* Define a source $s$ and a sink $t$ and let $E_G$ be a set of directed edges with capacity $c_e$ for edge $e \in E_G$. Let $G = (N_G, E_G)$ be a directed flow graph. The set of nodes $N_G$ is defined by $N_G = \{s \cup Q \cup P \cup t\}$. The set of directed edges $E_G$ contains a directed edge $e$ (1) from node $s$ to node $j$ for all $q \in Q$ with capacity $c_e = v_q$; (2) from node $q$ to $p$ for all $q \in Q$ and $p \in P_q$, with unit capacity $c_e = 1$ (this implies that, for each job, there is a directed edge to each instant at which it is available); (3) from $p$ to $t$ for all $p \in P$, with unit capacity $c_e = 1$.

Once the flow graph has been determined, determine the maximum flow through the flow graph $G$ from the source $s$ to the sink $t$ and

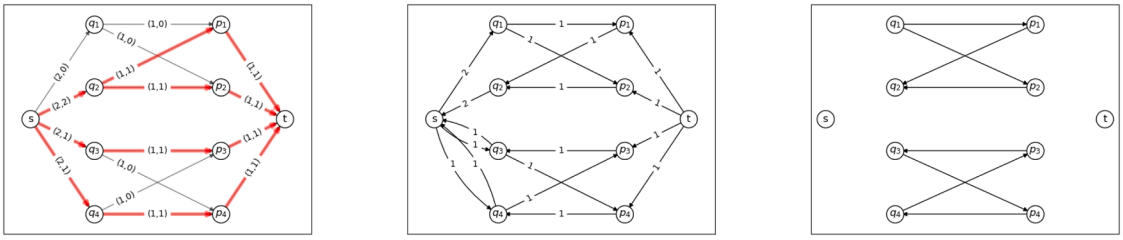



**Fig. 5.** Left: Flow graph $G$ corresponding to the RAPP model. Middle: Residual graph $R$ corresponding to the infeasible solution. Right: Reachable Components graph $H$ separating the reachable components that are reachable from $s$.

denote the resulting flow through each edge $e \in E_G$ by $f_e$. The RAPP is considered to be feasible if and only if the value of the maximum flow equals the sum of all durations, or, equivalently, equals the sum of all capacities on edges departing from $s$, i.e. if and only if

$$\sum_{e \in E_G} f_e = \sum_{q \in Q} v_q = \sum_{e \in \{(s,v) \in E_G : v \in Q\}} c_e. \tag{36}$$

The satisfaction of the aforementioned condition(s) represents the fact that all jobs have been completely scheduled.

To illustrate the RAPP cut generation, an example instance is presented where one maintenance team has to perform four jobs: $Q = \{q_1, q_2, q_3, q_4\}$. Jobs $q_1$ and $q_2$ can both be performed at instants $p_1$ and $p_2$ (i.e. $P_1 = P_2 = \{q_1, q_2\}$ and jobs $q_3$ and $q_4$ can be performed at instants $p_3$ and $p_4$ (i.e. $P_3 = P_4 = \{q_3, q_4\}$). As a result, the set of all instants $P = \{p_1, p_2, p_3, p_4\}$. All jobs have a duration of 2 instants. Fig. 5 (left) pictures the associated flow graph with the assigned flow on each of the edges in $G$. Edges $e$ are annotated as $(c_e, f_e)$: the first index represents the edge capacity and the second index represents the assigned edge flow. Red-colored edges (color: see online) represent edges through which a strictly positive flow is assigned. The maximum flow is 4, whereas the sum of all job durations is 8, meaning that by Eq. (36) the RAPP is not feasible. The remainder of the current section discusses how this infeasible solution can be used to generate cuts.

*Step 2: Determine the residual graph.*. To find jobs that cannot occur together, the concept of *minimum cuts* from graph theory is used. The capacity of the minimum cut is equal to the value of the maximum flow, and the cut itself provides information about the edges that form a bottleneck in the current graph (Taha, 2011, p. 269). To determine the minimum cut, the concept of *residual graph* is used (Cormen et al., 2009, p. 716). It offers information on how the flow between edges can be changed and represents the amount of possible additional flow through each edge. It may also contain so-called *reverse edges*, that represent the possibility of canceling already assigned flow. To formally define the concept of the residual graph, let $R$ be a directed graph with the same nodes as $G$ and let its set of edges be denoted by $E_R$, that is, $R = (N_G, E_R)$. Then, the set of edges $E_R$ is constructed as follows. For every edge $e \equiv (u, v) \in E_G$, there is an edge $e' \equiv (u, v) \in E_R$ with capacity $c_{e'} = c_e - f_e$ if and only if $c_e - f_e > 0$; and there is an edge $e'' \equiv (v, u) \in E_R$ with capacity $c_{e''} = f_e$ if and only if $f_e > 0$. The nodes that are reachable from $s$ comprise the minimum cut, and the edges connecting one of these nodes to one of the unreachable ones together constitute the bottleneck.

Fig. 5 (middle) displays the residual graph $R$ corresponding to the earlier example. Each directed edge represents the residual capacity between two nodes, if positive. Take, for instance the positive residual capacity of 2 from $s$ to $q_1$: this signifies that an additional flow can be assigned from $s$ to $q_1$ (corresponding to the situation in which $q_1$ is scheduled). However, in this case, the flow must continue to $p_1$ and $p_2$ (meaning that $q_1$ is scheduled during $p_1$ and $p_2$). This can only be achieved if already assigned flow to $p_1$ and $p_2$ flows back to $q_2$ (signifying that $q_2$, which was formerly scheduled at $p_1$ and $p_2$, is not scheduled anymore) and from there flow further back to the source $s$. The fact that there apparently exists a path from $s$ via $q_1$, $p_1$ and $q_2$ back to $s$ is an important observation: it signifies that $q_1$ and $q_2$ are conflicting. This, in turn, means that $q_1$ and $q_2$ cannot be scheduled together and can be added as a cut. In fact, all jobs on every path starting from $s$ and returning to $s$ constitute an infeasible combination of jobs.

*Step 3: Define the Reachable Components graph.*. To formalize the idea of conflicting jobs, the *Reachable Components graph* $H$ is introduced. Its aim is to separate components that define different combinations of jobs, each of which cannot occur together (i.e. result in an infeasible solution of the RAPP). Let $H$ be a directed graph and let it have the same nodes as $G$ and with the set of edges $E_H$, i.e. $H = (N_G, E_H)$. Let $E_H$ contain all edges in $R$ that are not connected to the source $s$ or sink $t$, that is, $E_H = \{(u, v) \in R : u \notin \{s, t\}, v \notin \{s, t\}\}$. Let $D(F, n)$ be the set of all nodes reachable in some graph $F$ starting from some node $n$ (also called the *descendants* of $n$ in $F$). This set of reachable nodes can be obtained efficiently by the application of a depth-first search (Cormen et al., 2009, p. 603–606).

From this, a set of cuts can be determined. Note that all separate sets of reachable nodes can be obtained by starting at some job $j \in J$ that is reachable from $s$ in $R$ and obtaining all jobs among its descendants. In other words, for all $q \in Q : (s, q) \in R$ the set $C_q = \{q \cup (D(H, q) \cap Q)\}$ comprises a set of jobs that cannot occur together. These jobs result in an infeasible RAPP solution and, as a consequence, in an infeasible APP solution; hence, they can be added as a cut.

To demonstrate the process of the determination of these cuts, return once again to the previous example. Fig. 5 (right) presents the graph $H$ with two different components. In $R$, the nodes $q_1, q_3$ and $q_4$ are reachable from $s$. Hence, the cuts generated in this way are $\{q_1, q_2\}, \{q_3, q_4\}$ and $\{q_4, q_3\}$. This shows that $q_1$ and $q_2$ cannot occur together, and similarly $q_3$ and $q_4$ cannot occur together.

*Step 4: Cut set post-processing.* All cuts according to the above described procedure can be added to the MLCP, but some of these may be superfluous. First, the same cuts may be generated more than once. Second, some cuts may be generated while a more specific cut is also generated: for example, consider the generation of two cuts, the first with jobs $X, Y$ and $Z$ and the second with jobs $X$ and $Y$. The latter makes the former redundant.

To remove redundant cuts, a straightforward procedure is applied that iteratively adds cuts only if it is not a superset of a more efficient cut that was already added. To this end, let $C$ be the set of all cuts generated by the RAPP and let $\tilde{(C)}$ be the set of cuts with all redundant cuts from $C$ removed. Algorithm 3 gives pseudo-code for this procedure.

### 5.5. Demonstration of the algorithm

To illustrate the MSLCP, below an explanatory instance is provided in which the initial maintenance location capacity is violated, but where the MSLCP finds a feasible solution in a next iteration. The naive cut generation method is used to generate cuts.

Table 2 shows four jobs that need to be performed during this particular shift according to the MLCP solution. The duration of job 1 is

**Algorithm 3** Remove redundant cuts after min-cut cut generation.

```
function REMOVE REDUNDANT CUTS(C)
    sort C by the cardinality of all its elements c ∈ C
    C̃ ← ∅
    for c ∈ C do
        add ← true
        for c̃ ∈ C̃ do
            if c ⊇ c̃ then
                add ← false
            end if
        end for
        if add = true then
            C̃ ← C̃ ∪ {c}
        end if
    end for
    return C̃
end function
```

**Table 2**
Assigned jobs before and after an iteration of the MSLCP algorithm.

| job | release | deadline | time | iteration 0 | iteration. 1 |
|---|---|---|---|---|---|
| 1 | 9:49 | 10:48 | 0.5 | x | x |
| 2 | 13:12 | 16:48 | 1 | x | x |
| 3 | 13:22 | 14:48 | 1 | x | x |
| 4 | 13:22 | 14:48 | 1 | x | |

**Table 3**
Final activity planning after running the MSLCP.

| job | n | start time | end time |
|---|---|---|---|
| 1 | 1 | 9:49 | 10:19 |
| 2 | 1 | 15:48 | 16:48 |
| 3 | 1 | 13:48 | 14:48 |

0.5 h, the duration of jobs 2, 3 and 4 is 1 h. It can be seen that jobs 3 and 4 both need to be performed between 13:22 and 14:48; this cannot be performed by one maintenance team. In the naive cut generation method, these four jobs together are added as a cut, which makes sure that in a next iteration of the MSLCP, not all jobs 1–4 can be performed anymore. In the next iteration job 4 has disappeared, meaning that the MLCP solution in the second iteration does not assign maintenance to the maintenance shift corresponding to job 4 anymore. Clearly, the new set of jobs can be performed by one team, and hence the APP results in a feasible solution, shown in Table 3. Since a solution has been found that satisfied the capacity constraints, the MSLCP terminates.

## 6. Experimental results

The current section investigates the performance of the MSLCP model on real-life instances. It considers first smaller-scale instances to investigate the ability to find an optimal solution, and second a larger-scale instance to investigate how quickly the model is able to converge to a solution and its applicability in practice.

### 6.1. Scenario set-up

The MSLCP framework is demonstrated on realistic scenarios from the Dutch railways. The problem instance considered in the current section uses a rolling stock circulation originating from the main Dutch railway operator, Netherlands Railways (NS). These RS plans describe rolling stock movements on the Dutch railway network.

This research uses 10 RS plan data sets, for ten consecutive periods between 2017 and 2019. Each RS plan describes a period of 8–10 weeks. These RS plans have comparable sizes, as the number of RS units in these RS plans ranges from 134 to 138. These data sets are the same as those that have been used by Zomer et al. (2021). For reasons of confidentiality, the actual data could not be provided, but synthetic data were made available instead.

In particular, 4 rolling stock types are considered: ICM4, DDZ4, DDZ6 and DD-AR3. These rolling stock types are used mainly for long distance intercity services. This comprises a total of approximately 140 rolling stock units, depending on the specific data set.

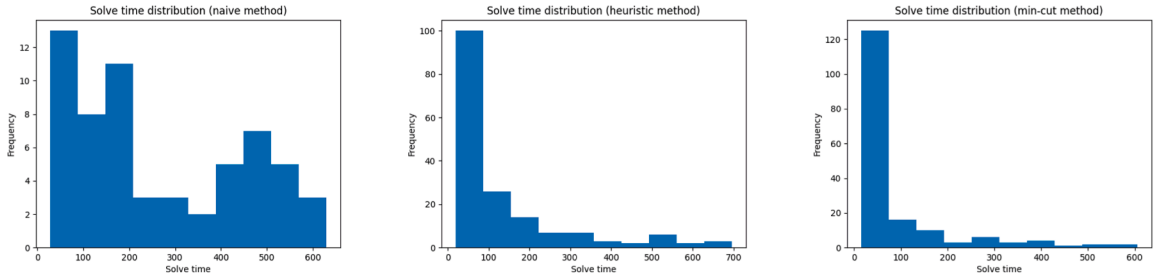
The planning horizon is set to 7 days, equal to the total number of days in the input data. The set of nighttime maintenance locations $L^N$ and the set of potential daytime maintenance locations $L^D$ are assumed to be equal to the set of all locations in the BDu. It is assumed that 5 locations can be opened for daytime maintenance at maximum, i.e. $L^D_{\max} = 5$. It is assumed that all locations are open for nighttime maintenance, i.e. $y^N_l = 1 \forall l \in L$, which corresponds to the current practice. Two maintenance types are included, maintenance type A having a duration of 30 m and an interval of 24 h, maintenance type B having a duration of 60 m and an interval of 48 h. Rolling stock units are assumed to be as-good-as-new at the start of the planning horizon, i.e. $b_{ik} = 0$ for all $i \in I, k \in K$. The technical parameter $\varepsilon$ has a value of $\varepsilon = 0.001$, which is, for the considered rolling stock plans, small enough to ensure that the first component of the objective function (6) is always larger than and hence dominant over the second part of the objective function. It is assumed that at each maintenance shift (at each location, on each day), one maintenance team is available, i.e. $\overline{N} = 1$. Note that this assumption is actually necessary for the min-cut cut generation procedure, which is only defined for one maintenance team.

Only the capacity of daytime maintenance shifts is considered, while the capacity of nighttime maintenance shifts is ignored. This choice is reasonable in the light of the gradual introduction of a policy of daytime maintenance in practice, where capacity for daytime maintenance at first is limited. The set of shifts $S$ and the maximum running time of the entire procedure $\ell$ are dependent on the scenario used and are discussed below.

The experimental set-up consists of three parts. First, we aim to reduce the capacity violation of single shifts. We investigate 3 different *cut generation variants*. We apply the heuristic cut generation method using 15 cuts, after an exploratory analysis showed that this number seemed to be a good choice. The set $S$ contains one shift each time. The running time is restricted at $\ell = 600$ s. The most important performance indicators are the number of shifts that could be solved within the maximum running time, and the actual running time of these shifts.

Second, we investigate hard shifts, i.e. shifts from the first part that could not be solved within the given time. Here, we allow more running time and investigate whether, given more time, a solution without violations can be found. Similar to the first part, the set of shifts $S$ contains one shift at a time. The running time in these cases is restricted to $\ell = 10$ h. The most important performance indicator in this case is the time it takes to reach an optimal solution (i.e. where the capacity violation is solved).

Third, we take a more practical perspective in a case study where we demonstrate that the MSLCP can also be used to solve multiple shift violations at the same time. This case study accounts for the fact that, in practice, reaching an optimal solution may take too much computation time. In particular, a sub-optimal solution with limited capacity violations obtained quickly may be preferred over an optimal solution without any violations taking excessive computation time. In this set-up, the set of shifts $S$ contains maintenance shifts for all possible combinations of maintenance location and date in the planning horizon. The running time is restricted at $\ell = 2$ h. The most important performance indicator is the number of shifts for which the required capacity exceeds the available capacity, i.e. the number of capacity violations, as a function of time and as a function of the number of iterations. It has been verified that a solution to the MLCP can be obtained in which all maintenance is performed during nighttime is feasible. Hence, if one would




(a) Naive method (b) Heuristic method (c) Min-cut method

**Fig. 6.** Solve time distribution of all solved shifts, for all three cut methods, for time horizon $T = 7$ days.

**Table 4**
Performance of various cut generation methods in various RS plans, for time horizons $T = 3$ and $T = 7$ days. The method that results in the highest proportion of solved shifts is printed in bold.

| | | | naive | | heuristic | | min-cut | |
|---|---|---|---|---|---|---|---|---|
| RS plan | $T | # shifts | violated | solved | violated | solved | violated | solved |
| 1 | 3 | 15 | 9 | 4 (44 %) | **9** | **7 (78 %)** | **9** | **7 (78 %)** |
| | 7 | 34 | 21 | 4 (19 %) | 24 | 17 (71 %) | **24** | **19 (79 %)** |
| 2 | 3 | 15 | 8 | 5 (63 %) | **9** | **7 (78 %)** | 8 | 5 (63 %) |
| | 7 | 35 | 20 | 5 (25 %) | 21 | 19 (90 %) | **22** | **20 (91 %)** |
| 3 | 3 | 15 | 10 | 6 (60 %) | **8** | **7 (88 %)** | 8 | 5 (63 %) |
| | 7 | 35 | 21 | 4 (19 %) | **21** | **20 (95 %)** | 20 | 19 (95 %) |
| 4 | 3 | 15 | 11 | 5 (45 %) | 10 | 8 (80 %) | **11** | **9 (82 %)** |
| | 7 | 35 | 23 | 5 (22 %) | **21** | **18 (86 %)** | 22 | 17 (77 %) |
| 5 | 3 | 15 | 11 | 6 (55 %) | 11 | 10 (91 %) | **12** | **11 (92 %)** |
| | 7 | 35 | 26 | 7 (27 %) | **24** | **23 (96 %)** | 26 | 24 (92 %) |
| 6 | 3 | 15 | 11 | 5 (45 %) | **10** | **9 (90 %)** | **10** | **9 (90 %)** |
| | 7 | 35 | 27 | 7 (26 %) | 26 | 25 (96 %) | **27** | **26 (96 %)** |
| 7 | 3 | 15 | 11 | 2 (18 %) | **9** | **7 (78 %)** | 9 | 6 (67 %) |
| | 7 | 35 | 25 | 5 (20 %) | **13** | **9 (69 %)** | 14 | 9 (64 %) |
| 8 | 3 | 15 | 8 | 5 (63 %) | **7** | **6 (86 %)** | 7 | 5 (71 %) |
| | 7 | 35 | 16 | 7 (44 %) | **13** | **11 (85 %)** | 14 | 11 (79 %) |
| 9 | 3 | 15 | 7 | 4 (57 %) | **6** | **6 (100 %)** | 7 | 6 (86 %) |
| | 7 | 35 | 23 | 7 (30 %) | **18** | **17 (94 %)** | 18 | 15 (83 %) |
| 10 | 3 | 15 | 11 | 7 (64 %) | 8 | 5 (63 %) | **11** | **8 (73 %)** |
| | 7 | 34 | 24 | 9 (38 %) | 23 | 11 (48 %) | **22** | **12 (55 %)** |

allow for enough computation time, the number of capacity violations would converge to zero with certainty.

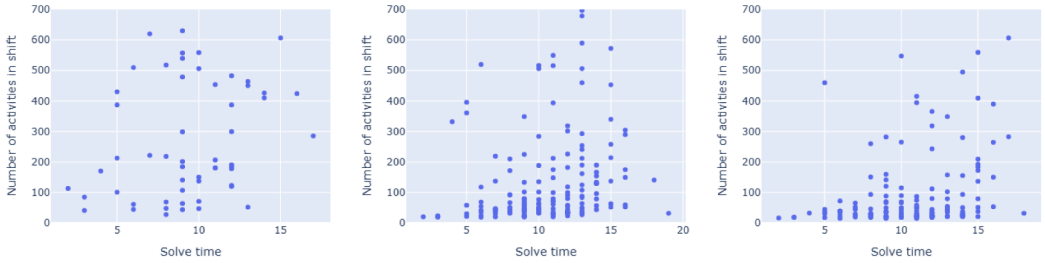
The MSLCP, MLCP, APP and RAPP are implemented implemented using Python and solved using Gurobi. For the implementation of the RAPP, the package NetworkX (Hagberg et al., 2008) is used. The corresponding maximum flow problem is solved using the preflow-push algorithm (see e.g. Cormen et al. (2009, p. 765)), that is included in the implementation of NetworkX. An implementation code of the MSLCP model is provided by Zomer et al., 2020. For reasons of confidentiality, the actual data could not be provided, but synthetic data is made available instead.

### 6.2. Results for single shifts

In the single-shift experiments, we allow for 600 s of running time for each experiment. We perform experiments of all 10 RS plans, for two different planning horizons. Within each run, we have various shifts for which the capacity is initially violated and this number varies per instance. We determine whether a shift capacity is violated or not by running the APP, maximizing its runtime by 15 s. For each shift with violated capacity, we apply the MSLCP, attempting to find a solution in which the capacity violation for this specific shift is solved.

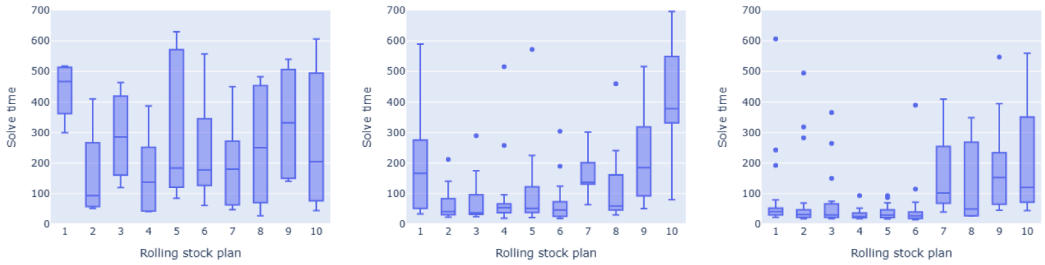
Table 4 shows the results for single shift scenarios. It shows for each RS plan and each time horizon, how many shifts the instance contains, for how many of these the capacity was initially violated, and for how many of these shifts the violation was eventually solved by each cut generation variant. Note that the number of initially violated shifts varies slightly across cut generation variants; this is a result of the fact that the runtime of the APP, which is used to identify initial violations, was restricted and hence did not find an optimal solution in all cases. For $T = 3$ ($T = 7$), around 15 (35) shifts were present. And typically, around 9 (21) violations were observed, respectively. It becomes clear that both the heuristic and min-cut cut generation variants outperform the naive method. Also, there is no clear benefit of the heuristic method compared to the min-cut method: both solve approximately the same number of shifts. Looking at different time horizons, in the majority of the cases, the naive method solved equal (or sometimes marginally more) violated shifts when increasing the time horizon. Instead, both heuristic and min-cut were capable of solving significantly more (often more than 100 % more) when increasing the horizon, and thus, showing stronger performance of the 2 cut generation methods.

Next, let us examine the computational performance of each of the methods. Fig. 6 displays the distribution of the solve time across all solved shifts (i.e. shifts for which the capacity violation could not be solved within the given time are excluded), across all RS plans. The majority of shifts is solved within 100 s by all methods, while a proportion of shifts still requires more than 100 s. In particular, the proportion is significantly smaller for the naive method: only 13 out of 60 (21.7 %) solved shifts were solved within 100 s (47 were solved in more than 100 s, and another 166 are not solved within the time limit at all), compared to 109 out of 172 (63.4 %) for the heuristic method and even 134 out of 170 (78.8 %) for the min-cut method. The heuristic method and the min-cut method are competitive, although the number of shifts that could be solved quickly is somewhat larger for the min-cut method.



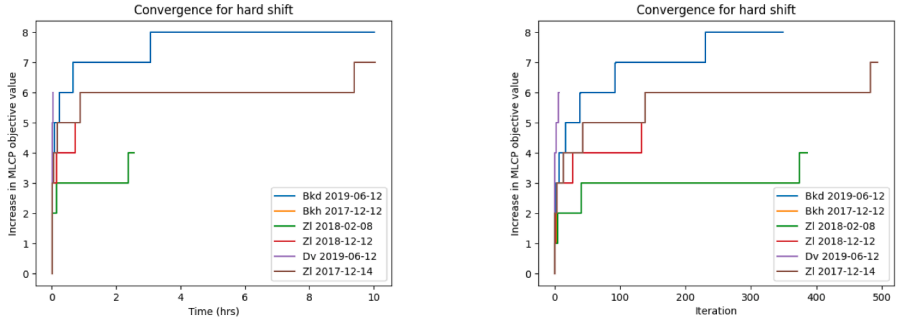

(a) Naive method (b) Heuristic method (c) Min-cut method

**Fig. 7.** Solve time of all solved shifts by the corresponding number of activities in those shifts, for all three cut methods.




(a) Naive method (b) Heuristic method (c) Min-cut method

**Fig. 8.** Solve time of all solved shifts by the corresponding rolling stock plan, for all three cut methods.




(a) Convergence by elapsed time (b) Convergence by current iteration

**Fig. 9.** Convergence of the MSLCP in the single-shift scenario for hard shifts. For each cut generation variant, the evolution of the value of the MLCP is displayed as a function of elapsed time (Fig. 9(a)) and as a function of the current iteration (Fig. 9(b)).

In order to gain more understanding into the factors contributing to the computation time, Fig. 7 displays the solve time of all solved shifts by the number of activities that those shifts contain. Even though the variation in solve time across those shifts is high, there is a clear tendency that shifts with higher numbers of activities are harder to solve than shifts with lower number of activities. Similarly, Fig. 8 displays the solve time of all solved shifts by the rolling stock plan. Shifts in some specific rolling stock plans take shorter times to solve (e.g. rolling stock plans 3, 4, 5, and 6) while shifts in other rolling stock plans take longer (e.g. rolling stock plans 9 and 10), even though these rolling stock plans contain approximately the same number of rolling stock units, forming an indication that structure of a rolling stock plan influences the computational performance. In both cases, the naive method shows more variation (Figs. 7(a) and 8(a)), hence the above relation is especially visible for the more sophisticated heuristic method and min-cut method.

To further describe the computational performance, Table 5 shows the mean solve time, like in Fig. 6 averaged over all solved shifts, but disaggregated by RS plan and time horizon. Only solved shifts (i.e. those shifts for which the capacity violation was solved within 600 s) are included; since not all methods solve an equal number of shifts (cf. Table 4), the number of shifts contained within the average figures differs for each figure. The results confirm that the naive method is outperformed by the heuristic method and the min-cut method in all cases. Moreover, it becomes clear that in most cases the performance of the min-cut method is better than the heuristic method. Also, the length of the time horizon has a significant impact on the average solve time: the increase of the time horizon from 3 to 7 days (i.e. increase by a factor 2.3) results in an increase in computation time. For example, for RS plan 1 for the min-cut method, the increase from 27.6 s to 86.4 s corresponds to a factor 3.1 increase, i.e. more than proportional

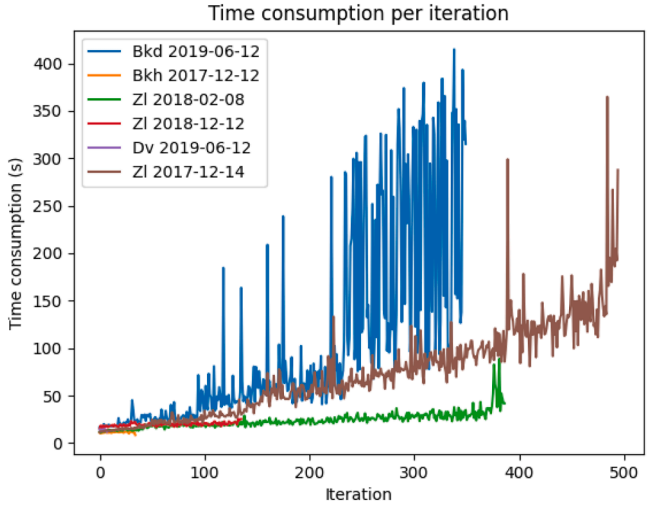



**Fig. 10.** Computation time of the MLCP in seconds, per iteration of the MSLCP for the heuristic cut generation version with 15 cuts per iteration, in an extended run of 10 h.

**Table 5**
Time consumption of various cut generation methods in various RS plans, in seconds, for time horizons $T = 3$ and $T = 7$ days.

| RS plan | $T$ | naive | heuristic | min-cut |
|---|---|---|---|---|
| 1 | 3 | 70.5 | **25.4** | 27.6 |
| | 7 | 239.2 | 210.1 | **86.4** |
| 2 | 3 | 37.3 | **32.1** | 32.3 |
| | 7 | 199.8 | **46.0** | 50.2 |
| 3 | 3 | 49.2 | **25.3** | 31.4 |
| | 7 | 240.7 | 44.5 | **34.4** |
| 4 | 3 | 76.6 | 49.7 | **26.8** |
| | 7 | 222.7 | 96.3 | **62.0** |
| 5 | 3 | 61.8 | 15.6 | **13.1** |
| | 7 | 248.3 | 52.3 | **37.3** |
| RS plan | $T$ | naive | heuristic | min-cut |
| 6 | 3 | 64.5 | 14.4 | **12.8** |
| | 7 | 250.5 | 40.4 | **31.1** |
| 7 | 3 | 96.8 | **38.7** | 44.5 |
| | 7 | 233.9 | 111.5 | **78.5** |
| 8 | 3 | 47.2 | **24.9** | 30.0 |
| | 7 | 124.4 | **47.5** | 57.2 |
| 9 | 3 | 36.3 | **3.2** | 12.7 |
| | 7 | 229.0 | 80.4 | **78.7** |
| 10 | 3 | 50.4 | 37.0 | **33.9** |
| | 7 | 202.7 | 274.2 | **149.8** |

compared to the time horizon length increase. The factor by which the time consumption increases for the heuristic and min-cut method has two outliers, for RS plans 1 and 9. Apart from these outliers the average factor of increase for the naive method is 3.8 (significantly more than proportional), for the heuristic method is 2.9 (more than proportional) and for the min-cut method is 2.3 (approximately proportional). Hence, the time consumption of the min-cut method is most stable under increasing time horizons.

In addition, we mention that there are also differences between RS plans. For example, RS plan 4 seems to be harder to solve than RS plans 1–3: both the heuristic and min-cut method have a higher average solve time in this case. In other words, not all RS plans are equally hard to solve. Also, there are some extreme observations where the min-cut method significantly outperforms the heuristic method: e.g. RS plans 7 and 10 for a time horizon of 7 days. The min-cut method seems to be more stable in its performance.

Lastly, recall that the goal of this paper is to find a model that moves maintenance activities from nighttime to daytime. Table 6 shows the minimum, average and maximum number of maintenance activities performed during daytime, for those shifts that were solved without violations, for various RS plans and time horizons $T$. Some daytime shifts contain up to 16 maintenance activities, which implies reduction of required nighttime capacity. Furthermore, no clear differences between cut methods are identified. The shifts that could only be solved by the naive method (and are typically 'simpler' to solve) do not necessarily contain less activities. Moreover, some differences between RS plans are found. For example, in RS plan 9 significantly less activities could be moved to daytime than in other RS plans. Furthermore, the chosen time horizon does not seem to systematically lead to higher or lower numbers of maintenance activities performed during daytime. Hence, there is no evidence that a choice for a shorter time horizon leads to less daytime maintenance activities. Note that from a practical point of view, it is interesting to investigate the total number of maintenance activities that can be moved from nighttime to daytime, as this results in the reduction of capacity issues. Detailed experiments on this matter have been presented in the research that we build upon by Zomer et al. (2021).

### 6.3. Results for single shifts: Focusing on hard shifts

In an attempt to find an optimal solution for hard shifts that could not be solved to optimality in the first part, the MSLCP procedure has been run for a longer period of time. We investigate six randomly chosen unsolved shifts. For this attempt, the heuristic method using 15 cuts was chosen as it showed somewhat better performance in the number of violations in Section 6.2.

The evolution trajectory of the MLCP objective value is given in Fig. 9, both by time and by iteration. The y-axis represents the increase in MLCP objective value compared to the initial solution (still containing capacity violations). For four of the investigated shifts, an optimal solution (without capacity violations) was found within the time limit: Bkh 2017-12-12, Zl 2018-02-08, Zl 2018-12-12, Dv 2019-06-12. The computation time needed for solving these was less than 1 h for 3, and as much as 2.5 h for the last one. The increase in objective value varies between 2 (for Bkh 2017-12-12) and 6 (for Dv 2019-06-12), corresponding to the increase in nighttime maintenance activities in the optimal solution compared to the initial solution. The number of iterations varies between 8 (for Dv 2019-06-12) and 386 (for Zl 2018-02-08). For the other two shifts, Bkd 2019-06-12 and Zl 2017-12-14, even the extended time horizon was not enough to find an optimal solution, they terminated after 10 h limit. Note that for the former four shifts, the optimal objective value is now known, whereas for the two shifts for which the capacity violation is not yet solved, the final objective value reached is a lower bound to the optimal value.

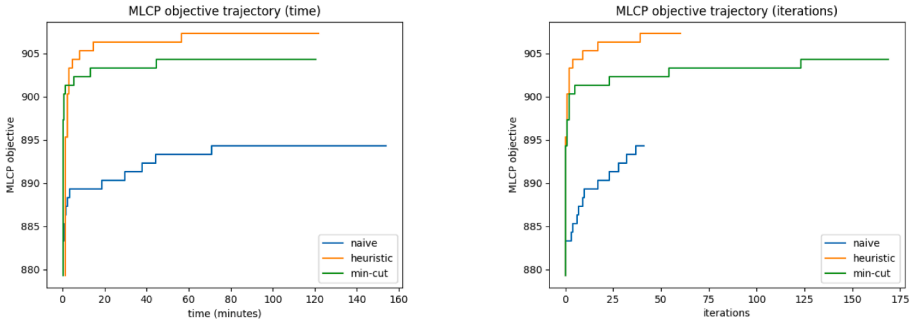
Fig. 10 presents the running time per iteration as a function of the current iteration. It shows that the running time of the MLCP (as well as its variance) increases for later iterations. The largest contributor to this higher computation time is, by far, the MLCP. Due to the added cuts, the MLCP becomes increasingly constrained and solving it becomes increasingly difficult.

### 6.4. Case study: Solving all capacity violations simultaneously for multiple shifts

The capacity constraint for this single maintenance shift appeared to be highly complicating for some shifts but much easier in other shifts. In many practical contexts, practitioners will aim to solve capacity violations across all shifts, or at least for as many shifts as possible. The current section demonstrates on a single case that the MSLCP can also be applied to all shifts simultaneously. In this example, the all-shifts setup is run with $T = 7$ and with RS plan 2. In this instance, 25 out of 35 shifts had a capacity violation (i.e. required more than 1 maintenance team).

**Table 6**
Number of maintenance activities performed per shift during daytime.

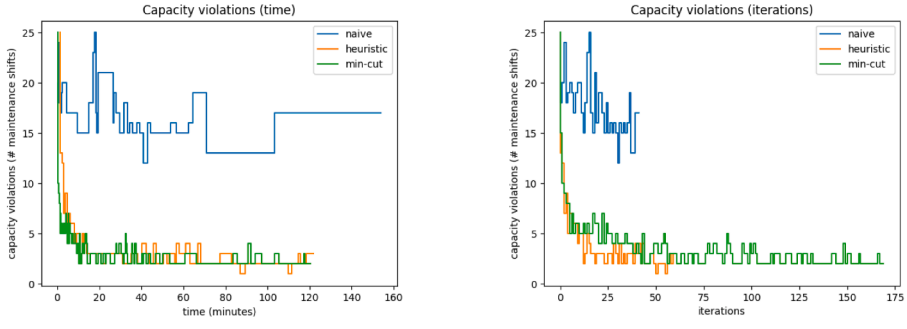
| | | naive | | | heuristic | | | min-cut | | |
|---|---|---|---|---|---|---|---|---|---|---|
| RS plan | $T$ | min | avg. | max | min | avg. | max | min | avg. | max |
| 1 | 3 | 3 | 10.1 | 15 | 3 | 10.6 | 16 | 3 | 10.3 | 15 |
| | 7 | 2 | 9.1 | 16 | 2 | 10.1 | 16 | 2 | 10.4 | 17 |
| 2 | 3 | 0 | 8.2 | 14 | 0 | 8.4 | 13 | 0 | 8.2 | 14 |
| | 7 | 0 | 7.3 | 18 | 0 | 8.9 | 19 | 0 | 8.8 | 18 |
| 3 | 3 | 0 | 9.5 | 14 | 0 | 8.9 | 16 | 0 | 7.8 | 15 |
| | 7 | 0 | 7.1 | 17 | 0 | 9.2 | 17 | 0 | 9.3 | 17 |
| 4 | 3 | 2 | 9.3 | 13 | 2 | 9.8 | 14 | 2 | 9.5 | 14 |
| | 7 | 1 | 7.0 | 14 | 0 | 9.6 | 15 | 1 | 9.4 | 15 |
| 5 | 3 | 0 | 6.3 | 12 | 0 | 7.0 | 13 | 0 | 7.7 | 13 |
| | 7 | 2 | 8.2 | 14 | 1 | 9.4 | 15 | 2 | 9.2 | 15 |
| 6 | 3 | 0 | 6.1 | 10 | 0 | 6.4 | 12 | 0 | 7.9 | 13 |
| | 7 | 2 | 7.3 | 12 | 0 | 8.3 | 16 | 1 | 8.9 | 16 |
| 7 | 3 | 6 | 9.2 | 13 | 0 | 8.2 | 14 | 0 | 7.7 | 14 |
| | 7 | 1 | 6.7 | 13 | 0 | 3.3 | 15 | 0 | 3.2 | 15 |
| 8 | 3 | 0 | 5.7 | 13 | 0 | 6.4 | 13 | 0 | 5.6 | 14 |
| | 7 | 0 | 5.6 | 12 | 0 | 4.7 | 16 | 0 | 5.2 | 15 |
| 9 | 3 | 0 | 1.7 | 9 | 0 | 1.3 | 9 | 0 | 1.4 | 9 |
| | 7 | 0 | 4.9 | 11 | 0 | 5.1 | 15 | 0 | 5.3 | 15 |
| 10 | 3 | 7 | 10.0 | 15 | 0 | 7.2 | 12 | 7 | 10.0 | 15 |
| | 7 | 0 | 7.3 | 15 | 0 | 6.2 | 13 | 0 | 6.8 | 15 |




(a) Objective trajectory by elapsed time

(b) Objective trajectory by iteration

**Fig. 11.** MLCP objective trajectory, for three cut generation variants, as a function of elapsed time (Fig. 11(a)) and as a function of the current iteration (Fig. 11(b)).




(a) Capacity violations by elapsed time

(b) Capacity violations by iteration

**Fig. 12.** Number of shifts for which the capacity is violated (i.e., more than 1 maintenance team), for three cut generation variants, as a function of elapsed time (Fig. 12(a)) and as a function of the current iteration (Fig. 12(a)). The naive cut generation procedure took longer than 2 h since solving the MLCP in the last iteration (that started before the threshold of 2 h of running time) took very long; the process terminated as soon as this iteration was finished.

Fig. 11 demonstrates the MLCP objective value evolution trajectory. The trajectory starts at 879.285, which is the initial MLCP objective value before solving any violations, and corresponds to a solution of 879 nighttime maintenance activities. Clearly, the MLCP objective value is strictly increasing due to the iterative addition of constraints. The naive cut generation procedure took longer than 2 h since solving the MLCP in the last iteration (that started before the threshold of 2 h of running time) took very long; the process terminated as soon as this iteration was finished. An upper bound to the optimal objective value, has been determined by computing the MLCP solution when only nighttime maintenance is allowed: the objective value in this case is 1164.163. This corresponds to a number of 1163 nighttime maintenance activities. By running MSLCP, we aim to move some shifts to daytime to improve the performance of the MLs. It must be noted that none of the methods solve to optimality, meaning that there are capacity violations remaining in all three methods after applying the MSLCP procedure.

Within the time limit, naive, heuristic and min-cut performed 45, 70 and 170 iterations, respectively. Looking at the Objective Function (OF), the heuristic approach reaches the best solution (OF value) of 907.27, which corresponds to a number of 907 nighttime maintenance activities. The heuristic method converges quickest, even though less iterations were performed compared to the min-cut method. Hence, the cuts generated by the heuristic method are more efficient than those produced by the min-cut method. Even though this solution still contains some violations, this solution reflects a reduction of the number of maintenance activities during nighttime of more than 22 %.

Fig. 12 presents the number of capacity violations as a function of elapsed time and as a function of the current iteration for all cut generation procedures. First, it becomes clear that, for each cut generation procedure, the number of maintenance shifts for which the capacity is violated starts at 25 (the initial number of capacity violations) and is decreasing. However, the decrease is not monotonic. The added cuts as a result of the violation of capacity in one of the maintenance shifts, may induce a new MLCP solution that assigns maintenance in such a way that the capacity of maintenance shift which was formerly sufficient, now becomes violated. However, a significant difference in performance of the naive method on one side and heuristic and min-cut methods on the other is visible. The naive cut generation variant is the worst performing. After two hours of running time, it contains considerably more capacity violations than the other two cut generation variants, 17 compared to 3. More strikingly is the development of the number of violations in the min-cut cut generation variant compared to the heuristic cut generation variant. When looking at the development in terms of the elapsed time, the capacity violations in the min-cut cut generation variant initially decrease more sharply than in the heuristic cut generation variant, after which they in both remain constant for around 3 capacity violations. The min-cut cut generation variant found a solution with 3 maintenance shift violations or less after 9.6 m, whereas the heuristic cut generation procedure found such a solution after approximately 11.4 m. This is a hint that, when one aims to get a good sub-optimal solution as quick as possible, the min-cut cut generation variant may be slightly preferred. Note that, even though a limited amount of capacity violations still persist in the current method, its performance is significantly better than the MILP formulation of the problem (Section 4.3) and therefore better usable in practice.

To gain more understanding on this behavior, observe also the capacity violations as a function of the current iteration. At the beginning, the heuristic and min-cut cut generation variants show a similar path. This implies that, in each iteration, the resulting cuts in both variants lead to similar benefits in the reduction of capacity violations. However, the running time of the min-cut cut generation procedure per iteration is lower than the heuristic cut generation procedure, leading to a better performance in terms of computation time.

Finally, note that the heuristic and the min-cut method reach the same number of violations (approximately 3), but the heuristic method achieves a higher OF value. This is an indication that the heuristic method converges more quickly, and hence that, although both methods have attained a similar number of violations, the heuristic method may be nearer to its optimum than the min-cut method.

## 7. Conclusion

The current work proposes the Maintenance Scheduling and Location Choice Problem MSLCP, which provides a rolling stock maintenance schedule and a maintenance location choice, taking into account the available capacity of maintenance locations. The use of Logic-Based Benders' Decomposition is more efficient than solving a straightforward MILP model, and novel in the context of rail operation planning. Of the three alternative procedures for the generation of *cuts* (cuts are required for this approach), the heuristic and the min-cut methods significantly outperform a more naive method.

The subproblem (APP) provides the required number of maintenance teams within seconds for realistic problem sizes. This is beneficial since in the MSLCP context this needs to be done for every iteration and therefore contributes to the efficiency of the MSLCP model. In addition, it provides an optimal maintenance shift planning. As such, it is not only a valuable addition to the MLCP, but it can also be useful in operational contexts, where the planning of maintenance shifts is a manual and increasingly complex task that therefore leads to sub-optimal results. The proposed model may assist operational planners as it provides an optimal solution very quickly.

The current research investigates the performance of the MSLCP in two scenarios. The first scenario focuses on one particular maintenance shift for which the capacity is violated, demonstrating that the binary search heuristic cut generation procedure with 15 cuts is the most promising procedure to solve the capacity violations of a hard-to-solve instance. The second scenario focuses on solving the capacity issues in multiple maintenance shifts. The number of maintenance shifts for which the required capacity exceeded the available capacity could be reduced from 21 to 5 in less than 8 m using the min-cut cut generation procedure.

The current research has some limitations. First, it is assumed that maintenance jobs need to be performed sequentially and uninterruptedly. Although, in many practical cases, this is an acceptable or even standard way of working, this assumption does not allow for the opportunity that maintenance activities of different types are performed separately. Second, the min-cut cut procedure is defined for one maintenance team only. In the current situation at Dutch railways this is acceptable; since Dutch railways are making a shift to daytime maintenance, it is reasonable to expect that at most one maintenance team is available for daytime maintenance. However, for applications in which the available number of maintenance teams is higher than one, the current min-cut cut generation procedure cannot be used.

Several directions for future research can be recommended. First, the cut generation process within the MSLCP offers opportunities for improvement, and the lessons learned from its development can potentially be used in many other research areas related to scheduling of activities on locations and the capacities of these locations. Second, for broader applicability, the min-cut cut generation procedure used by the MSLCP shall be generalized to handle an arbitrary number of teams. This can potentially be achieved by generating additional instants in the RAPP, so that each team has its own dedicated instants. Third, there is a number of parameters that may impact the presented results, such as the planning horizon, the number of maintenance teams, the number of maintenance locations, and the duration and interval of maintenance activities. Future research may focus on the effect that these parameters have on model performance and solution quality. Fourth, an interesting next research topic is the improvement of the computational performance of the MLCP. This improvement is especially relevant in the light of the MSLCP algorithm, since it requires to run the MLCP in each iteration. Looking at the structure of the MLCP, it may potentially be decoupled into multiple smaller sub-problems that are easier to solve, e.g. by decoupling by rolling stock unit, creating sub-problems for each individual rolling stock unit, or by considering a rolling horizon, first optimizing a few days ahead and iteratively adding more days to the optimization.

## CRediT authorship contribution statement

**Jordi Zomer:** Writing – original draft, Methodology, Investigation, Conceptualization; **Nikola Bešinović:** Writing – review & editing, Supervision; **Mathijs M. de Weerdt:** Writing – review & editing, Supervision; **Rob M.P. Goverde:** Writing – review & editing, Supervision.

## Declaration of competing interest

None

## Appendix A. Notation

**Table A.1**
Overview of mathematical notation.

| Notaton | Meaning |
|---|---|
| **Sets and indices** | |
| $i \in I$ | Index for a rolling stock unit from the set of rolling stock units $I$ |
| $j \in J_i$ | Index for an MO from the set of MOs $J_i$ |
| $k \in K$ | Index for a maintenance activity type from the set of maintenance activity types $K$ |
| $l \in L$ | Index for potential maintenance location of the set of locations $L$ |
| $l_{ij} \in L$ | Index for the location corresponding to MO $j$ for rolling stock unit $i$, from the set of locations $L$ |
| $m \in M$ | Index for a moment from the set of moments $M$ |
| $n \in N$ | Index for a maintenance team from the set of maintenance teams $N$ |
| $q_{ij} \in Q$ | Index for a maintenance job for MO $j$ of rolling stock unit $i$ from the set $Q$ |
| $u \in U$ | Index to identify a shift |
| $\kappa \in \mathbb{N}$ | Counter to identify the current iteration of the MSLCP algorithm |
| **Parameters** | |
| $b_{ik} \in \mathbb{R}^+$ | Time since the last maintenance activity of type $k$ for rolling stock unit $i$ at midnight of the first day |
| $d_{ij} \in \{0,1\}$ | Binary input parameter used to indicate whether MO $i$ for rolling stock unit $j$ is during the day ($d_{ij} = 1$) or during the night ($d_{ij} = 0$) |
| $e_{ij} \in \mathbb{R}$ | End time of MO $j$ for rolling stock unit $i$ |
| $L^D_{\max} \in \mathbb{N}$ | Maximum number of daytime maintenance locations that may be opened |
| $\ell' \in \mathbb{R}^+$ | Restriction on the total computation time |
| $N_u \in \mathbb{N}$ | Number of maintenance teams available in shift $u$ |
| $o_k \in \mathbb{R}^+$ | Maximum interval between two subsequent maintenance activities of type $k$ |
| $r_{ij}, r_q \in \mathbb{R}^+$ | Release time of maintenance associated to MO $j$ for RS unit $i$ (MSLCP) or associated to job $q$ (APP) |
| $s_{ij} \in \mathbb{R}$ | Start time of MO $j$ for rolling stock unit $i$ |
| $T \in \mathbb{R}$ | Length of the planning horizon |
| $t_{ij}, t_q \in \mathbb{R}^+$ | Deadline time of maintenance associated to MO $j$ for RS unit $i$ (MSLCP) or associated to job $q$ (APP) |
| $v_k \in \mathbb{R}^+$ | Duration of maintenance activity of type $k$ |
| $v_q \in \mathbb{R}^+$ | Total duration of all maintenance activities in job $q$ |
| $y^N_l \in \{0,1\}$ | Binary input variable equal to 1 if location $l \in L$ is available for nighttime maintenance and 0 otherwise |
| $\delta^D \in (0,24)$ | Hour when the daytime maintenance window starts |
| $\delta^N \in (0,24)$ | Hour when the nighttime maintenance window starts |
| $\varepsilon, \varepsilon_2 \in \mathbb{R}^+$ | Technical parameters to denote a small penalty term |
| **Model variables** | |
| $APP(Q)$ | Objective value obtained after running the APP for set of jobs $Q$ |
| $h_{nmu} \in \mathbb{R}^+$ | Start time of the moment $m$ for team $n$ in shift $u$ |
| $w_{ij} \in \{0,1\}$ | Binary decision variable equal to 1 if MO $j$ for RS unit $i$ is used for any maintenance |
| $x_{ijk} \in \{0,1\}$ | Binary decision variable equal to 1 if maintenance of type $k$ is performed to rollingstock unit $i$ at MO $j \in J_i$, and 0 otherwise |
| $y^D_l \in \{0,1\}$ | Binary decision variable equal to 1 if location $l \in L$ is available for daytime maintenance, and 0 otherwise |
| $z_{nmuij} \in \{0,1\}$ | Binary decision variable equal to 1 if maintenance team $n$ on moment $m$ in shift $u$ performs the maintenance that was assigned to MO $j$ for RS unit $i$, and 0 otherwise |
| $\gamma_n \in \{0,1\}$ | Binary decision variable equal to 1 if team $n$ in shift $u$ is active and 0 otherwise |
| $\mu_{nmuij} \in \mathbb{R}^+$ | Decision variable that is equal to the duration of maintenance in MO $j$ of RS unit $i$ if that maintenance is assigned to team $n$ on moment $m$ in shift $u$; and 0 otherwise. |
| $\rho^\kappa$ | Solution of the MLCP after the $\kappa$th iteration of the MSLCP |

## References

Amorosi, L., Dell'Olmo, P., & Giacco, G. L. (2024). An integrated model for high-speed rolling-stock planning and maintenance scheduling. *Engineering Optimization*, *56*(6), 811–832.

Andrés, J., Cadarso, L., & Marín, A. (2015). Maintenance scheduling in rolling stock circulations in rapid transit networks. *Transportation Research Procedia*, *10*, 524–533.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252.

Canca, D., & Barrena, E. (2018). The integrated rolling stock circulation and depot location problem in railway rapid transit systems. *Transportation Research Part E: Logistics and Transportation Review*, *109*, 115–138.

Clarke, L., Johnson, E., Nemhauser, G., & Zhu, Z. (1997). The aircraft rotation problem. *Annals of Operations Research*, *69*, 33–46.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.

Dinmohammadi, F., Alkali, B., & Shafiee, M. (2016). A risk-based model for inspection and maintenance of railway rolling stock. In *European safety and reliability conference (ESREL)* (pp. 2016–2029).

Feo, T. A., & Bard, J. F. (1989). Flight scheduling and maintenance base planning. *Management Science*, *35*(12), 1415–1432.

Folco, P., Sahli, A., Belmokhtar-Berraf, S., & Bouillaut, L. (2024). A rolling horizon for rolling stock maintenance scheduling problem with cyclical activities. *Computers & Industrial Engineering*, *196*, 110460.

Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, *8*, 399–404.

Gopalan, R. (2014). The aircraft maintenance base location problem. *European Journal of Operational Research*, *236*(2), 634–642.

Gopalan, R., & Talluri, K. T. (1998). The aircraft maintenance routing problem. *Operations Research*, *46*(2), 260–271.

Hagberg, A., Swart, P., & Daniel, S. C. (2008). Exploring network structure, dynamics, and function using NetworkX. Technical Report Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Herr, N., Nicod, J.-M., Varnier, C., Zerhouni, N., & Malek Cherif, F. (2017). Joint optimization of train assignment and predictive maintenance scheduling. In *7th international conference on railway operations modelling and analysis (raillille 2017)* (pp. 699–708).

Hooker, J. (2011). Logic-based methods for optimization: combining optimization and constraint satisfaction (vol. 2). John Wiley & Sons.

Keita, K., Pellegrini, P., & Rodriguez, J. (2020). A three-step benders decomposition for the real-time railway traffic management problem. *Journal of Rail Transport Planning & Management*, *13*, 100170.

Kravchenko, S. A., & Werner, F. (2009). Minimizing the number of machines for scheduling jobs with equal processing times. *European Journal of Operational Research*, *199*(2), 595–600.

Lamorgese, L., & Mannino, C. (2015). An exact decomposition approach for the real-time train dispatching problem. *Operations Research*, *63*(1), 48–64.

Lamorgese, L., Mannino, C., & Natvig, E. (2017). An exact micro–macro approach to cyclic and non-cyclic train timetabling. *Omega*, *72*, 59–70.

Lamorgese, L., Mannino, C., & Piacentini, M. (2016). Optimal train dispatching by benders'-like reformulation. *Transportation Science*, *50*(3), 910–925.

Lin, B., Shen, Y., Wang, Z., Ni, S., & Zhao, Y. (2023). An iterative improvement approach for high-speed train maintenance scheduling. *Transportation Research Part B: Methodological*, *173*, 292–312.

Maróti, G., & Kroon, L. (2007). Maintenance routing for train units: The interchange model. *Computers and Operations Research*, *34*(4), 1121–1140.

Sarac, A., Batta, R., & Rump, C. M. (2006). A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, *175*(3), 1850–1869.

Taha, H. A. (2011). Operations Research: An Introduction. Pearson.

Tönissen, D. D., & Arts, J. J. (2018). Economies of scale in recoverable robust maintenance location routing for rolling stock. *Transportation Research Part B: Methodological*, *117*, 360–377.

Tönissen, D. D., Arts, J. J., & Shen, Z.-J. M. (2019). Maintenance location routing for rolling stock under line and fleet planning uncertainty. *Transportation Science*, *53*(5), 1252–1270.

Trepat Borecka, J., Leutwiler, F., & Corman, F. (2022). Studying complexity of decomposition in railway traffic planning. In *22nd swiss transport research conference. ascona, switzerland, 18–20 may*.

Van Hövell, M. E., Goverde, R. M. P., Bešinović, N., & de Weerdt, M. M. (2022). Increasing the effectiveness of the capacity usage at rolling stock service locations. *Journal of Rail Transport Planning & Management*, *21*, 100297.

Wagenaar, J. C., Kroon, L. G., & Schmidt, M. (2017). Maintenance appointments in railway rolling stock rescheduling. *Transportation Science*, *51*(4), 1138–1160.

Xu, X., & Dessouky, M. M. (2022). Train shunting with service scheduling in a high-speed railway depot. *Transportation Research Part C: Emerging Technologies*, *143*, 103819.

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

Zomer, J. (2020). Simultaneous optimization of rolling stock maintenance scheduling and rolling stock maintenance location choice. Master's thesis. Delft University of Technology.

Zomer, J., Bešinović, N., de Weerdt, M. M., & Goverde, R. M. P. (2020). Demo code for the maintenance scheduling and location choice problem model. GitHub repository. https://github.com/jzomergit/MSLCP/.

Zomer, J., Bešinović, N., de Weerdt, M. M., & Goverde, R. M. P. (2021). The maintenance location choice problem for railway rolling stock. *Journal of Rail Transport Planning & Management*, *20*, 100268.