A Multi-Swap Heuristic for Rolling Stock Rotation Planning with Predictive Maintenance

Felix Prause
Zuse Institute Berlin
Berlin, Germany
prause@zib.de

ABSTRACT

We present a heuristic solution approach for the rolling stock rotation problem with predictive maintenance (RSRP-PdM). The task of this problem is to assign a sequence of trips to each of the vehicles and to schedule their maintenance such that all trips can be operated. Here, the health states of the vehicles are considered to be random variables distributed by a family of probability distribution functions, and the maintenance services should be scheduled based on the failure probability of the vehicles. The proposed algorithm first generates a solution by solving an integer linear program and then heuristically improves this solution by applying a local search procedure. For this purpose, the trips assigned to the vehicles are split up and recombined, whereby additional deadhead trips can be inserted between the partial assignments. Subsequently, the maintenance is scheduled by solving a shortest path problem in a state-expanded version of a space-time graph restricted to the trips of the individual vehicles. The solution approach is tested and evaluated on a set of test instances based on real-world timetables.

KEYWORDS

Rolling stock rotation planning, Predictive maintenance, Heuristic, State-expanded graph model, Integer linear program

1 INTRODUCTION

Planning rolling stock rotations is essential for the operation of rail transportation and has been studied in the literature for quite some time. However, against the backdrop of climate change and the associated decarbonization of the transport sector, rail transportation represents a possible solution. It can therefore be assumed that the volume of freight and passengers transported by rail will continue to increase, which will also increase the complexity of the train scheduling. In addition, the availability of sensors and the analysis of the data they provide by the application of machine learning or traditional data mining methods enables a predictive scheduling of the vehicle maintenance. There is therefore a need to develop solution approaches for the automated dispatching of vehicles that are capable of integrating predictive maintenance strategies.

1.1 Related Work

The rolling stock rotation problem (RSRP) has already been investigated by a great variety of authors. For an introduction we refer to [14]. On the one hand, the contributions can be distinguished by the applied maintenance regime: Either preventive time- or distance-based maintenance regulation are employed, see for example [14], a predictive maintenance strategy is used, e.g., [4, 10, 15, 18], or no maintenance is considered at all. For an overview on the literature concerning RSRP we refer to [17]. On the other hand, the presented approaches can be categorized by the employed solution methods. The commonly utilized approaches are the direct application of integer linear programs (ILP), column generation, or the usage of heuristics, see [16]. In the following, we restrict ourselves to scenarios where maintenance is considered and focus on articles that apply heuristics to RSRP.

There exists a variety of heuristics that already have been applied to RSRP, but the most common ones are local search algorithms. Here, already determined solutions are modified to make them feasible or to improve their objective value. In [5], the RSRP is modeled by a sequence graph in which the trips correspond to nodes and the arcs indicate if two trips can be performed consecutively. They try to obtain a feasible solution by solving an ILP and employ a local search algorithm to include unassigned trips into the vehicle schedules if only a partial solution could be obtained. This is done by shifting trips between vehicles. This approach was further developed by [6], where an initial solution is derived from a stable set of trip nodes. Another local neighborhood search for the rescheduling variant of the RSRP was presented by [11]. Their approach is based on a space-time graph in which the nodes correspond to departure or arrival events of the trips, while the trips, waiting periods and deadhead trips are represented by the arcs. They apply a 2-opt heuristic to vehicles that meet at a station and interchange their subsequent trips.

Local search approaches are also applied to hypergraph formulations of the RSRP for maintenance scheduling. In [3], the authors state that the non-maintenance relaxation of their model is not that hard to solve. Therefore, a local neighborhood search is used to construct feasible solutions out of rotations that violate the maintenance constraints. This hypergraph model was subsequently used by other authors. In [1], a backtrack heuristic is given for the insertion of long-term maintenance services into predetermined rotations, while [8] present a heuristic relying on the observation that in real-life instances maintenance services can usually be performed during over-night stops. This yields a local neighborhood search for generating feasible solutions from maintenance-infeasible ones.

But also other types of heuristics have been applied to RSRP. In [17], the problem is formulated as a resource-constrained shortest path instance with side constraints in a space-time graph. The authors consider a scenario with short-term maintenance and solve the problem by applying a resource-constrained shortest path algorithm within a hill climbing heuristic. Finally, [4] present a variety of heuristics for RSRP with predictive maintenance. These include a genetic algorithm and three greedy algorithms that take the remaining useful life (RUL) of the considered vehicles into account.

1.2 Predictive Maintenance

The underlying idea of predictive maintenance in this article is the assumption that maintenance decisions should rely on the predicted health states of the considered vehicles, which cannot be measured directly. If we consider for example the doors of the vehicles, then their conditions must be approximated by either observing the number of occurring opening-closing cycles or by deriving them from sensor measurements like the voltage applied to the actuators or the vibration of the bearings. Since measurement errors occur here and further uncertainties arise when determining the health states from these values, e.g., by applying machine learning methods, the health states must be regarded as uncertain. In addition, the future load and operating conditions of the doors have to be predicted, which further increases the arising uncertainty. We therefore assume that the health states must be treated as random variables. The maintenance decisions are thus based on the probability that these random variables exceed some predefined threshold indicating a failure. We will denote this probability as the failure probability.

1.3 Contribution

In this article, we introduce the rolling stock rotation problem with predictive maintenance (RSRP-PdM), describe a graph model that approximates this problem, and present a local search heuristic to solve it. The proposed solution approach is able to handle predictive maintenance scenarios, where the health states of the vehicles are considered to be random variables and the maintenance is scheduled based on the failure probability of the vehicles. However, it can easily be adapted to handle distance- or time-based maintenance strategies. It also allows non-linear functions for modeling the degradation of the health states. Finally, the algorithm is tested and evaluated on a set of instances based on real-world timetables.

2 PROBLEM FORMULATION

In RSRP-PdM, we are given a set of vehicles $\mathcal{V}$, where each vehicle $v \in \mathcal{V}$ possesses a health state $H_v \in \mathbb{R}$. Here, $H_v = 1$ indicates that the vehicle is as good as new, while $H_v \leq 0$ corresponds to a breakdown of the vehicle. These health states are random variables to reflect their uncertainty and are distributed by a family of probability distribution functions Π with parameter space $\Theta \subseteq \mathbb{R}^n$, i.e., we have $H_v \sim \Pi_\theta \in \Pi$ for some $\theta \in \Theta$. H_v represents all possible health states of v with their respective probability of occurrence. Next, each $v \in \mathcal{V}$ has an initial state $H_v^0 \sim \Pi_{\theta_0}$, described by a $\theta_0 \in \Theta$. During the operation of trips and other services, e.g., deadhead trips, the conditions of the vehicles deteriorate, which is expressed by updating the parameters of their health states.

Furthermore, $\mathcal{L}$ is the set of all considered locations and $\mathcal{K}$ is a finite time horizon. Moreover, we are given a timetable $\mathcal{T}$ consisting of individual trips that need to be operated. To each trip $t \in \mathcal{T}$ we associate a departure location $l_t^d \in \mathcal{L}$ and a departure time $k_t^d \in \mathcal{K}$, as well as an arrival location $l_t^a \in \mathcal{L}$ and an arrival time $k_t^a \in \mathcal{K}$. Additionally, each trip possesses a degradation function $\Delta_t : \Theta \to \Theta$ altering the parameters of the health state of the vehicle operating t. We assume Δ_t to be continuous and monotonically increasing, i.e., $\nabla_{e_i} \Delta_t(\theta) \geq 0$ for all $i \in \{1, \dots, n\}$ and $\theta \in \Theta$, but we do not require it to be linear. Note that we associate similar degradation functions with the other activities of the vehicles, i.e., with waiting at stations, deadhead trips, and maintenance services. Finally, $n_t \in \mathbb{N}$ determines how many vehicles are required to operate t.

We associate costs with each of the possible operations, i.e., trips, waiting, deadhead trips, and maintenance services, and assume that breakdown costs arise when a vehicle failure occurs.

Next, we call a vehicle rotation balanced if the number of vehicles at each location $l \in \mathcal{L}$ is equal at the beginning and at the end of the considered time horizon. This balancedness is important as it gives rise to schedules that can be repeated on a weekly basis.

The task of RSRP-PdM is then to assign a sequence of trips, deadhead trips, and maintenance operations to each vehicle such that all given trips are operated and the resulting rotations are balanced. Here, the objective is to find a solution with minimum total costs, taking into account the operating costs, maintenance costs, and the expected costs of vehicle failures.

3 UTILIZED GRAPH MODELS

The proposed algorithm relies on two different graph models, that are presented in this section. The first is the *space-time graph*, which is widely used in the literature to model the RSRP. We present it briefly in the following and refer to [7] for a more detailed description. Afterwards, we introduce the *state-expanded event-graph*, which is a parameter-expanded version of the space-time graph and has been utilized in [13].

3.1 The Space-Time Graph

Given a RSRP-PdM instance as described in Section 2, the nodes of the space-time graph represent the departure and arrival events of the trips contained in $\mathcal{T}$. Each trip $t \in \mathcal{T}$ therefore induces two nodes $v_t^d = (l_t^d, k_t^d)$ and $v_t^a = (l_t^a, k_t^a)$, and corresponds to the arc $a_t = (v_t^d, v_t^a)$. By iterating over all trips and collecting the resulting nodes and arcs, we obtain the set of departure nodes V_+, the set of arrival nodes V_-, and the set of trip arcs $A_\mathcal{T}$. Afterwards, we add artificial start and end nodes for each location, i.e., $v_l^0 = (l, 0)$ and $v_l^\infty = (l, k_{\max})$ for each $l \in \mathcal{L}$, where $k_{\max} := \max\{\mathcal{K}\}$. These nodes form the sets of the start nodes V_0 and the end nodes V_∞. Thus, we define the node set to be $V := V_0 \cup V_+ \cup V_- \cup V_\infty$.

Next, we construct the arcs of the graph. First, we consider arcs representing that a vehicle waits at its current location. For this purpose, the nodes of each location $l \in \mathcal{L}$ are sorted in time-ascending order and an arc is added between each pair of time-consecutive nodes. This yields A_W. Finally, we construct the deadhead arcs A_D. Therefore, we iterate over all nodes $v_1 = (l_1, k_1) \in V_0 \cup V_-$ and add an arc to each $v_2 = (l_2, k_2) \in V_+ \cup V_\infty$ with $l_1 \neq l_2$, which has the smallest k_2 among the nodes at l_2 such that $k_1 + k(l_1, l_2) \leq k_2$, where $k(l_1, l_2)$ is the time required to travel from l_1 to l_2.

Combining these arc sets yields $A := A_\mathcal{T} \cup A_W \cup A_D$ and $G_{ST} = (V, A)$ is the resulting space-time graph. Note that we assign each arc the costs associated with its corresponding operation.

3.2 The State-Expanded Event-Graph

The space-time graph just described in Section 3.1 is well suited to determine assignments of trips to vehicles, however it is not trivial to incorporate maintenance constraints into it. Usually, these constraints are modeled by considering the vehicle rotations as resource-constrained paths, where a maintenance service is required when a certain resource threshold is exceeded and a resource consumption is associated with each of the arcs. The arising problem becomes even more complicated if we consider non-linear degradation functions, i.e., non-linear resource consumption. However, this should not be excluded, as mechanical components generally exhibit non-linear deterioration behavior. Therefore, we utilize the state-expanded event graph G_{SE}, which provides a linear approximation to this non-linear problem.

Given a discretization $\mathcal{D}$ of the parameter space Θ, i.e., a finite set $\mathcal{D} \subseteq \Theta$, we construct the nodes of G_{SE} by creating multiple copies of the nodes in G_{ST} for each $\theta \in \mathcal{D}$. This yields the node set $V' := \{(l, k, \theta) \mid (l, k) \in V(G_{ST}), \theta \in \mathcal{D}\}$ of G_{SE}.

Next, we generate the arcs. As in the construction of G_{ST}, each arc corresponds to a particular operation and the idea is that the arcs implicitly model the degradation, i.e., the resource consumption, of this operation. A vehicle traversing arc $a = (v_1, v_2)$ from $v_1 = (l_1, k_1, \theta_1)$ to $v_2 = (l_2, k_2, \theta_2)$ has a health state distributed by Π_{θ_1} before performing the task corresponding to a and a health state distributed by Π_{θ_2} afterwards. Here, the update of the health state parameters is given by the degradation function Δ_a of the arc, which is the degradation function of the associated operation.

But we do not necessarily have $\Delta_a(\theta_1) \in \mathcal{D}$, so there does not have to exist a head for a in V'. To resolve this problem, we define the following function that rounds to the nearest element of $\mathcal{D}$:

$$\lfloor \cdot \rceil_{\mathcal{D}} : \Theta \to \mathcal{D}, \quad \lfloor \theta \rceil_{\mathcal{D}} := \operatorname{argmin}_{\varphi \in \mathcal{D}} \left\{ \|\theta - \varphi\|_2 \right\}$$

Using this function, we construct the arcs of G_{SE} by iterating over its nodes and copying the outgoing arcs of their counterparts in G_{ST}. Let therefore $v_1 = (l_1, k_1, \theta_1)$ be a node in G_{SE}, and let $u_1 = (l_1, k_1)$ be the corresponding node in G_{ST}. Furthermore, consider any arc $a = (u_1, u_2) \in \delta^+(u_1)$, for some $u_2 = (l_2, k_2)$. Then, we determine $\theta_2 := \lfloor \Delta_a(\theta_1) \rceil_{\mathcal{D}}$ and add an arc from v_1 to $v_2 = (l_2, k_2, \theta_2)$. We repeat this procedure for all arcs originating from u_1 and subsequently for all nodes of G_{SE}. This results in the arc set A' of G_{SE} consisting of trip arcs, waiting arcs, and deadhead arcs. Note that this construction leads to multiple arcs corresponding to each of the trips.

Next, we introduce maintenance arcs. Their construction is similar to the construction of the deadhead arcs for the space-time graph and they are generated for every arrival node in G_{SE}. The difference in the construction is that instead of the time required to travel from location l_1 to l_2, the sum of the times necessary to travel from l_1 to the workshop, carry out the maintenance service there and then travel to l_2 is now considered. Moreover, as in the generation of the other arcs of G_{SE}, we apply $\lfloor \Delta_M(\cdot) \rceil_{\mathcal{D}}$ to the parameters of the tail node of each maintenance arc, where Δ_M is the degradation function associated with the maintenance activities. This resets the parameters to values that are as good as new.

Finally, the costs of the arcs in the state-expanded event-graph are equal to the costs of their corresponding arcs in the space-time graph, but we add the expected failure costs to the trip arcs. Therefore, we need to determine the failure probability of the vehicles during the operation of the trips. Consider any arc $a = (v_1, v_2) \in A'$ corresponding to some trip $t \in \mathcal{T}$. Let θ_1, θ_2 be the parameters of

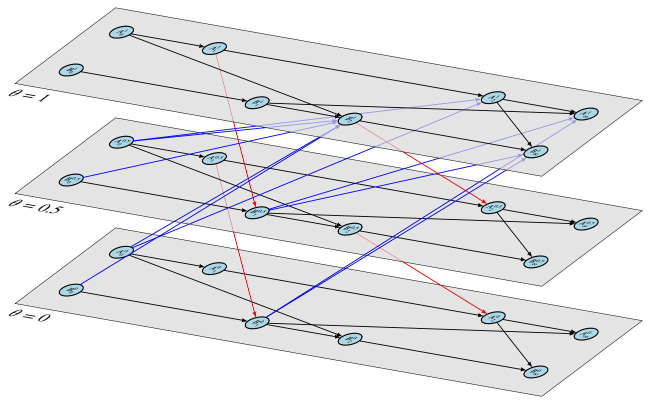

Figure 1: Example of a state-expanded event-graph adapted from [13].

v_1, v_2 respectively, then traversing a depicts that a vehicle v with health state $H_v \sim \Pi_{\theta_1}$ is operating t and its health state gets updated to $H_v \sim \Pi_{\theta_2}$ due to the occurring degradation. Thus, the failure probability of the vehicle is given by the probability that H_v exceeds the given failure threshold. Assuming that a breakdown occurs when H_v falls below zero, we need to determine $\mathbb{P}[H_v \leq 0] = \int_{-\infty}^{0} \Pi_{\theta_2}(x)\, dx$. The expected failure costs are then calculated by multiplying this value with the breakdown costs.

An example of a state-expanded event-graph is given in Figure 1, where the layers of nodes having the same parameter values are shaded in gray. The waiting and deadhead arcs are depicted in black, while the trip arcs are colored red. Traversing these arcs decreases the parameter by 0.5, while the maintenance arcs (blue) reset it to one. Note that a projection onto the space-time plane, i.e., ignoring the parameters, yields the underlying space-time graph.

4 ALGORITHM

Now, we present our solution approach for RSRP-PdM. This is based on the same observation as made in [3], where the authors state that the non-maintenance relaxation of RSRP is not that hard to solve. Thus, we first solve the RSRP ignoring the maintenance constraints and postpone the service scheduling to a subsequent step.

4.1 Generating Initial Solutions

To solve the non-maintenance relaxation of RSRP-PdM, we need to assign the trips to the vehicles in a cost-minimal way such that each trip is operated by the required number of vehicles and the vehicle balance of each location is even. Such an assignment corresponds to a flow in the space-time graph, which sufficiently covers the trip arcs and can be determined by solving ILP formulation (NMF).

In this formulation, $c_a \in \mathbb{R}_{\geq 0}$ are the costs associated with the arcs of the underlying space-time graph, and the objective function (1) aims at minimizing the total costs of all contained operations. Constraints (2) ensure the flow conservation and constraints (3) guarantee the balancedness of the resulting vehicle rotations. Constraints (4) depict the initial positioning of the vehicles in the beginning of the scenario, where $n_l \in \mathbb{N}_0$ is the number of vehicles located at $l \in \mathcal{L}$. Trip coverage is enforced by constraints (5), since the required number of vehicles is assigned to each trip. Finally, the domains of the variables are defined in (6).

$$\text{(NMF)} \quad \min \sum_{a \in A} c_a x_a \tag{1}$$

$$\text{s.t.} \sum_{a \in \delta^+(v)} x_a = \sum_{a \in \delta^-(v)} x_a \qquad \forall v \in V \setminus \{V_0 \cup V_\infty\} \tag{2}$$

$$\sum_{a \in \delta^+(v_l^0)} x_a = \sum_{a \in \delta^-(v_l^\infty)} x_a \qquad \forall l \in \mathcal{L} \tag{3}$$

$$\sum_{a \in \delta^+(v_l^0)} x_a \leq n_l \qquad \forall l \in \mathcal{L} \tag{4}$$

$$x_{a_t} = n_t \qquad \forall t \in \mathcal{T} \tag{5}$$

$$x_a \in \mathbb{N}_0 \qquad \forall a \in A \tag{6}$$

The solution of (NMF) represents a flow in the space-time graph given by a set of paths. Each path corresponds to the tasks assigned to one of the vehicles and consists of a sequence of trips connected by waiting or deadhead arcs. Since maintenance is not considered here, the path of each vehicle is determined by the assigned trips together with its origin and destination, as these have to be connected by the most cost-effective paths consisting of waiting and deadhead arcs.

Note that, the value of an optimal solution to (NMF) is a lower bound to the objective value of RSRP-PdM, since the costs can only increase if the deterioration of the vehicles, the associated expected failure costs and their maintenance are taken into account.

4.2 Scheduling Maintenance

After solving the non-maintenance relaxation of RSRP-PdM, we need to incorporate the maintenance services into the schedules of the individual vehicles. Therefore, we present a method that approximates the parameters of the health states of the vehicles and schedules the maintenance accordingly.

Recall that given a subset of trips $S \subseteq \mathcal{T}$ together with an origin $l_0 \in \mathcal{L}$ and a destination $l_\infty \in \mathcal{L}$, we can reconstruct the corresponding path in G_{ST} by sorting the trips in time-ascending order and connect the respective trip arcs by paths consisting of waiting and deadhead arcs. These paths will be termed *idle paths* in the following. We now transfer this idea to the state-expanded event-graph. Due to the construction of G_{SE}, based on a discretization $\mathcal{D}$, we may have multiple arcs corresponding to each of the trips. Consider a trip $t \in \mathcal{T}$ that starts at (l_t^d, k_t^d), then we added an outgoing arc corresponding to t to each node in $\{(l_t^d, k_t^d, \theta) \in V(G_{SE}) \mid \theta \in \mathcal{D}\}$. In the following, we will denote the set of arcs corresponding to a trip $t \in \mathcal{T}$ by $A(t)$. Thus, given a subset of trips $S \subseteq \mathcal{T}$ and $l_0, l_\infty \in \mathcal{L}$, we can determine a feasible schedule by selecting one arc from each $A(t)$, for all $t \in S$, and connecting them by idle paths. Note that the idle paths in G_{SE} can also contain maintenance arcs. In addition, the desired path must take into account the parameters of the initial health state of the assigned vehicle, i.e., $\theta_0 \in \Theta$, therefore it has to start at $v_0 = (l_0, 0, \lfloor \theta_0 \rceil_{\mathcal{D}}) \in V(G_{SE})$.

To find such a path, we first restrict G_{SE} to the arcs that are necessary for a vehicle that starts with parameters θ_0 at l_0, operates the trips in S and arrives at l_∞. We then determine a shortest path in this restricted graph. This graph will be referred to as $G_{SE}\big|_r$, for $r = (S, l_0, l_\infty, \theta_0)$, and can be obtained from G_{SE} as follows: First, we delete all trip arcs that belong to any $A(t)$, for $t \notin S$. Then, we remove all arcs that are not contained in an idle path that connects two of the remaining trip arcs. Next, all arcs that have a time overlap with one of the trip arcs are deleted. Finally, we add a sink node v_s to $G_{SE}\big|_r$ and add artificial arcs with costs equal to zero from all nodes in $\{v = (l_\infty, k_{\max}, \theta) \mid \theta \in \mathcal{D}\}$ to v_s.

Due to the construction of $G_{SE}\big|_r$, a shortest v_0-v_s-path thus corresponds to a minimum-cost schedule for $r = (S, l_0, l_\infty, \theta_0)$ w.r.t. the applied discretization $\mathcal{D}$. Note that the approximation quality depends on the granularity of $\mathcal{D}$. In the following, we assume that a solution x to the RSRP without maintenance is given by a set of schedules, where the schedule of each vehicle v_i can be represented as $s_i = (S_i, l_{0,i}, l_{\infty,i})$, for $i \in \{1, \ldots, |\mathcal{V}|\}$. Then, we can derive an approximate solution to RSRP-PdM from x by scheduling the maintenance of each vehicle, i.e., by determining a shortest path in each $G_{SE}\big|_{r_i}$, for $r_i = (s_i, \theta_{0,i})$. We will refer to this procedure by $\texttt{scheduleMaintenance}(x, G_{SE})$.

4.3 Improving Schedules by Swapping Trips

After presenting an approach to determine a solution for the non-maintenance relaxation of RSRP-PdM and a procedure for incorporating maintenance into the resulting vehicle schedules, we describe a local search algorithm that aims to improve a given solution by swapping parts of the vehicles' schedules.

Suppose we are given a non-maintenance solution x consisting of vehicle schedules $s_i = (S_i, l_{0,i}, l_{\infty,i})$, for $i \in \{1, \ldots, |\mathcal{V}|\}$. Then, we randomly select two vehicles $v_1, v_2 \in \mathcal{V}$ and consider their corresponding trip sets $S_1, S_2 \subseteq \mathcal{T}$. First, we sort S_1 and S_2 in ascending order of their departure times. Afterwards, we iterate over time-consecutive pairs $(t_{1,i}, t_{1,i+1})$ of S_1 and $(t_{2,j}, t_{2,j+1})$ of S_2 and check if it is possible to operate $t_{2,j+1}$ after $t_{1,i}$ and $t_{1,i+1}$ after $t_{2,j}$. If this is the case, (i, j) is a possible swap position. To find all swap positions, we next examine whether a swap is possible at the beginning or after the end of the schedules. Therefore, we check whether there are trips of S_1 that can be operated before the first trip of S_2 and if it is possible to reach them from $l_{0,2}$, i.e., the origin of v_2. The same is then repeated for S_2. Analogously, we check whether it is possible to swap trips after the last trip of S_1 or S_2.

These swap positions separate S_1 and S_2 into two ordered collections of equal cardinality. Their contained subsets of trips having the same indices can be exchanged without violating the feasibility of the resulting schedules. This procedure will be referred to as $\texttt{getSwappingParts}(S_1, S_2)$ and an example is illustrated in Figure 2. Here, the trips contained in blocks standing underneath each other can be exchanged arbitrarily and both resulting schedules would be feasible. For example, it would be possible to swap $\{t_9, t_{11}\}$ and $\{t_{10}, t_{12}\}$, or to shift t_8 to S_2.

Since a swap can occur before the first trip of a schedule, it is possible that the initial departure location of a schedule is changed.

S_1:	t_1	t_3, t_4, t_6	t_8	t_9, t_{11}		t_{15}, t_{17}
S_2:		t_2, t_5, t_7		t_{10}, t_{12}	t_{13}, t_{14}	t_{16}

Figure 2: Example of the exchangeable parts of two schedules.

It could therefore be advisable to assign this trip sequence to another vehicle at another origin, which can reach the new departure location with a less expensive deadhead trip. We thus define `matchVehiclePositions`(x), which determines a matching between the vehicles and the schedules of x. For this purpose, we calculate the minimum costs of the idle paths that connect the origins of the vehicles with the initial departure location of the trip sequences, provided they can be reached in time. Then, we assign the vehicles to the sequences according to the solution of the minimum-cost matching. This yields the updated origins $l_{0,i}$ of the schedules. To ensure the balancedness, we then solve another minimum-cost matching, which assigns the sequences to the destinations of the vehicles such that each location occurs as a destination exactly as often as it was employed as an origin. This results in the updated destinations $l_{\infty,i}$.

Thus, given a non-maintenance solution x, we define the procedure `multiSwap`(x) as follows: First, we randomly select two schedules S_1 and S_2 contained in x and apply `getSwappingParts`(S_1, S_2) to determine the subsets of trips that can potentially be swapped between them. We then decide at random for each of the corresponding parts which part is assigned to S_1 and S_2, respectively. This results in a modified solution y. Subsequently, we reassign the vehicle origins and destinations to the trip sequences by applying `matchVehiclePositions`(y) and obtain the solution z, which is the result of the multi-swap procedure.

Note that `multiSwap`(x) is a generalization of 2-opt, since it is obtained by selecting a certain swap position and interchanging all subsequent trip parts, while leaving the parts before unchanged.

4.4 The Resulting Algorithm

Combining the procedures presented throughout this section, yields the multi-swap heuristic for RSRP-PdM, see Algorithm 1. First, formulation (NMF) is solved to generate an initial solution. If this formulation is infeasible, there cannot be a solution to RSRP-PdM since the ILP solves the non-maintenance relaxation of the problem. Subsequently, `multiSwap`(x) is utilized to modify the current best solution and thus to explore the solution space, while `scheduleMaintenance`(x, G_{SE}) is employed to schedule the maintenance based on the approximated health states of the vehicles. The algorithm is stopped when the given time limit is reached.

5 COMPUTATIONAL RESULTS

In this section, we present the results of the proposed solution approach to RSRP-PdM and compare them to the LP-based lower bound given in [13]. This lower bound is based on a relaxation of an ILP in which the vehicle rotations are represented by paths in G_{SE} and collectively cover each trip n_t times. The algorithm was tested on the data set provided in [12], which originates from genuine timetables. The characteristics of the individual instances are listed in the first four columns of Table 1. All instances are based on a rail network with a length of 4,221 km and three maintenance facilities. The health states are assumed to be distributed by two-parameter normal distributions and the trips possess non-linear degradation functions. The considered components are the doors of the vehicles, which are not safety-relevant. Thus, the vehicles can continue to be operated even after a failure, but this causes additional costs.

Algorithm 1: Multi-swap heuristic for RSRP-PdM

```
   Data: RSRP-PdM instance $\mathcal{I}$, discretization $\mathcal{D}$
   Result: Solution to $\mathcal{I}$ or infeasible
1  $x \leftarrow$ solution to (NMF)
2  if $x$ is infeasible then
3      return infeasible
4  end
5  $G_{SE} \leftarrow$ state-expanded event-graph based on $\mathcal{D}$
6  $sol \leftarrow$ scheduleMaintenance($x, G_{SE}$)
7  repeat
8      $y \leftarrow$ multiSwap($x$)
9      $z \leftarrow$ scheduleMaintenance($y, G_{SE}$)
10     if $v(z) < v(sol)$ then
11         $x \leftarrow y$
12         $sol \leftarrow z$
13     end
14 until time limit is reached
15 return $sol$
```

It is assumed that the doors perform 1,500 opening-closing cycles before failing and undergo zero to four cycles at each stop of a trip.

5.1 Computational Setup

We performed all computations on a machine with Intel(R) Xeon(R) Gold 6342 @ 2.80GHz CPUs, eight cores and 64GB of RAM. The algorithm was implemented in Julia v1.9.1 [2] and Gurobi v10.0.2 [9] was used to solve (NMF) and the LPs for the lower bounds. All computations had a time limit of one hour.

5.2 Results

The results of the conducted computational experiments are given in the last five columns of Table 1. These contain the value of the best solution and the best lower bound for each instance, the resulting gap between these values, the gap after 180 seconds, and the time when the best solution was found. The obtained gaps show the effectiveness of the proposed algorithm, as they vary between 0 and 3.5% and are less than 4.3% after just 180 seconds.

The progression of the gap between the best heuristic result and the lower bound over time is quite similar across all instances and depicted in Figure 3. The majority and most significant improvements were achieved during the first 400 seconds. Afterwards, the solution value could still be enhanced, but the gained improvements decreased, and after 30 minutes almost no further progress could be recorded. An exception to this behavior is instance T5, where the best solution was found after just 14 seconds with a gap of 0.01%. These outcomes emphasize that the presented heuristic not only generates high-quality solutions, but is also capable of finding good results in a short time.

6 CONCLUSION

In this article we presented a heuristic solution approach to RSRP-PdM. We first defined the problem and then introduced two graph models: The first is used to model the non-maintenance relaxation of RSRP-PdM, while the other provides an approximation to the

Table 1: Characteristics and results for the test instances.

Instance	Trips	Destinations	Vehicles	Solution Value	Lower Bound	Gap in %	Gap after 180 s in %	Running Time in s
T1	566	8	6	269,728.67	261,432.23	3.08	3.08	3,517
T2	608	10	7	436,955.86	428,348.63	1.97	2.73	2,988
T3	636	15	16	1,427,088.22	1,380,028.25	3.30	4.28	3,597
T4	679	9	8	196,410.58	189,576.54	3.48	3.48	3,387
T5	813	16	14	327,804.96	327,770.13	0.01	0.01	14
T6	919	17	29	2,355,022.71	2,290,595.54	2.74	3.45	3,545

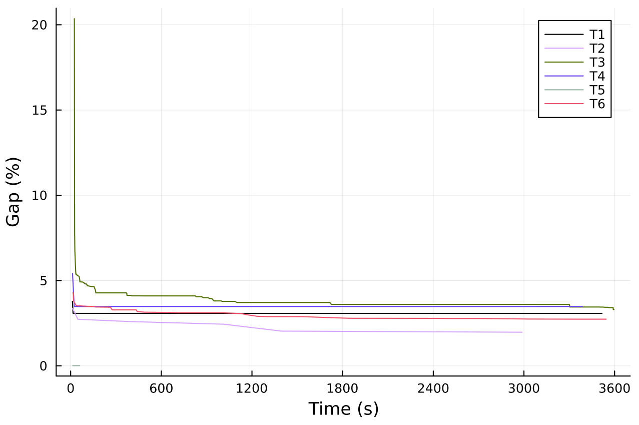

Figure 3: Gaps of all instances over time.

problem itself. Then, we discussed the individual steps of the proposed algorithm. These consist of an ILP to find an initial solution to the non-maintenance relaxation, a method for approximate maintenance scheduling and finally a local search procedure based on the multiple random swapping of trips. The effectiveness of the proposed algorithm is then demonstrated by conducting computational experiments on a set of test instances.

Possible next steps for future research are a combination of the presented approach with simulated annealing as well as the investigation of other meta-heuristics. Furthermore, it would be interesting to examine whether it is more effective to employ these heuristics on the non-maintenance relaxation and postpone the maintenance scheduling to a later step, or whether it is advantageous to apply these approaches directly to the state-expanded event-graph.

ACKNOWLEDGMENTS

The author would like to thank Boris Grimm for his comments and interesting discussions related to this work. This research was supported by the Pro FIT innovation funding program (grant no. 10174564) of the State of Berlin and co-financed by the European Union.

REFERENCES

[1] Timo Berthold, Boris Grimm, Markus Reuther, Stanley Schade, and Thomas Schlechte. 2019. Strategic planning of rolling stock rotations for public tenders. In *RailNorrköping 2019. 8th International Conference On Railway Operations Modelling And Analysis, Norrköping, Sweden, June 17th–20th, 2019.* Linköping University Electronic Press, 128–139.

[2] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A fresh approach to numerical computing. https://julialang.org. *SIAM review* 59, 1 (2017), 65–98. https://doi.org/10.1137/141000671

[3] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, Kerstin Waas, and Steffen Weider. 2016. Integrated optimization of rolling stock rotations for intercity railways. *Transportation Science* 50, 3 (2016), 863–877. https://doi.org/10.1287/trsc.2015.0633

[4] Omar Bougacha, Christophe Varnier, and Noureddine Zerhouni. 2022. Impact of decision horizon on post-prognostics maintenance and missions scheduling: a railways case study. *International Journal of Rail Transportation* 10, 4 (2022), 516–546. https://doi.org/10.1080/23248378.2021.1940329

[5] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. 2012. A fast heuristic algorithm for the train unit assignment problem. In *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems.* Schloss Dagstuhl-Leibniz-Zentrum für Informatik. https://doi.org/10.4230/OASIcs.ATMOS.2012.1

[6] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. 2019. An effective peak period heuristic for railway rolling stock planning. *Transportation Science* 53, 3 (2019), 746–762. https://doi.org/10.1287/trsc.2018.0858

[7] Jean-François Cordeau, François Soumis, and Jacques Desrosiers. 2001. Simultaneous assignment of locomotives and cars to passenger trains. *Operations research* 49, 4 (2001), 531–548. https://doi.org/10.1287/opre.49.4.531.11226

[8] Boris Grimm, Ralf Borndörfer, Markus Reuther, and Thomas Schlechte. 2019. A Cut Separation Approach for the Rolling Stock Rotation Problem with Vehicle Maintenance. In *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019).* Schloss Dagstuhl-Leibniz-Zentrum für Informatik. https://doi.org/10.4230/OASIcs.ATMOS.2019.1

[9] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual, Version 10.0.2. http://www.gurobi.com.

[10] Nathalie Herr, Jean-Marc Nicod, Christophe Varnier, Noureddine Zerhouni, and Pierre Dersin. 2017. Predictive maintenance of moving systems. In *2017 Prognostics and System Health Management Conference (PHM-Harbin).* IEEE, 1–6. https://doi.org/10.1109/PHM.2017.8079111

[11] Rowan Hoogervorst, Twan Dollevoet, Gábor Maróti, and Dennis Huisman. 2021. A Variable Neighborhood Search heuristic for rolling stock rescheduling. *EURO Journal on transportation and logistics* 10 (2021), 100032. https://doi.org/10.1016/j.ejtl.2021.100032

[12] Felix Prause and Ralf Borndörfer. 2023. *Construction of a Test Library for the Rolling Stock Rotation Problem with Predictive Maintenance.* Technical Report 23-20. ZIB, Takustr. 7, 14195 Berlin.

[13] Felix Prause, Ralf Borndörfer, Boris Grimm, and Alexander Tesch. 2023. *Approximating the RSRP with Predictive Maintenance.* Technical Report 23-04. ZIB, Takustr. 7, 14195 Berlin.

[14] Markus Reuther. 2017. *Mathematical optimization of rolling stock rotations.* Ph.D. Dissertation. Technische Universität Berlin (Germany). https://doi.org/10.14279/depositonce-5865

[15] Pegah Rokhforoz and Olga Fink. 2021. Hierarchical multi-agent predictive maintenance scheduling for trains using price-based approach. *Computers & Industrial Engineering* 159 (2021), 107475. https://doi.org/10.1016/j.cie.2021.107475

[16] Thomas Schlechte, Christian Blome, Stefan Gerber, Stefan Hauser, Jens Kasten, Gilbert Müller, Christof Schulz, Michel Thüring, and Steffen Weider. 2023. *The Bouquet of Features in Rolling Stock Rotation Planning.* Technical Report. EasyChair.

[17] Per Thorlacius, Jesper Larsen, and Marco Laumanns. 2015. An integrated rolling stock planning model for the Copenhagen suburban passenger railway. *Journal of Rail Transport Planning & Management* 5, 4 (2015), 240–262. https://doi.org/10.1016/j.jrtpm.2015.11.001

[18] Meng-Ju Wu and Yung-Cheng Lai. 2019. Train-set Assignment Optimization with Predictive Maintenance. In *RailNorrköping 2019. 8th International Conference on Railway Operations Modelling and Analysis (ICROMA), Norrköping, Sweden, June 17th–20th, 2019.* Linköping University Electronic Press, 1131–1139.