Strategic Planning of Rolling Stock Rotations for Public Tenders

Timo Berthold [a], Boris Grimm [b], Markus Reuther [c],
Stanley Schade [b,1], Thomas Schlechte [c]
[a] Fair Isaac Germany GmbH c/o Zuse Institute Berlin
Takustr. 7, 14195 Berlin, Germany
[b] Optimization, Zuse Institute Berlin
Takustr. 7, 14195 Berlin, Germany
[1] E-mail: schade@zib.de, Phone: +49 (0) 30 84185 336
[c] LBW Optimization GmbH
Obwaldener Zeile 19, 12205 Berlin, Germany

Abstract
Since railway companies have to apply for long-term public contracts to operate railway lines in public tenders, the question how they can estimate the operating cost for long-term periods adequately arises naturally. We consider a rolling stock rotation problem for a time period of ten years, which is based on a real world instance provided by an industry partner. We use a two stage approach for the cost estimation of the required rolling stock. In the first stage, we determine a weekly rotation plan. In the second stage, we roll out this weekly rotation plan for a longer time period and incorporate scheduled maintenance treatments. We present a heuristic approach and a mixed integer programming model to implement the process of the second stage. Finally, we discuss computational results for a real world tendering scenario.

Keywords
Rolling Stock Rotation Planning, Strategic Planning, Longterm, Maintenance

1 Introduction

Due to structural changes in the railway system all over Europe, the operation of trains does not lie solely in the hands of single state owned railway companies like it used to be. Instead railway companies compete in public tenders to receive public contracts to operate for example a certain railway line or the public railway network of a city, see Schlechte (2012) and Abbink et al. (2018). This market forces the companies to act as cost-efficient as possible to outperform others in the competition for public contracts. Since the planning of railway operations by a single company is already a complex problem, planning in a segregated market with several stakeholders becomes even more elaborate.

One of the arising problems is the estimation of costs to operate a railway enterprise for long-term periods. If a private company participates in a public tender for the operation of a railway line, it is critical to make a cost-effective offer, and, thus, to estimate the cost for all operational aspects. In this paper we analyze the costs for the rolling stock rotations including short-term, mid-term, and long-term maintenance schedules. A special feature of

this analysis is that we cannot solely rely on known optimization techniques or the planning of rolling stock rotations, because they are not suitable for periods of several years. To the knowledge of the authors the rolling stock rotation problem has not been addressed in the literature on such a long-term scale.

The objective of this paper is to obtain a vehicle and maintenance schedule for a given fleet on a given timetable that is to be operated and maintained over the course of ten years. The interesting aspect in this exposition is the presence of two scales. While the vehicles travel only several thousand kilometers per week, the maintenance procedures have to be performed after several ten thousand kilometers. Hence, they are not performed every week, and cannot be included into a cyclical weekly rotation plan. Over the course of several years it is still necessary to determine when and how often these maintenances have to be performed and it may also be of interest whether they can be distributed evenly in order to avoid peak workloads at the maintenance facilities.

In the following section we give a detailed description of the problem and our solution approach which is divided into two stages. We present the algorithmic details to solve the second stage separately in Section 3. Finally, we discuss the computational results that we obtained and summarise our findings.

2 Methodology

2.1 Problem and Data

As a problem, we use an anonymized real-world instance provided by the TransDev GmbH. The task is to plan the rolling stock rotations on a regional railway line between three cities GB, WB and HAM that are $32km$ and $42km$ apart from each other. The data specify a weekly timetable. There are 41 trips in each direction, which can be valid for different days of the week, with varying required passenger capacities. Furthermore, fuelling and maintenance intervals are provided and there are two kinds of vehicles available. One kind has 400 seats available, the other kind has 200 seats. All vehicles can be coupled to increase the capacity or to reduce the number of deadhead trips. Except for Saturday and Sunday there are usually no trips between 1am and 5am in the morning. Since only the timetable was provided, the cost for the vehicles, trips and so on had to be estimated. It was also given that refuelling has to occur every $1{,}000km$ and an IS maintenance has to occur every $40{,}000km$. The duration of the refuelling was assumed as $15min$. The duration of the IS maintenance depends on the level. It was estimated to take at least 10 hours for level 1. At multiples of $40{,}000km$ a higher level IS maintenance is required that may take more time. The highest level is 5, necessary after $640{,}000km$, which was estimated to take full 24 hours.

2.2 Approach

Due to the dual scale of the problem, we decompose it into two stages and use a sequential optimization approach. We will calculate a weekly rotation plan in the first stage and take care of the long-term maintenances in the second stage, where we track every single vehicle with the passage of time. If necessary or beneficial, we will break the rotation plan of the first stage apart and reassemble it to suit our needs.

First Stage
We use the rolling stock rotation optimizer ROTOR (Borndörfer et al. (2016); Reuther (2017)) to calculate a one week rotation plan for the first stage. For this part the long-term maintenance intervals are left out. ROTOR transforms the input timetable into a graph where trips are vertices that have to be connected by arcs that represent turns and empty trips, i.e. ROTOR models the problem using a mixed integer linear programming approach. To reduce the problem size the connecting arcs are created dynamically in the solution process. The pricing step that determines which arcs will be included into the problem is guided by a coarser version of the problem that simplifies the problem to an assignment problem. Furthermore, ROTOR uses heuristics like rapid branching to find primal solutions faster.

Table 1: Solution info

cycles	3
trips/turns	764
vehicles	4
total trip distance	27,974.31km
deadhead trip distance	242.83km
deadhead trips	6
solution time	47min
gap	0.00%

For the given timetable, ROTOR achieved an optimal solution using 4 of the vehicles with a capacity of 400 passengers each although vehicles with 200 passengers were available as well, see Table 1 for some statistics. The higher vehicle capacity was accounted for by a cost factor of 1.5. It was possible to include a 24 hour idle time for one vehicle on Sunday, since the trip density on that day was lower. This time is sufficient to include any of the required maintenances. The second longest idle time was less than 8 hours and, therefore, not even sufficient for an IS level 1 maintenance. ROTOR could be set up to include the refuelling in its calculations. However, it is easier to verify that there are sufficiently many time windows of $15min$ or longer in a postprocessing step, which we have done.

Since we want to roll out the rotation plan in the next phase, it is of importance that ROTOR provides a cyclical rotation plan. Apart from that it is conceivable to use other methods for the first stage, e.g. the method proposed by Frisch et al. (2018).

Second Stage
In the second stage, we intend to roll out the weekly rotation plan on a long-term planning interval of ten years and consider different strategies to incorporate the required maintenance treatments. A schematic drawing of the rotation plan is given in Figure 1 and the respective distances in Table 2. It can be seen that the cyclical rotation plan consists of 3 cycles. The opportunity for a maintenance lies at A_3 in the first cycle. Hence, only a single vehicle could undergo the required IS treatments. In order to enable the maintenance of the other vehicles, the rotation plan has to be modified. Our approach to connect every vehicle to a maintenance opportunity is to possibly swap vehicles if they are located at the same station overnight. It turns out that from Monday to Saturday either the vehicles on A and C or the vehicles on B and D are both located in GB. We ruled out Saturday morning, since

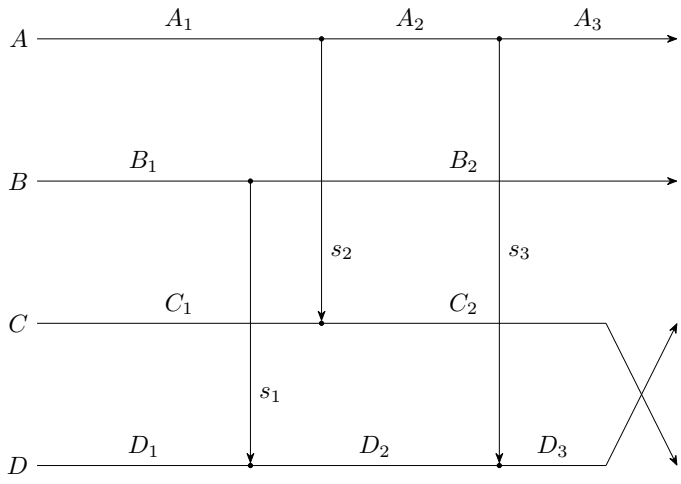

Figure 1: Schematic representation of the rotation plan computed in the first stage. The four rows labeled from A to D represent possible sets of schedules during a week for a vehicle. A vehicle that performs A or B will perform the same schedule in the next week. A vehicles that performs C will perform D in the following week and vice versa. The capital letters denote the various schedules/distances to be covered by the vehicles. The indexed capital letters denote subschedules and subdistances. Since C and D form a cycle together, the rotation plan consists of the 3 cycles in total. Since the vehicle on schedule B and D are both located in GB on Wednesday night, they can be exchanged at this point. The same applies to A and C on Thursday night and A and D on Sunday morning. These possibilities for switching the vehicles are denoted by s_1, s_2 and s_3.

the operations from Friday night to Saturday are more or less continuous. That would leave two options to swap between A and C or B and D. In order to keep the approach as simple as possible (also with regard to the practical implementation), we decided to take the latest possible options on Thursday and Friday only. These are denoted as s_1 and s_2 in Figure 1. Furthermore, at the beginning of the 24 hour idle time in GB, the vehicles on A and D are both located in GB. Therefore, either of them could idle or be maintained, while the other does the schedule D_3. This opportunity for a swap is denoted as s_3. Since railway planners are often interested in single cycles in one rotation, it is worth mentioning that by switching vehicles at s_1 and s_3 every time that these switches are possible, one can transform the three cycles into one single cycle.

What is now left is to determine an actual rotation plan that spans several weeks. This is the topic of the next section.

Table 2: Distances (in km) of vehicle schedule

A_1	3,830.400	B_1	3,222.030	C_1	4,296.040	D_1	2,947.070
A_2	2,316.580	B_2	4,261.810	C_2	3,304.210	D_2	3,147.760
A_3	0	B	7483.840	C	7,600.250	D_3	891.240
A	6,146.980					D	6,986.070

3 Solution Approaches

3.1 Backtracking Heuristic

Since the rotation plan from the first stage does not assure that every vehicle can be maintained, we have to modify it on a week by week basis. We have the option s_1 to intersect the paths B and D, s_2 to intersect A and C, and s_3 to intersect A and D. If we want to maintain the vehicle that starts the week on path B, we can apply the intersections s_1 and s_3 simultaneously. To determine which option should be applied in which week, such that every vehicle is maintained in time, we suggest a backtracking algorithm. For a given number of weeks, it performs a depth first search. The algorithm decides on one of the following options for a given week:

- run the rotation plan as determined by the first stage,
- apply the intersections s_1 and s_3,
- apply the intersection s_2,
- apply the intersection s_3 without s_1.

Furthermore, it decides if a maintenance shall be performed during said week. The general idea of backtracking is to have the algorithm select one option for a week and, then, to advance to the next week. This process is iterated until the desired number of weeks is reached or the maintenance interval of a vehicle is violated. In the first case, the algorithm has successfully determined a feasible solution and terminates. In the second case, it moves one week back and determines whether there is another choice out of the above options that was not already tested and test it, if so. If no options are left, it moves one more week back and so forth. If the options of the first week have run out, the algorithm has enumerated all possibilities and it will detect that there is no feasible solution. In the case of success, it will return the first solution that comes along.

We have two ways to control the heuristic. The first way is the range of decisions that we allow for every week. The options given above were selected, such that every vehicle could be maintained at the end of the week, regardless whether it started the week on A, B, C or D. The second way of control is the order that the algorithm uses to test the various options. We have used the order as stated above. Each of the options is tested without performing maintenance first. If that does not work, the algorithm tests the option with performing maintenance. If that does not work either, it moves on to the next option. If it was the other way, the algorithm would recommend to perform a maintenance every week, since it does not take the costs for a maintenance into account.

Table 3: Results of the backtracking heuristic. A periodic pattern that is repeated infinitely is given by weeks 5 to 9. k – number of week. v_i – position of vehicle i at the beginning of the week. block – km travelled since last maintenance. current – km travelled in the current week. total – total km travelled. m – perform maintenance at the end of the week?

k	v_1	block	current	total	v_2	block	current	total	swaps	m
0	A	0	0	0	B	0	0	0		n
1	A	6147	6147	6147	B	7484	7484	7484		y
2	A	0	6147	12294	B	14968	7484	14968	s_1, s_3	y
3	C	7038	7038	19332	A	0	6370	21337	s_1, s_3	y
4	D	14638	7600	26932	C	7038	7038	28376	s_1, s_3	y
5	B	21847	7209	34141	D	14638	7600	35976		n
6	B	29331	7483	41625	C	21625	6986	42962	s_1, s_3	y
7	A	0	6370	47995	D	29225	7600	50562	s_3	y
8	C	7038	7038	55033	A	0	6095	56657	s_1, s_3	y
9	D	14638	7600	62633	C	7038	7038	63695	s_1, s_3	y

k	v_3	block	current	total	v_4	block	current	total	swaps	m
0	C	0	0	0	D	0	0	0		n
1	D	7600	7600	7600	C	6986	6986	6986		y
2	C	14586	6986	14586	D	14586	7600	14586	s_1, s_3	y
3	D	22187	7600	22187	B	21795	7209	21795	s_1, s_3	y
4	B	29395	7209	29395	A	0	6370	28165	s_1, s_3	y
5	A	0	6370	35765	C	7038	7038	35203		n
6	A	6147	6147	41912	D	14638	7600	42803	s_1, s_3	y
7	C	13185	7038	48950	B	21847	7209	50012	s_3	y
8	D	20785	7600	56551	B	29331	7484	57496	s_1, s_3	y
9	B	27994	7209	63760	A	0	6370	63866	s_1, s_3	y

The heuristic was implemented in `Python` and it only takes a few seconds to run up to week 520. However, it is actually sufficient to run it for 14 weeks to see that the weeks 5 to 9 give a periodic pattern that is repeated subsequently. The results of the first 9 weeks are presented in Table 3. The only deviation from this pattern may happen in the last weeks where a maintenance may be left out. Since only one maintenance per week is possible, the algorithm has to plan ahead, so that only one vehicle reaches its maintenance interval and has to be maintained at a time. At the end of the time horizon, a maintenance may be left out, since the algorithm does not look ahead anymore. Thus, several vehicles can be close to their kilometer limit and require maintenance in the week after the end of the time horizon.

3.2 MILP Model

In addition to the backtracking heuristic, we have come up with a Mixed Integer Linear Programming (MILP) model. The model is basically a multi-commodity flow with one commodity per vehicle. The coupling of the commodities happens via the integral s vari-

ables that allow the transition of a flow between the four possible paths as shown in Figure 1. If an s is set to 1 for one commodity, it means that the flow of the commodity transitions downwards (A being on top and D being on the bottom). In that case, the corresponding s of another commodity has to be set to -1, which means that the flow of the corresponding commodity flows upwards.

Parameters

- N – number of weeks considered
- $A = A_1 + A_2 + A_3$, $B = B_1 + B_2$, $C = C_1 + C_2$, $D = D_1 + D_2 + D_3$ – distances of the different vehicle schedules, cp. Figure 1
- U – maximum maintenance interval in kilometers

Variables

We assume $i, j \in \{1, 2, 3, 4\}$ and $k \in \{1, \ldots, N\}$ if not specified otherwise. Also, if not specified otherwise, the variables are non-negative and continuous.

- $m_k \in \{0, 1\}$ – perform a maintenance in week k
- $s_{i,l,k} \in \{-1, 0, 1\}$ vehicle i transitions at s_l ($l \in \{1, 2, 3\}$) in week k, cp. Figure 1
- $y_{i,j,k} \in [0, 1]$ – vehicle flow of vehicle i on path j ($j = 1$ corresponds to A, $j = 2$ to B and so on) in week $k \in \{1, \ldots, N + 1\}$
- $x_{i,k} \in \mathbb{R}_+$ – kilometers travelled since last maintenance by vehicle i after week $k \in \{1, \ldots, N\}$ or before week 1 ($k = 0$).
- $t_{i,k} \in \mathbb{R}_+$ – transition variable to check that U is not exceeded
- $w_{i,k} \in \mathbb{R}_+$ – auxiliary variable to reset x if a maintenance is performed

Model

$$\min \sum_{k=1}^{n} m_k \tag{1}$$

s.t.

$$y_{i,j,1} = \delta_{i,j} \qquad \text{for } i,j \in \{1,2,3,4\} \tag{2}$$

$$y_{i,1,k+1} = y_{i,1,k} - s_{i,2,k} - s_{i,3,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{3}$$

$$y_{i,2,k+1} = y_{i,2,k} - s_{i,1,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{4}$$

$$y_{i,3,k+1} = y_{i,4,k} + s_{i,1,k} + s_{i,3,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{5}$$

$$y_{i,4,k+1} = y_{i,3,k} + s_{i,2,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{6}$$

$$\sum_{i=1}^{4} s_{i,l,k} = 0 \qquad \text{for } l \in \{1,2,3\},\ k \in \{1,\dots,N\} \tag{7}$$

$$y_{i,2,k} - 1 \leq \quad s_{i,1,k} \leq y_{i,2,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{8}$$

$$y_{i,4,k} - 1 \leq -s_{i,1,k} \leq y_{i,4,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{9}$$

$$y_{i,1,k} - 1 \leq \quad s_{i,2,k} \leq y_{i,1,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{10}$$

$$y_{i,3,k} - 1 \leq -s_{i,2,k} \leq y_{i,3,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{11}$$

$$y_{i,1,k} - s_{2,k} - 1 \leq \quad s_{i,3,k} \leq y_{i,1,k} - s_{2,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{12}$$

$$y_{i,4,k} + s_{1,k} - 1 \leq -s_{i,3,k} \leq y_{i,4,k} + s_{1,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{13}$$

$$\begin{aligned} t_{i,k} = {} & x_{i-1,k} + Ay_{i,1,k} + By_{i,2,k} \\ & + Cy_{i,3,k} + Dy_{i,4,k} \\ & + (D_2 + D3 - B2)s_{i,1,k} \\ & + (C_2 - A_2 - A_3)s_{i,2,k} \\ & + (D_3 - A_3)s_{i,3,k} \end{aligned} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{14}$$

$$x_{i,k+1} = t_{i,k} - w_{i,k} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{15}$$

$$w_{i,k} \leq Um_k \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{16}$$

$$w_{i,k} \leq Uy_{i,0,k+1} \qquad \text{for } i \in \{1,2,3,4\},\ k \in \{1,\dots,N\} \tag{17}$$

Description

The objective of the model is to perform as few maintenances as possible. The constraint (2) sets the starting positions of the vehicles. Here, $\delta_{i,j} = 1$ if and only if $i = j$, otherwise $\delta_{i,j} = 0$. The constraints (3-6) are flow conservation constraints and (7) couples the different commodities. With a pure flow model, it would be possible to have a flow back in time, e.g. a flow on D_1 and D_2 that reverses on s_3 and A_2 and then uses s_2. In order to forbid these kind of flows and keep causality in the model, we use the constraints (8-13). Moreover, (14) takes account of the travelled kilometers since the last maintenance. For example, if vehicle 2 is scheduled for A in week 3, then $y_{2,1,3}$ is 1. If the vehicle is supposed to change on schedule C during that week, then $s_{2,2,3} = 1$. Thus, the remainder of the first schedule $A_2 + A_3$ will be removed from the milage, while the second part C_2 of the third schedule will be added, cf. Figure 1. The constraints (15-17) ensure that the kilometers are reset if a maintenance is performed.

The model and the underlying problem could also be interpreted as a special case of the Train Dispatching Problem presented in Boccia et al. (2013) and Mannino (2011) or the Train Timetable Rescheduling Problem presented in Cacchiani et al. (2014). This leads to slightly different model formulations, which are comparable in the sense of tractable problem sizes.

The model was implemented and solved using the Python interface of the FICO Xpress solver version 8.5.7. The solutions of the backtracking heuristic were used as initial solutions. The tests were performed on a Dell Precision Tower 3620 with 30GB of main memory and 8 Intel® Xeon(R) CPU E3-1245 v5 @ 3.50GHz. For the instances with a time horizon of 15 or fewer weeks, it turned out that the solution provided by the backtracking heuristic was optimal, but it could take long to obtain the proof, see Table 4. For a 20 week instance, the memory did not suffice.

Table 4: Solution info

weeks	rows	cols	nodes	optimal value	time (s)
10	850	430	411,303	7	88
15	1265	511	47,045,565	11	17,867

3.3 Lower Bounds

Since proving the optimality of the heuristic solution using a general purpose MILP solver was not always possible or took long, one might try to find lower bounds oneself to directly prove the optimality of a backtracking solution or to provide these bounds to the solver. An easy bound is obtained as follows. The maintenance interval is $40{,}000km$ and the vehicles travel $27{,}974.31km$ per week (see Table 1). Hence, assuming a cyclical solution, the vehicles have to be maintained at least ceil$(N \cdot 27{,}974.31/40{,}000)$ times in an N week scenario, e.g. for a cyclical 5 week solution at least 4 maintenances are required. If the solution is not cyclic, we have to account for the fact that the vehicles could start with $0km$ travelled since the last maintenance but finish the scenario with almost $40{,}000km$ since the last maintenance. Hence, the lower bound is decreased by 4.

These bounds are not tight enough to prove the optimality of our solution for the 10 year scenario. One way to overcome this may be to use a cyclical solution. Borndörfer et al. (2008) and Borndörfer et al. (2018) argue why it is reasonable to do that for long periods. However, there are still ways to improve our bounds further. For example, we can determine the distance to the maintenance facility for every vehicle at the beginning of schedule A, B, C and D. Say a vehicle ends at position B. Then the vehicle needs to be able to travel $B_1 + D_2 = 6{,}369.79km$ before attaining its kilometer limit in order to reach the maintenance facility. (The vehicle will have to switch at s_1 and s_3 to reach the maintenance facility on Sunday.) We can determine analogous constraints for A, C and D. Using this adaption of our MILP model, we can show that if the vehicles are only maintained three times during five weeks, one vehicle will violate its maximum kilometer limit in the seventh week. To do so, we bound the maintenances in the first 5 weeks by three and solve a 6 week scenario, where we require all vehicles to be able to reach the maintenance facility in week

7. This MILP is infeasible. Hence, we know that at least four maintenances are required within every consecutive five weeks of the solution that do not include the last two weeks.

We now apply this knowledge to a solution of the backtracking heuristic for a long scenario, i.e. 517 weeks. Given any five consecutive weeks of the first 515 weeks, there have to be at least four maintenances within these five weeks. Hence, we can deduct that that in the weeks 1 to 515, $515/5 \cdot 4 = 412$ maintenances have to be performed. This way, there can be one week without maintenance in weeks 1 to 5, 6 to 10 and so on. If there are fewer maintenances, we would find an $\ell \in \{0, \ldots, 103\}$, such set there are two weeks within week $5\ell + 1$ to week $5\ell + 5$ where no maintenance is performed. Thus, one of the vehicles would violate its kilometer limit in week $5\ell + 7$ and the underlying solution would not be feasible. Hence, we obtain a lower bound of 412 maintenances for a 517 week scenario. Since the backtracking heuristic gives us a solution with 412 maintenances, we know that this solution is optimal. For the 520 week scenario, the backtracking heuristic provides a solution with 415 maintenances, but we do not have a strict mathematical proof of the optimality with this method. Obviously, there still have to be at least 412 maintenances. We can even improve this bound to 414 by arguing that there have to be at least 4 maintenances within the weeks 514 to 518. For the 522 week scenario, we know that 416 maintenances are optimal. Given the optimal solution values of 7 and 11 for $\ell = 10$ and $\ell = 15$, it is quite possible that 415 maintenances is the optimal value for the 520 week scenario, but we did not formally prove this claim.

4 Discussion

We have considered the problem of integrating a long-term maintenance schedule and a weekly rotation plan on a real-world scenario. The specific issues that had to be addressed were that

- a weekly rotation plan that includes a suitable idle time had to be found,
- the determined weekly rotation plan did only allow for the maintenance of one vehicle per week,
- the weekly rotation plan did consist of several cycles, of which only one contained the necessary idle time and
- that the time scale of the long-term maintenances made planning over several weeks necessary.

We recognized that the lower density of the timetable on Sundays makes it possible to include a 24 hour idle time for one vehicle and introduced transitions between the cycles of the rotation plan, so that every vehicle can undergo the long-term maintenance treatments. We provided a practical algorithm that quickly calculates a rotation plan on a scale of several weeks or even years, which distributes the required maintenances uniformly and even determines gaps in the maintenance calendar.

We could proof the optimality of the obtained solutions for a time horizon of up to 15 weeks. For longer time horizons, we obtained the optimal solutions for scenarios of length $k = 5\ell + 2$ for a positive integer ℓ. For k we found solutions that are at least close to the optimum, but we did not prove their optimality.

Altogether, we provide a starting point for an estimate of the cost for a 10 year railway enterprise.

Acknowledgement

This work has been supported by the Research Campus MODAL Mathematical Optimization and Data Analysis Laboratories funded by the Federal Ministry of Education and Research (BMBF Grant 05M14ZAM). The authors would like to thank Christof Schulz and Steffen Weider of LBW Optimization GmbH for providing the problem and dataset. Further thanks goes to Julian Bushe, as well as Alistair Benford, John Curtis Lynch, and Ariel Nikas for assisting in the preliminary work for this article during the G-RIPS program 2018. Also, thanks to one of the anonymous referees for their extensive comments.

References

Abbink, E., Bärmann, A., Besinovic, N., Bohlin, M., Cacchiani, V., Caimi, G.,Dollevot, T., de Fabris, S., Fischer, F., Fügenschuh, A., Galli, L., Goverde, R.M.P., Hansmann, R., Homfeld, H., Huisman, D., Johann, M., Klug, T., Kroon, L., Lamorgese, L., Liers, F., Mannino, C., Medeossi, G., Pacciarelli, D., Reuther, M., Schlechte, T., Schmidt, M., Schöbel, A., Schülldorf, H., Stieber, A., Stiller, S., Törnquist Krasemann, J., Toth, P. and Zimmermann, U.T. 2018. *Handbook of Optimization in the Railway Industry*. Vol. 268 1 ed.
URL: *http://www.springer.com/us/book/9783319721521*

Boccia, M. and Mannino, C. and Vasilyev, I. 2013. "The dispatching problem on multitrack territories: Heuristic approaches based on mixed integer linear programming." *Networks* 62(4):315–326.

Borndörfer, R., Karbstein, M., Liebchen, C. and Lindner, N. 2018. A Simple Way to Compute the Number of Vehicles That Are Required to Operate a Periodic Timetable. In *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, ed. Ralf Borndörfer and Sabine Storandt. Vol. 65 of *OpenAccess Series in Informatics (OASIcs)* Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik pp. 16:1–16:15.
URL: *http://drops.dagstuhl.de/opus/volltexte/2018/9721*

Borndörfer, R., Reuther, M., Schlechte, T., Waas, K. and Weider, S. 2016. "Integrated Optimization of Rolling Stock Rotations for Intercity Railways." *Transportation Science* 50(3):863 – 877.

Borndörfer, R. and Liebchen, C. 2008. When Periodic Timetables Are Suboptimal. In *Operations Research Proceedings 2007*, ed. Jörg Kalcsics and Stefan Nickel. Berlin, Heidelberg: Springer Berlin Heidelberg p. 449–454.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L. and Wagenaar, J. 2014. "An overview of recovery models and algorithms for real-time railway rescheduling." *Transportation Research Part B: Methodological* 63:15–37.

Frisch, S., Hungerländer, P., Jellen, A. and Weinberger, D. 2018. "A Mixed Integer Linear Program for Optimizing the Utilization of Locomotives with Maintenance Constraints.".

Mannino, C. 2011. Real-time traffic control in railway systems. In *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, ed. Alberto Caprara and Spyros Kontogiannis. Vol. 20 of *OpenAccess Series in Informatics (OASIcs)* Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik pp. 1–14.
URL: *http://drops.dagstuhl.de/opus/volltexte/2011/3262*

Reuther, M. 2017. Mathematical Optimization of Rolling Stock Rotations PhD thesis.
URL: *https://depositonce.tu-berlin.de/handle/11303/6309*

Schlechte, T. 2012. Railway Track Allocation: Models and Algorithms PhD thesis.
URL: *http://opus.kobv.de/tuberlin/volltexte/2012/3427/pdf/schlechte_thomas.pdf*