Job order assignment at optimal costs in railway maintenance

Job Order Assignment at Optimal Costs in Railway Maintenance

Keywords: Railway Maintenance Planning, Job Order Scheduling, Operations Research

Abstract: Tamping is an important part of railway maintenance. Well tamped ballast reduces track irregularities and increases travel safety and comfort. But if the ballast is in a bad condition, the train speed must be restricted, which leads to delays and penalty costs for the operator. In this paper a novel model for the tamping scheduling problem in a short-term planning horizon is presented. In contrast to other railway maintenance scheduling problems the penalty costs caused by deferring tamping activities are considered in the scheduling process beside the travel costs. Three greedy heuristics are presented and compared in different benchmarks. An outlook discusses issues of interest for further research.

1 Introduction

Tamping is an important part of railway maintenance. Well tamped ballast reduces track irregularities and increases travel safety and comfort. But if the ballast is in a bad condition, the train speed must be restricted, which leads to delays and penalty costs for the operator. By scheduling the tamping works these penalty costs should be minimised together with incurred travel costs.

Within the ACEM-Rail project of the Europeans Seventh Framework Programme a novel solution approach for the tamping scheduling problem will be developed. In this paper the model is presented, first solution approaches are shown and a look out of the further research is given.

There are different papers dealing with railway maintenance scheduling.

(Higgins and Ferreira, 1999) develops an integer programming model to reduce train delays caused by maintenance activities. The problem is solved with a Tabu-Search heuristic.

Budai develops solution approaches for the preventive maintenance scheduling problem (PMSP) (Budai et al., 2004; Budai et al., 2009). Aim of the PMSP is to minimize track possession costs caused by scheduled routine activities and projects. For this purpose they combine as much tasks as possible. The problem is formulated as integer programming model and solved with heuristics and evolutionary computation approaches.

In (Miwa, 2002; Oyama and Miwa, 2006) an integer programming model for optimally scheduling a multiple tie tamper is shown. The objective is to maximize the improvement of track condition under bounded maintenance costs. The resulting schedule defines for each 10 day term where to locate the tamper and which lots to be maintained.

(Gorman and Kanet, 2010; Peng et al., 2011) present a time-space network to schedule larger projects to maintenance crews and execution weeks. They consider the specification of the crews (not every crew could execute all tasks), time windows (earliest start and latest resolving time of a project), travel costs, and cross-job constraints (precedence, non-concurrent, simultaneity).

An integer programming model to minimize the tamping effort is presented by (Vale et al., 2012). The optimal time-allocation to 90 day terms is searched, such that the track quality keeps a given level. They take into account four aspects of tamping: the time dependent deterioration process, the track layout, the imperfect track quality after maintenance and the track quality limits that depend on the maximal permissible train speed.

In (Quiroga and Schnieder, 2010) a heuristic approach for the tamping scheduling problem is presented. Aim is to find a set of N interventions, one per night, which maximise a defined objective function, e.g. the expected track condition one year later. An intervention is defined by a start depot, the tamping works, and an end depot.

The model presented here is different. On the one hand the planning horizon is short-termed – a few weeks or months – and the schedule defines explicitly the execution times of small tamping works. On the other hand so-called daily costs – penalty costs for restrictions in railway services caused by the untamped track – are considered in the decision process and will be minimised together with the costs for traveling between the tamping works.

The paper is organised as follows: In section 2 the short-term tamping scheduling problem is defined.

Three Greedy Heuristics are presented and compared in section 3. Issues of interest for further research – extensions of the model and concepts for general solution approaches – are presented in section 4.

2 Problem Formulation

The short-term tamping scheduling problem is defined as follows. Given a set of jobs that are defined by

- working duration
- daily costs
- location in the network.

There is a single tamping machine operating at the network and resolving jobs one after the other. The jobs are executed during the night in an eight hour working shift.

For each job the execution time has to be assigned in order to minimize maintenance costs. At the time at which the plan is calculated the tamping machine is located in section A and when all jobs are resolved, the machine will be parked at section B (the depot).

Each job refers to a small section of the track with a short working duration (about half an hour) such that the planning flexibility is high, but the number of jobs is not too large.

The daily costs are caused by traffic restrictions (like speed limitations) resulting from a bad track condition. They have to be paid for every day from the beginning until the job is resolved. If the track condition is still acceptable, the daily costs of the corresponding job are zero. Because of the short planning horizon (a few weeks or months) the time-dependence of track condition and thus the time-dependence of the daily costs is not considered.

The costs and time for traveling between jobs are calculated based on the locations in the network. The travel times contain 15 minutes for changeover between travel and working mode. Between jobs of consecutive track sections the time and costs for travelling are zero. In practice they are executed within one larger working step and without traveling. If the work will be continued in the next night at the same location the machine stays nearby over day. Thus no costs and time for travelling to a machine depot are incurred. If the work starts next night on another location the machine travels over day to the new location.

The material costs, machine rent and the employee's wages are not included in the model, because they are fixed and must be paid no matter if the job is executed today or in a week. Thus only the travel costs and the daily costs must be considered in an objective function.

The problem could be described as an integer programming model. Given a set of jobs J, $|J| = n$. Let for each job $j \in J$

- $d(j)$ the working duration
- $c_d(j)$ the daily costs
- $c_t(j,k)$ the travel costs between the jobs $j,k \in J$
- $t_t(j,k)$ the travel time between the jobs $j,k \in J$

Aim of the optimisation is to assign an execution time $t_e(j)$, such that the maintenance costs are minimal.

For the start point A and the end point B two artificial jobs j_A and j_B, are defined and $J_{AB} := J \cup \{j_A, j_B\}$. For both working suration and daily costs are zero. The travel costs and time are calculated from the locations in the network.

The execution time $t_e = (t_{e_1}, t_{e_2})$ consists of two components: the execution day $t_{e_1} \geq 1$ and the execution minute $t_{e_2} \in [T_1, T_2]$. The execution minute is restricted by the working shift, that starts at $T_1 \mathrel{\widehat{=}}$ 10:00 p.m. and ends at $T_2 \mathrel{\widehat{=}}$ 6:00 a.m. For j_A the execution time is set to $t_e(j_A) = t_{start}$, which is the time at which the plan is calculated.
The problem can be formulated as follows:

$$\begin{aligned} \min z = & \sum_{j \in J_{AB}} \sum_{k \in J_{AB}} c_t(j,k) \cdot x(j,k) \\ & + \sum_{j \in J_{AB}} t_{e_1}(j) \cdot c_d(j) \end{aligned} \tag{1}$$

with

$$\begin{aligned} & t_e(j_A) = t_{start} \\ & t_e(j_B) = (\infty, 0) \end{aligned} \tag{2}$$

$$x(j,k) = \begin{cases} 1 \text{ , if } \nexists l \in J_{AB} : \ t_e(j) < t_e(l) < t_e(k) \\ 0 \text{ , else} \end{cases} \tag{3}$$

subject to

$$\begin{aligned} t_e(k) \geq t_e(j) + d(j) + t_t(j,k) \\ \forall\, j,k \in J_{AB} : \ x(j,k) = 1 \end{aligned} \tag{4}$$

$$\left.\begin{aligned} t_{e_1}(j) \geq 1 \\ t_{e_2}(j) \geq T_1 \\ t_{e_2}(j) + d(j) \leq T_2 \end{aligned}\right\} \qquad \forall\, j \in J \tag{5}$$

With the objective function (1) the maintenance costs, i.e. the sum of the costs for traveling between the jobs and the daily costs, are minimised.
The maintenance machine starts at t_{start} in track section A and will be parked at track section B, when all jobs are resolved. The execution times of the respective jobs j_A and j_B are defined in equation (2). Binary variables $x(j,k)$ state whether two jobs are resolved directly one after the other and thus if travel costs occur or not (see (3)). A distinction whether two jobs are

on consecutive track sections is not necessary because the travel time and travel costs between jobs of consecutive track sections are zero. By constraint (4) and (5) it is ensured that the scheduled execution times are feasible.

The problem can be reduced to a job order assignment problem. Instead of determining the explicit execution time for each job, the order $J_O = (j_1, j_2, ..., j_n)$ to resolve the jobs $j_i \in J$ has to be specified. Then for the order $(j_A, j_1, j_2, ..., j_n, j_B)$ the minimal feasible execution times are calculated based on the equations (2)–(5).

The tamping scheduling problem is similar to the travelling salesman problem. Instead of visiting cities jobs are resolved. The main difference is the objective function. Not only the costs for travelling between the jobs are minimised, but also the daily costs, that depends on the execution day, have an influence on the solution quality.

Also, the problem shows resemblance to the rural postmen problem (a variant of the Chinese postmen problem). There a shortest closed path is searched in a graph that pass through an edge subset. The graph is given by the track network and the edges are the track sections with a job. Again, the objective function differs because of the daily costs.

Both problems are NP-hard, thus the tamping scheduling problem is NP-hard, too.

3 Greedy Heuristics

Dependent on the ratio between daily and travel costs different solution approaches are reasonable. For example, if the daily costs of all jobs are zero or very small, a heuristic that minimises travel costs leads to good results. Contrariwise, if the travel costs are much lower than the daily costs it is important to prefer jobs with high daily costs, even if this causes detours. Three different greedy heuristics are presented, which show good results in different benchmarks.

All greedy heuristics follows the same procedure, see algorithm 1.
Starting with an arbitrary job $k \in J$ the job order $J_O(k) = (k)$ is built up step-by-step: A job $j \in J$ that is not contained in the job order so far is selected by a heuristic-specific selection criterion and added to the end of $J_O(k)$. This step is repeated until all jobs are contained of the job order. Then j_A and j_B are added to complete the order, the execution times are determined based on the equations (2)–(5), and the costs of the job order are calculated.

Algorithm 1:
General Greedy Heuristic procedure

```
J*_O := ()
z* := ∞
for k ∈ J do
    J_O(k) := (k)
    repeat
        select j ∈ J \ J_O(k)
        append j at the end of J_O(k)
    until J \ J_O(k) == ∅
    determine execution times
    calculate maintenance costs z(J_O(k))
    if z(J_O(k)) < z* then
        z* := z(J_O(k))
        J*_O := J_O(k)
    end
end
return J*_O
```

To improve the solution quality a job order $J_O(k)$ is generated for each job $k \in J$ and the best of these job orders is selected.

In the following the three selection criterions are presented. In section 3.4 a comparison of the three heuristics is made.

3.1 Nearest Job

At each step of algorithm 1 the nearest uncontained job, thus the one with the lowest travel costs to the last job in the order, is selected and added. If there are two or more jobs with the same distance, then the job with the highest daily costs is chosen.

With the Nearest Job Greedy Heuristic the travel costs are kept small, but the daily costs play a minor role in the decision process.

3.2 Most Expensive Job

At each step of algorithm 1 the most expensive job is selected and added to the job order. If there is more than one job with the same costs, the nearest of them is selected. In our model it is assumed that the daily costs of consecutive track sections typically fluctuate. Then these selection criterion leads to high fragmented solutions. Often the machine travels to resolve only a single job, and then travels to a different track section for the next job. To avoid such a fragmentation of the job order, direct neighboured jobs are preferred. That means if there is a job that could be reached without traveling and has daily costs in the same cost range as the most expensive one, this job is added instead of the most expensive job.

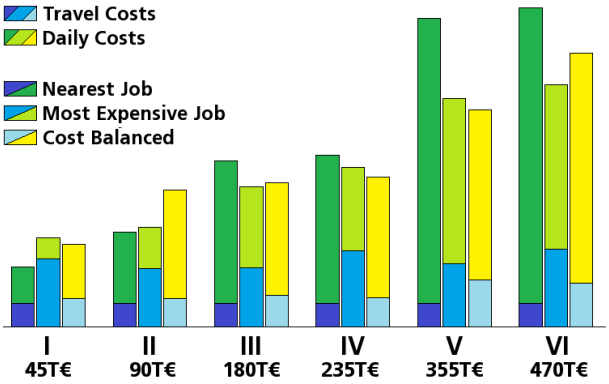

Figure 1: Results of the three greedy heuristics in six benchmarks.

The Most Expensive Job Greedy Heuristic usually results in a job order where the jobs with non-zero daily costs are always added first. The jobs without daily costs are added similar to the Nearest Job Greedy Heuristic.

3.3 Cost Balanced

Here, both costs are considered by defining a combined cost measure. With this the travel costs and a very rough estimation for the savings in daily costs are summed. For each job that is not contained in the job order the cost measure is calculated by

$$m_c(j) = c_t(j_l, j) - \alpha \cdot c_d(j)$$
$$\text{with } \alpha = \frac{1}{2} \cdot \frac{\text{\# unresolved jobs}}{\text{\# jobs resolved per day}} \quad (6)$$

where j_l is the last job in the current order and α is an estimation on the number of days the job will stay unresolved until now. If one of the direct neighboured jobs has daily costs in the same cost range as j_l, this job is chosen. Again, this avoids a fragmentation of the solution like in the Most Expensive Job Greedy Heuristic. Otherwise the job with the minimal cost measure m_c is selected.

With the Cost Balanced Greedy Heuristic both kinds of costs are considered in the decision process.

3.4 Comparison

The three greedy heuristics are tested in different benchmarks. In figure 1 the results of the heuristics are compared in six benchmarks (I–VI). All benchmarks are built up of 1415 jobs on a network with more than 1400 km of track. 353 of them are afflicted with daily costs ranging from 45,000 € per day (benchmark I) up to 470,000 € in benchmark VI. The jobs with non-zero daily costs are located in the centre of the track network. The travel costs are 1000 € per kilometre. To resolve all jobs at least 184,500 € have to be paid for travelling.

In the bar graphs the travel costs (lower part) and the daily costs (upper part) are stacked. The left bar shows the results of the Nearest Job Greedy Heuristic (short NJ), in the middle bar the results of the Most Expensive Job Greedy Heuristic (MEJ) are plotted, and the right bars represents the results of the Cost Balanced Greedy Heuristic (CB).

Due to the fact that the benchmarks only differ in the daily costs, NJ always generates the same solution for a certain first job $k \in J$. In each benchmark the same job order J_O^* is selected and thus always the same travel costs occur. Only the daily costs differ due to the different daily costs in the benchmarks.

Benchmark I – III and V are very similar: in benchmark II the daily costs of benchmark I are doubled, in benchmark III they are fourfold and in benchmark V the costs are eightfold.

In benchmark I and II the share of the daily costs is low. NJ starts with the most expensive track sections and resolves the other jobs without much traveling. This leads to a plan with low travel and daily costs. The heuristic CB leads to a solution with higher daily and higher travel costs than NJ. One reason is that the daily costs are underestimated and that the jobs with non-zero daily costs are resolved as long as the detour is short. So the travel costs are higher as with NJ and there are no savings in daily costs. By MEJ the jobs afflicted with daily costs are resolved first. This leads to long detours and to an increase in travel costs, which cannot be compensated by the savings in daily costs.

In benchmark III NJ obtains the same plan as in benchmark I and II, but the daily costs are much higher, so the expensive jobs should be preferred. Also with CB the daily costs are underestimated and thus they have a high share in the costs of the job order. By MEJ the most expensive jobs are resolved first with long detours, but this time this can be is compensated by the savings in daily costs.

In benchmark V the daily costs have a huge share in the overall costs. The cost measure of jobs with non-zero daily costs is low and these jobs will be resolved first. Because of the consideration of travel costs in the decision process CB leads to better results than MEJ. The detours are shorter, but the jobs afflicted with daily costs are nevertheless resolved first.

In the benchmark IV and VI the daily costs differ even more. There are 115 jobs with high daily costs in the centre of the network on a line. The other jobs have only small daily costs and are located around. Resolving the jobs with small daily costs first leads to long detours, which could be compensated by the

savings in daily costs only in benchmark VI. In benchmark IV the small daily costs are low enough, so that long detours to resolve them are not necessary. Therewith CB obtains the best result.

The heuristics have also been tested in further benchmarks, with different jobs on the network and different assignments of daily costs to jobs. We noticed that it is hard to predict which heuristic obtains the best result. But some statements are possible:

- If the daily costs have a small share in the maintenance costs, then NJ and CB obtains good results, because they minimise the travel costs.
- If the daily costs are high and scattered over the track network, then CB obtains the best results. MEJ resolves the expensive jobs more ordered by daily costs and thus travels crisscross through the network, where CB resolves the expensive jobs first, too, but ordered by location. The resulting savings in travel costs compensate the small increment in daily costs.
- If the daily costs are medium or high and clustered, then MEJ obtains good results, because of the lower travel effort. Within CB the daily costs often are underestimated in this type of benchmarks, which leads to worse results.

4 Issues of Interest for Further Research

The model presented in section 2 will be extended by some additional points, see section 4.1.

In the further research solution approaches will be developed that are suitable for arbitrary ratios between daily and travel costs, and that can handle the additional restrictions from the model extension. The metaheuristic Simulated Annealing (section 4.2) and a multilevel solution approach (section 4.3) will be implemented and compared in terms of solution quality and computation time in different benchmarks.

4.1 Model extension

(1) K machines are available to resolve the jobs. So K disjunctive job orders have to be defined and non-simultaneous restrictions must be considered (e.g. a minimal distance must be kept between two tamping machines).
(2) Maintenance is not always possible. Due to night trains, freight traffic, and other maintenance activities the possible execution times of a certain job will be restricted. In the model time windows will be defined for each job when maintenance is possible. If the time window ends before the maintenance works are finished, the crew must leave the track and wait for the next time window.
(3) Consideration of depots. In some railway networks it is not possible or not common that the maintenance machine stays close to the track over the day. There the machine stays in a depot. Then not only the jobs are scheduled, but also the best depot to stay over day must be determined and additional travel costs must be considered.

4.2 Simulated Annealing

The basic idea of Simulated Annealing comes from annealing processes in metallurgy. After heating the metal the atoms are inordinated. Through the slow cooling process they have enough time to order themself and to form crystals. This leads to a low-energy state.

The algorithm starts in the "hot stage" with a initial solution s and a high temperature T. Then the solution $\tilde{s}$ is created by modifying s. Dependent on the temperature T, the probability $P(s,\tilde{s},T) = \min\{1, e^{\frac{z(s)-z(\tilde{s})}{T}}\}$ to accept $\tilde{s}$ is calculated. If $\tilde{s}$ is accepted, then $s := \tilde{s}$. After that the temperature is cooled. This step – modify solution s, accept the modified $\tilde{s}$ with probability $P(s,\tilde{s},T)$, cool T – is repeated until a given minimal temperature is reached. Then the best solution is returned.

The challenge in the design of a proper Simulated Annealing approach for an optimisation problem is the definition of a modification heuristic to get the solution $\tilde{s}$ and of a cooling schedule for T.

For out tamping scheduling problem at first methods from solving the TSP are implemented. The SA starts with a random job order. The solution is modified by a 2-opt method (Meer, 2007): Two indices $0 \leq i < j \leq n+1$ are chosen randomly and the suborder between i and j is inverted. The temperature T is cooled exponentially. This leads to a fast improvement of the solution quality, but at the end the solution remains a bit fragmented. To smooth the solution a post-optimisation method will be developed. With this, irregularities – like jumping over a few jobs and resolve them later – will be removed without losing the solution structure.

4.3 Multilevel Branch and Bound

Multilevel solution approaches are common methods in graph partitioning (Karypis and Kumar, 1996) and VLSI-design (Cong et al., 2005). A multilevel solution approach consist of three steps:

1. Coarsening: merge objects to super objects

2. Solving: find a (nearly) optimal solution for the super object problem
3. Refinement: transfer the solution back to the original problem and post-optimise

Step 1 is repeated until the number of super objects fall below a given threshold. Then the small problem is solved (step 2). Step 3 is executed as often as step 1. On each level the post-optimisation step can be used to improve the solution quality.

For the tamping scheduling problem a multilevel Branch and Bound method seems to be a promising approach. In practise, mostly some consecutive jobs are resolved in one working step without travelling. Thus jobs of consecutive track sections can be merged to one super job. Therewith the problem size is decreased and the application of exact methods, like a Branch and Bound approach, is possible. The solution obtained is transferred back to the original problem. In the post-optimisation step the super job structure will be broken by rearranging single jobs in order to improve the solution.

Acknowledgements

We acknowledge the financial support from the European Communitys Seventh Framework Programme under Grant Agreement no. 265954, ACEM-Rail project. The EC is not liable for the use that can be made of the information contained herein.

REFERENCES

Budai, G., Dekker, R., and Kaymak, U. (2009). Genetic and memetic algorithms for scheduling railway maintenance activities. Technical report, Econometric Institute, Erasmus University Rotterdam.

Budai, G., Huisman, D., and Dekker, R. (2004). Scheduling preventive railway maintenance activities. Technical report, Econometric Institute, Erasmus University Rotterdam.

Cong, J., Fang, J., Xie, M., and Zhang, Y. (2005). Mars-a multilevel full-chip gridless routing system. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(3):382 – 394.

Gorman, M. F. and Kanet, J. J. (2010). Formulation and solution approaches to the rail maintenance production gang scheduling problem. *Journal of Transportation Engineering*, 136:701–708.

Higgins, A. and Ferreira, L. (1999). Scheduling rail track maintenance to minimise overall delays. In *Proceedings The 14th International Symposium on Transportation and Traffic Theory*.

Karypis, G. and Kumar, V. (1996). Parallel multilevel graph partitioning. In *Parallel Processing Symposium, 1996., Proceedings of IPPS '96, The 10th International*, pages 314 –319.

Meer, K. (2007). Simulated annealing versus metropolis for a tsp instance. *Information Processing Letters*, 104(6):216 – 219.

Miwa, M. (2002). Mathematical programming model analysis for the optimal track maintenance schedule. *QR of RTRI*, 43:131–136.

Oyama, T. and Miwa, M. (2006). Mathematical modeling analyses for obtaining an optimal railway track maintenance schedule. *Japan J. Indust. Appl. Math.*, 23:207224.

Peng, F., Kand, S., Li, X., and Ouyang, Y. (2011). A heuristic approach to the railroad track maintenance scheduling problem. *Computer-Aided Civil and Infrastructure Engineering*, 26:129–145.

Quiroga, L. M. and Schnieder, E. (2010). A heuristic approach to railway track maintenance scheduling. *WIT Transactions on The Build Environment*, 114:687–699.

Vale, C., Ribeiro, I. M., and Calcada, R. (2012). Integer programming to optimize tamping in railway track as preventive maintenance. *Journal of Transportation Engineering*, 138:123–131.