Robust Train Routing and Online Re-scheduling

Alberto Caprara[1], Laura Galli[1],
Leo Kroon[2,3], Gábor Maróti[3], and Paolo Toth[1]

[1] D.E.I.S., University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy.
{alberto.caprara,l.galli,paolo.toth}@unibo.it
[2] Netherlands Railways, Utrecht, The Netherlands.
[3] Rotterdam School of Management, Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR, Rotterdam, The Netherlands. {gmaroti,lkroon}@rsm.nl.

Abstract. Train Routing is a problem that arises in the early phase of the passenger railway planning process, usually several months before operating the trains. The main goal is to assign each train a stopping platform and the corresponding arrival/departure paths through a railway station. It is also called Train Platforming when referring to the platform assignment task. Railway stations often represent bottlenecks and train delays can easily disrupt the routing schedule. Thereby railway stations are responsible for a large part of the delay propagation in the whole network. In this research we present different models to compute robust routing schedules and we study their power in an online context together with different re-scheduling strategies. We also design a simulation framework and use it to evaluate and compare the effectiveness of the proposed robust models and re-scheduling algorithms using real-world data from Rete Ferroviaria Italiana, the main Italian Railway Infrastructure Manager.

1 Introduction

The Train Routing problem arises in the early phase of the passenger railway timetabling process, after the departure and arrival times have been defined. The main goal is to route the trains through a railway station, for example a busy station, and assign them a stopping platform. Thus it is also known as Train Platforming. This problem represents a major issue for medium and large sized stations. Such stations are rather common throughout Europe, have complex topologies and can severely impact on the train schedule operation. More precisely, solving a train routing problem for a given railway station means considering all the trains (in the timetable) passing through it and assigning each of them (*i*) a stopping platform and (*ii*) a pair of arrival and departure paths to reach and leave the platform, respectively.

Unfortunately even an optimal plan can be rather useless in a real-life context when the inevitable disturbances affecting the system modify the conditions we have optimised for. For this reason the latest research projects have been focusing on dynamic aspects. These approaches can be divided into two main branches: robust planning and online re-scheduling. Robust planning, on one

hand, is meant to reduce delay propagation in a railway system, i.e., train conflicts, thus limiting the effect of disturbances on the system. Online re-scheduling, on the other hand, deals with recovery strategies that can be used to react real-time, whenever the disturbances of the system hinder the nominal plan and a new one is needed according to the current conditions. Our goal is to use a simulation framework to investigate the effectiveness of different combinations of robust plans and re-scheduling strategies for train routing.

2 Motivation and Outline

Substantial research has been conducted on the train routing problem. For example, [1], [5], [6], [9] [10], [13], and [14] presented optimisation models and algorithms for train routing, whereas [8] proposed a theoretical study of the track assignment problem and show how some variations can be solved as special classes of graph colouring problems. More recently, [2] developed a model to find delay-tolerant train routes, while [7] considered a re-scheduling setting. Both [3] and [12] described models to minimise the number of crossing train routes (and the time overlap thereof). Finally [4] presented an exact (delay-)robust optimisation framework for train routing. There is, however, very little literature available on robustness and re-scheduling together. Our paper sets itself in this context.

The goal of this work is threefold. The first goal is to extend the routing model of [3] to robustness considerations, describing some variations where robustness is enhanced either by increasing the delay absorption capacity or by explicitly providing potential recovery possibilities. The second goal is to design different exact re-scheduling algorithms according to different recovery strategies. In particular, we consider three different recovery strategies, implemented as MIPs. The first strategy consists in simply propagating the delay, the second strategy relies on robust extra-resources (backup platforms), and the third strategy allows unlimited changes to the nominal plan. The third goal is to assess the performance of the robust plans and re-scheduling strategies in a simulation framework. In fact, robustness and recoverability are intriguingly difficult notions to quantify. In this research we propose a simulation framework to compare different combinations of routing plans and recovery strategies. More specifically, given a timetable of an entire day, a routing plan and some randomly-generated delays, the re-scheduling algorithms are applied to resolve the routing conflicts. In this way, the simulation allows us to compare the robustness of different plans together with different recovery strategies, the main criterion in the comparison being the global train delay.

This paper is organised as follows. In Section 3 we describe the Italian train routing problem, that represents our real-world case study, and we also sketch the model of [3] that represents the basis for our research. In Section 4 we describe some variants of the original model and in Section 5 we introduce three different re-scheduling strategies, implemented via MIPs. Finally we propose a

simulation framework in Section 6. Section 7 is devoted to computational results. In Section 8 we draw some conclusions and observations for future research.

3 The Train Routing Problem

In this section we recollect the train routing problem presented in [3]. The problem, as described by the main Italian Railway Infrastructure Manager (Rete Ferroviaria Italiana), aims at defining a routing plan for a given railway station, after the corresponding timetable has been defined. We are given a timetable containing a set T of trains that will either stop or travel through the railway station. The timetable defines arrival and departure times and directions for every train $t \in T$. Information on the railway station topology is also given. A railway station can be represented as a mesh of tracks connecting the railway line directions to stopping platforms. A path is a sequence of tracks connecting a direction to a stopping platform or vice versa. Different paths can share the same track or other physical resources along their lines, and in this case they are considered *incompatible*. Hard constraints forbid the assignment of incompatible resources at the same time or within a safety limit (few minutes) one after another. The goal of the train routing problem is to define for every train $t \in T$ a stopping platform and two paths, connecting the platform to the arrival and departure directions of train t. It is also possible to apply small changes to the timetable, called *shifts*. Hence, the routing plan can define new arrival and departure times for every train. In this way, the routing phase feeds back to the previous timetabling phase. The model presented in [3] is based on the concept of *patterns*. A pattern encapsulates all the information about the resources assigned to a train: stopping platform, arrival and departure paths, arrival and departure shifts. Clearly, each train $t \in T$ defines its own set of feasible patterns $\mathcal{P}_t$ according to its arrival and departure directions. Incompatibilities are represented using a graph whose nodes correspond to train patterns, and hard constraints are expressed via a set $\mathcal{K}$ of cliques of incompatible patterns. As explained in [3] the model is solved using pricing and separation techniques due to the large number of variables and constraints. This solution method is applied to all the three variants that we will present in the next section. Thus, for details on how to solve the corresponding MIP models, the reader can refer to [3].

4 Robust Planning

In this section we present three different variants of the model of [3]. Binary variables $x_{t,P}$ represent the assignment of pattern $P \in \mathcal{P}_t$ to train $t \in T$, and s_t are binary variables used to cancel (i.e., not assign) train $t \in T$. A large penalty M_t is associated with variable s_t in the objective function to minimise such occasions. The cost $c_{t,P}$ of a pattern P for train t represents the quality of the corresponding assignment for the given train (i.e., preference platforms, changes with respect to the nominal scheduled times, etc.).

4.1 Basic Platforming

In the first (non-robust) variant of the routing model, called *basic*, we simplify the original model of [3] by considering exclusively the cost of the patterns, without any additional fixed cost for the platforms used. In fact, in a robust model it may be desirable for the trains to spread platform occupation among different resources.

$$\min \sum_{t \in T} \sum_{P \in \mathcal{P}_t} c_{t,P}\ x_{t,P} + \sum_{t \in T} M_t s_t \tag{1}$$

subject to

$$s_t + \sum_{P \in \mathcal{P}_t} x_{t,P} = 1, \qquad t \in T, \tag{2}$$

$$\sum_{(t,P) \in K} x_{t,P} \leq 1, \qquad K \in \mathcal{K}, \tag{3}$$

$$x_{t,P}, s_t \in \{0,1\}, \qquad t \in T,\ P \in \mathcal{P}_t. \tag{4}$$

Constraints (2) either assign a pattern or cancel a train. Constraints (3) forbid resource conflicts and use cliques $K \in \mathcal{K}$ in order to strengthen the formulation.

4.2 Increasing the Absorption Capacity

The most straightforward way to cope with small delays without any particular re-routing strategy is to simply propagate the delay. This means resolving the given scenario by preserving the nominal scheduled train order. If a train is delayed then all its resources are locked and the subsequent trains, if allocated to one or more of these, are pushed-back, waiting for the corresponding resource to be freed up. For this reason, it is important to *increase the delay absorption capacity* of the routing plan. In fact, in the second variant, robustness is captured by penalising the cases in which incompatible resources are utilised by two trains in a short succession. These situations may give rise to conflicts in case of delays and thereby lead to propagation of such delays. So the aim is to spread the load on the infrastructure in time and space. A routing plan optimised for this criterion is expected to cope with small delays, without substantial recovery actions. The MIP formulation that we use has the following objective function:

$$\min \sum_{t \in T} \sum_{P \in \mathcal{P}_t} c_{t,P}\ x_{t,P} + \sum_{t \in T} M_t s_t + \sum_{(t_1,t_2) \in T^2} \sum_{P_1 \in \mathcal{P}_{t_1}} \sum_{P_2 \in \mathcal{P}_{t_2}} c_{t_1,P_1,t_2,P_2}\ x_{t_1,P_1}\ x_{t_2,P_2} \tag{5}$$

This objective function is subject to constraints (2)-(4). The cross-penalty c_{t_1,P_1,t_2,P_2} in the objective function depends on the distance (in time) among the utilisations of the incompatible resources (platforms and paths) associated with the two patterns P_1 and P_2 assigned to trains t_1 and t_2 respectively: the closer in time, the higher the penalty. A detailed discussion on how to linearise (5) in an effective way can be found in [3].

Even though the quadratic component can express a wide array of optimisation criteria, each of them would require a fine-tuning of the coefficients in the objective function. A common way to perform such a tuning is to apply scenario-based models such as stochastic programming. Still, the performance of the overall method would strongly rely on the chosen distribution. In this paper, we limit ourselves to a naive definition of these quadratic costs, see [11] for details.

4.3 Backup Platforms

In this third version, we intend to fight delay propagation by allowing each train to use a *backup* (i.e., recovery) platform for a given time period, whose length coincides with the maximum delay we intend to prevent. In this model, each train is assigned a *primary* and a *backup* platform. A platform can only be backup for a train if no other train simultaneously uses it as primary at the same time. The primary platform is intended to be used whenever possible, while the backup platform provides re-scheduling possibilities when the train is delayed. Suppose that train t has arrival time a and departure time d, and let ℓ be a limit on the delays that we are dealing with. Then train t occupies its backup platform during the time interval $[a; d + \ell]$. Note that the occupation of the backup platform does not depend on the shift applied to the train, since this is generally small with respect to ℓ. Platforms that have nearly identical approach paths are called *neighbouring* platforms. Note that the precise definition depends on the infrastructure of the studied railway station. We suggest to choose neighbouring primary and backup platforms for each train. The reason for this choice is that, if a train is moved to a neighbouring platform, it is very likely to use paths with the same structure as the primary one, thus it will not introduce many additional conflicts with other paths already assigned. Moreover, passengers will be less disappointed if asked to move to a close-by platform.

Backup platforms could be incorporated into the basic model (1)-(4) simply by extending the notion of patterns. However, such an extension would significantly increase the complexity of the pricing phase. This motivates a different approach. For each train $t \in T$, we define an interval I_t corresponding to the time window during which the backup platform for train t must be available, and we use binary variables $u_{\pi,t}$ that take value 1 if train t can use backup platform π in its corresponding time interval I_t. With this in mind we require for each train t the assignment of a primary platform *and* of a backup platform in the neighbourhood of the primary one:

$$\sum_{\pi \in N(b)} u_{\pi,t} \geq \sum_{P \in \mathcal{P}_t(b)} x_{t,P}, \qquad t \in T, b \in B, \tag{6}$$

$$u_{\pi,t} \in \{0, 1\}, \qquad \pi \in B, t \in T. \tag{7}$$

Here B is the set of platforms in the railway station, $N(b)$ is the set of neighbouring platforms of b and $\mathcal{P}_t(b)$ is the set of all the patterns associated with

train t that use b as primary platform. Of course, we also require the backup platform to be free from any primary assignment for each possible arrival instant $m \in I_t$:

$$\sum_{t' \in T-\{t\}, P \in \mathcal{P}_{t'} : \pi^P_{t'} = \pi, m \in [a^P_{t'}, d^P_{t'}]} x_{t',P} \leq 1 - u_{\pi,t}, \qquad \pi \in B, t \in T, m \in I_t. \quad (8)$$

Here the left-hand side considers all the patterns associated with trains $t' \in T-\{t\}$, such that π is their primary platform ($\pi^P_{t'} = \pi$) and their occupation time interval $[a^P_{t'}, d^P_{t'}]$ contains instant $m \in I_t$. Note that there is no cost associated to backup platforms. Moreover a backup platform can be assigned simultaneously (as backup, not as primary) to several trains.

Hence, the overall model has objective function (1) and constraints (2)-(4) and (6), (7), (8).

5 Online Re-scheduling

Once a (robust) routing plan is given, one has to test its effectiveness against delays by applying to it a recovery strategy which may be a general one or may be tailored according to the type of plan.

5.1 Recovery strategies

In this paper we consider three different recovery strategies.

Delay Propagation This is a general strategy in which each train keeps its nominally-assigned resources and the order of the trains on the trajectories is not modified. Thus conflicts are resolved by adjusting the arrival and departure times (i.e., by possibly propagating the delay). This strategy can be applied to any of the plans described in the previous section.

Backup Platform This strategy assigns trains either to their primary platform or to their backup platform, whichever leads to less delays. The associated re-scheduling algorithm tries to exploit the recovery resources provided by the plan in order to minimise the delay propagation. This strategy is only applicable for the backup robust type of plan, since we need specific information on the recovery resources.

Full This general recovery rule is the most powerful one. It allows any kind of re-scheduling action. Hence, trains may be assigned to completely different patterns with respect to the nominal plan. Being a general strategy, it can be applied to all types of plans.

5.2 Re-scheduling algorithms

For each of these strategies, we implemented an *optimisation-based* re-scheduling algorithm obtained by expressing mathematically the corresponding rules. This can be done by extending model (1)-(4). The re-scheduling process is performed by solving the associated mathematical program.

The objective function The simulation process provides each train $t \in T$ with the estimated arrival time $ETA(t)$ which is defined as the nominal arrival time plus the external delay. In our evaluation framework, the values $ETA(t)$ are provided by the simulation engine (as will be explained in the next section). The objective function of all re-scheduling algorithms aims at minimising the total *propagated* delay. We define the *propagated* delay for train t, Δ_t, as the *realised departure time* of t minus the *nominal departure time* of t. In other words, the propagated delay is the *delay upon departure*, i.e., the delay with respect to the nominal release time of the departure path. The reason for doing this is that the delay upon departure will affect subsequent railway stations in the train schedule and represents the delay exported by the railway station. Note that by minimising the delay upon departure we often also minimise the *delay upon arrival*, i.e., the delay with respect to the nominal release time of the arrival path, especially for tight train schedules. Clearly, the realised departure time of t depends upon the pattern $x_{t,P}$ assigned to t. $\Delta_{t,P}$ represents the delay propagated by train t if assigned to pattern P.

Shift constraints The MIP models for the re-scheduling algorithms extend the original structure of (1)-(4). On the other hand, the re-scheduling models allow rather large additional arrival/departure shifts in order to be able to deal with train delays. Hence, the number of admissible patterns is much higher than in the planning models. Further, the queues of trains on the in-bound tracks require additional constraints. In fact, trains that are delayed on arrival must wait outside the railway station on some in-bound tracks, since overtaking among trains on arrival is not possible. We model the in-bound queues by forbidding shift values that correspond to a train overtaking another. In other words if δ_t^a is the shift on arrival for train t, then $ETA(t_1) < ETA(t_2)$ implies $ETA(t_1)+\delta_{t_1}^a < ETA(t_2) + \delta_{t_2}^a$ for trains t_1 and t_2 entering the station from the same direction. We can express this with constraints of the form:

$$\sum_{(t,P)\in S} x_{t,P} \leq 1 \qquad S \in \mathcal{S} \tag{9}$$

These constraints forbid the simultaneous occurrence of a given sets S of patterns, if these together indicate that a train joins the queue of waiting trains earlier but leaves it later; $\mathcal{S}$ denotes a family of all such sets. For illustration, consider an example with three different trains (t_1, t_2, t_3), whose estimated arrival times are respectively $ETA(t_1)$ =10:00, $ETA(t_2)$ =10:02, $ETA(t_3)$ =10:05. For simplicity, suppose the maximum allowed shift forward on arrival (max δ_t^a) to be 5 minutes for all trains. Whenever we apply some shifts on these trains, we want to keep the original order corresponding to the ETA instants, i.e., in this case,: $t_1 \prec t_2 \prec t_3$.

In Figure 1 each big oval represents a train, with all the possible arrival times (represented by the associated nodes) according to the different shift values. An edge from one node to another means that the corresponding two shift values are incompatible. This happens either when the shifts make the two arrival

times equal or change the order. Note an important characteristic of this *shift incompatibility graph*: edges go from one train to the following one, but because of the transitive property, we do not need to specify the edges connecting non consecutive trains.

Each *shift-node* within a big oval represents all the patterns for the associated train that use a particular shift value s. The weight of a node is defined as the sum of all pattern variables associated with patterns that belong to the given node. Thus, separation of constraints (9) can be done by looking for maximum weight cliques on this shift incompatibility graph. This turns out to be easy by dynamic programming as explained in [11].

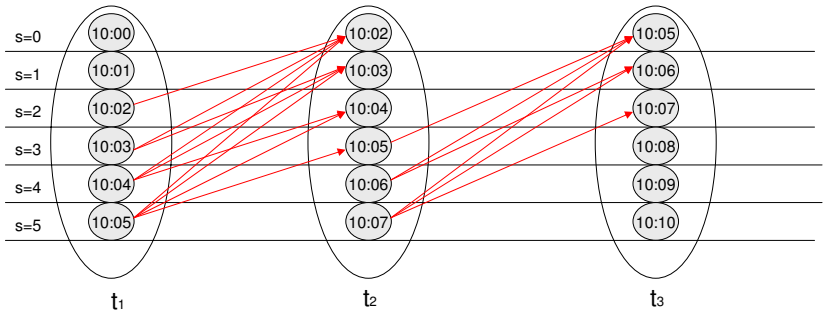

Fig. 1. Shift incompatibility graph.

The overall re-scheduling model reads as follows:

$$\min \sum_{t \in T} \sum_{P \in \mathcal{P}_t} \Delta_{t,P}\ x_{t,P} \tag{10}$$

subject to (2)-(4) and (9).

6 Simulation Framework

In the previous section we presented several ways to create robust routing plans as well as several re-scheduling strategies. In order to analyse the performance of these approaches, we designed a simulation framework. Its input consists of (*i*) a nominal routing plan, (*ii*) a re-scheduling strategy and (*iii*) a probability distribution for the train delays. Generally speaking, the framework computes, using a rolling horizon, the outcome of the re-scheduling process when the nominal plan is executed subject to the selected external delays. Then the robustness of the nominal plan and the effectiveness of the re-scheduling strategy are measured by the cumulative arrival and departure delays. In this section we discuss the simulator in detail.

The simulator first generates the external delay for each train $t \in T$ according to a given probability distribution (discussed in the next section). The estimated arrival time $ETA(t)$ equals the nominal arrival time of t plus the external delay of t. That is, $ETA(t)$ is the earliest time instant when train t can enter the station.

Once all the ETA values have been defined, the framework simulates a day, starting at 0:00 and advancing the simulated time by 1 minute in each iteration. In one iteration, the simulator collects the trains that are to arrive in the forthcoming 1 hour and passes these trains to the re-scheduler algorithm. That is, the simulator works with a 1-hour rolling horizon. The simulator also maintains the *current schedule* which is the train-pattern assignment of the already re-scheduled trains. In particular, the simulator provides for each train the ETA, the planned patterns as well as the pattern assigned by the current schedule (if any). From this input, the re-scheduler assigns a pattern to each train, and the simulator updates the current schedule accordingly. This process is repeated until the simulated time reaches the end of the day. Since this requires the solution of the rescheduling model for each simulated minute, in order to have reasonable computing times we stop as soon as the first integer solution is found. This is essentially a *diving heuristic* as explained in [3].

The simulator framework addresses two issues that are important for realistic train routing applications. The first issue is that some formerly taken decisions (such as those on the platform of a certain train) may not be altered any more because the decision has already been announced or because the train has already arrived. Therefore the simulator restricts the re-scheduling possibilities as follows.

- The platform assigned to a train can be modified (with respect to the current schedule) only if more than 10 minutes are left till the train's arrival.
- The arrival shift (i.e., the additional delay added by the re-scheduler upon arrival) can be changed only if the train has not arrived yet.
- The departure shift can be changed only if the train has not departed yet.

The second issue concerns the accuracy of the delay estimates. In real-life applications, the actual external delay is not known *exactly* till few minutes before the realised train arrival. Instead, a gradually improving estimate is available. The simulator may incorporate this uncertainty by providing distorted ETA values to the re-scheduler for trains that have more than few minutes till their arrival. We note that our preliminary computational results do not address this issue yet.

The quality of the re-scheduling process is measured by the cumulative departure (or arrival) delay, defined as the difference between the realised and nominal departure (or arrival) times, summed over all trains. We compare these values to the *cumulative external delay* defined as the sum of all external delays. By this comparison, one can analyse the effect of a combination of a nominal plan and a re-scheduling strategy have on the punctuality at the considered railway station, and on the propagation of delays through the network.

7 Preliminary Computational Results

We performed our preliminary computations on a real-world instance of Rete Ferroviaria Italiana associated with the station of Genova Porta Principe together with the corresponding tentative timetable. The station has 10 platforms and the timetable concerns a 12 hour period, contains 119 trains, and has an average dwell time of 5.4 minutes. The computations have been carried out on a standard PC with an Intel Duo Core 3.33GHz processor and with 3GB of internal memory under Windows XP. The algorithms are implemented in C/C++ using CPLEX 11.1. First we compute the three nominal plans: BASIC, using the basic model of Section 4.1; AC, according to Section 4.2; and BACKUP, according to Section 4.3. This is done using a branch-and-cut-and-price heuristic approach. BASIC and BACKUP can be solved in less than 10 minutes for both instances. In particular backup constraints do not seem to complicate the structure of the model. AC turns out to be more demanding (about 20 minutes of computation) since it requires many additional and complicating constraints for the linearisation of the objective function (see [3]).

In order to investigate the online re-scheduling problem, we consider the three strategies described in Section 5; we refer to the strategies as PROP, BACKUP and FULL, respectively. While the FULL strategy lends itself to be applied to every nominal plan, the BACKUP strategy can be used only for the BACKUP plan, whereas it does not make sense to apply the PROP strategy to a BACKUP plan as it would not use backup platforms at all.

The external delays are generated using a truncated exponential distribution, that is, an exponential distribution where large values are cut off and whose mean is 4 minutes. For the sake of simplicity we assume that all the external delays follow the same distribution.

For each nominal plan and re-scheduling strategy, we consider 15 random realisations of the random external delay values. Note that the number of trains varies a little as well as the cumulated external delays. This is because a few trains are cancelled by the robust planners. This may slightly affect the comparison of the delay propagation, but only marginally since the number of cancelled trains is small, so in these preliminary computations we did not consider this aspect. Table 1 presents the outcome of the simulation, showing the average of the cumulative external, arrival and departure delays. Further, the last two columns indicate the *increment* of the arrival and departure delays compared to the external delays.

From the results, it turns out that the FULL strategy is rather insensitive to the nominal plan. On the other hand, for the PROP strategy the AC plan appears to be much better than the BASIC one with an increment of the cumulative arrival delay of 8% rather than 15% and a decrement of the cumulative departure delay of 4% rather than an increment of 7%. Finally, the BACKUP strategy (applied to the BACKUP plan) does better than the PROP one applied to the AC plan, since the increment of the cumulative arrival delay is only 4% (the decrement of the cumulative departure delay being approximately the same). Given that the FULL strategy appears to be mainly of theoretical interest in practice, and

Table 1. Preliminary computational results for Genova.

Plan	Strategy	Number of trains	Cum. external	Cum. arr.	Cum. dep.	Incr. arr.	Incr. dep.
AC	FULL	119	313	316	280	1%	−11%
BACKUP	FULL	109	280	282	249	1%	−11%
BASIC	FULL	119	313	315	277	1%	−11%
AC	PROP	119	313	337	301	8%	−4%
BACKUP	BACKUP	109	281	293	270	4%	−4%
BASIC	PROP	119	313	359	335	15%	7%

the PROP and BACKUP ones closer to practice, these results suggest that the definition of backup platforms may be fairly helpful.

The only significant computation times for the simulation are spent on the re-scheduler and amount to 1-15 seconds for each call to the PROP-BACKUP re-schedulers and a couple of minutes for the FULL re-scheduler. This indicates that, after further fine-tuning, the proposed method is suitable for real-time applications.

8 Conclusions

In this paper we proposed robust routing models and re-scheduling algorithms for the train routing problem. Further we designed a simulation framework to compare the robustness of the nominal plans and the effectiveness of the re-scheduling algorithms.

We carried out preliminary computational results on a realistic instance of the main Italian railway infrastructure manager. The results indicate that incorporating our robustness considerations in the train routing problem, together with appropriately chosen online re-scheduling algorithms, can indeed lead to better punctuality of the trains. Further, the computation times are suitable both for robust nominal planning (in early planning stages) and for online re-scheduling (in real-time operations).

In our future research we will fine-tune the proposed methods, extend our test-bed to other instances, and we will consider novel heuristic recovery algorithms.

References

1. Billionnet A.: *Using Integer Programming to Solve the Train Platforming Problem.* Transportation Science, 37 (2003) 213-222.
2. Caimi G., Burkolter D., Herrmann T.: *Finding Delay-Tolerant Train Routings through Stations.* Operations Research Proceedings 2004, Fleuren H., (2007), 136-143, Springer.
3. Caprara A., Galli L., Toth P.: *Solution to the Train Platforming Problem.* Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems 2007. `http://drops.dagstuhl.de/portals/ATMOS07/`

4. Caprara A., Galli L., Stiller S., Toth P.: *Recovery-Robust Train Platforming via Network buffering.* Proceedings of RailZurich 2009.
5. Carey M., Carville S.: *Scheduling and Platforming Trains at Busy Complex Stations.* Transportation Research A, 37 (2003) 195-224.
6. Carey M., Crawford I.: *Scheduling Trains on a Network of Busy Complex Stations.* Transportation Research B, 41(2) (2007) 159-178.
7. Chakroborty P., Vikram D.: *Optimum Assignment of Trains to Platforms under Partial Schedule Compliance.* Transportation Research B, 42(2) (2008) 169-184.
8. Cornelsen S., Di Stefano G.: *Track Assignment.* Journal of Discrete Algorithms, 5 (2007) 250-261.
9. De Luca Cardillo D., Mione N.: *k L-List T Colouring of Graphs.* European Journal of Operational Research, 106 (1999) 160-164.
10. Fuchsberger M.: *Solving the Train Scheduling Problem in a Main Station Area via a Resource Constrained Space-Time Integer Multicommodity Flow.* Master Thesis, Institute for Operations Research, ETH Zürich, Switzerland, (2007).
11. Galli L.: *Combinatorial and Robust Optimisation Models and Algorithms for Railway Applications*, PhD Thesis, DEIS, University of Bologna, Italy, (2009). `http://www.or.deis.unibo.it/`
12. Kroon L.G., Maróti G.: *Robust Train Routing.* Technical Report 0123, EU ARRIVAL project.
13. Zwaneveld P.J., Kroon L.G., Romeijn H.E., Salomon M., Dauzere-Peres S., van Hoesel C.P.M., Ambergen H.W.: *Routing Trains Through Railway Stations: Model Formulation and Algorithm*, Transportation Science, 30 (1996) 181-194.
14. Zwaneveld P.J., Kroon L.G., van Hoesel C.P.M.: *Routing Trains through a Railway Station based on a Node Packing Model.* European Journal of Operations Research, 128 (2001) 14-33.