Real Time Large Railway Network Re-Scheduling

Erik, Christian, Emma

October 2019

Contents

Abstract

In this paper, we describe our first steps to tackle the Re-Scheduling Problem for Real-Time Large Railway Networks by a combination of techniques from operations research with different heuristics from ML and domain-specific heuristics.

The Industry State of the Art manages to resolve re-scheduling conflicts fully automatically only at narrowly defined hubs. In our approach, we aim at combining the best of two worlds: the rigor of the OR formulation with condensed experience in the form of a hypothesized Oracle. The Oracle's prediction could either narrow down the solution space (hard constraints) or speed up the solution process by strong priorities (soft constraints).

We believe that the very nature of railway system allows for very strong heuristics which could allow for the problem to become tractable for large networks in real-time scenarios.

The goal of this paper is to report the decompositional problem formulation and our first steps in tackling the Oracle. We hope that this will draw the attention of both academic and industrial researchers to find other and better approaches and collaboration across Railway companies and from different research traditions.

1 Introduction

Here we describe the Re-Scheduling problem.

Switzerland has a dense railway network with both freight and passenger trains running on the same infrastructure. More than 1.2 million people use trains on a daily basis. In Railway Operations, the operational schedule has to be continually re-computed because of many smaller and larger delays that arise during operations. Not all of those can be absorbed by extra times in the schedule, and if the delay has an impact on other trains, decisions on re-ordering or re-routing trains have to be taken to derive a new feasible operational plan. The industry state of the art is that delay propagation is efficiently re-computed by online IT systems Conflicts, however, have to be most often resolved by humans by explicitly deciding on re-ordering or re-routing based on their experience. Because of the massive combinatorial complexity of these microscopic models, Operations Research models are currently only applied in very restricted, highly condensed areas for re-ordering decisions but do not consider routing alternatives. Research Approach

To tackle this problem, our approach is to combine Operations Research and Machine Learning to get the best of both worlds: a Graph Neural Network acts as an Oracle for Operations Research by learning a heuristic that predicts the "impact" of a delay; we hope the Oracle could predict which trains and which departures are or could be affected by the delay based on past decisions. This piece of information from the oracle then helps the (MILP or CP) solver to constrain the search space or at least drive its search more efficiently (driving

the branching process). We hope that Graph Neural Networks are a natural way to represent the graph structure of railway networks.

Our considerations are based on the Re-Scheduling Effects Model of Figure 1: the Operational Schedule is based on being informed about delays and the schedule communicated to customers. Interventions may be redefine the allowed speeds per location or per train, the available routes, the connections to be kept or dropped and trains to be added or dropped. Interventions may be directly based on events happening in reality or based on the current network status aggregated from past events. Events from reality are interpreted when lead to a difference to the Operational Schedule.

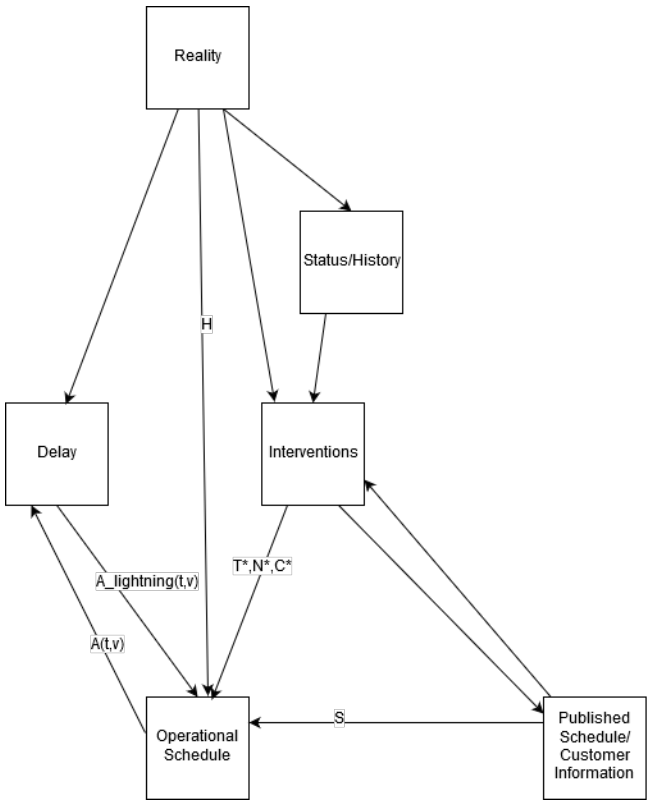

Figure 1: Re-Scheduling Effects Model. Arrows represent logical dependencies. The symbols on the arrows are those used in the formal problem description later on.

2 Related Work

TODO Related Work. How does it relate to [1]? TODO Other different approaches: Benders-like decomposition and loop, generate-and-verify-loop

3 Methodology

Here we describe the three hypotheses and the pipeline.

3.1 3 Hypotheses, one building on the previous

Hypothesis 1 We can compute good recourse actions, i.e., an adapted plan within a small fraction of the full solution time if all variables are fixed, except those related to services that are affected by the disruption implicitly or explicitly.

Hypothesis 2 Machine learning can predict services that are affected by disruptions implicitly or explicitly with respect to a fixed infrastructure and a fixed basic (published) schedule, without the intention of model training to generalize over infrastructures nor completely new timetable concepts.

Hypothesis 3 If hypothesis 2 is true, in addition, machine learning can predict the state of the system in the next time period after re-scheduling.

3.2 A General Framework

Our approach can be seen as a general framework with the following ingredients:

Detailed Model We have a model that reflects the detailed view but which cannot be applied because it would be intractable. A detailed model acts at the operational and could be produced in reality if no further delays or disruptions happen; it not only works at the more imprecise tactical level. Formally, the detailed model is a deterministic solver that produces an operational plan $\mathbf{y}^*(\mathbf{x})$ for a problem $\mathbf{x}$.

Oracle An Oracle predicts relevant parts of the solution such that the detailed exact model becomes tractable if we complement it by the informed guess by the Oracle. Formally, an Oracle $\mathcal{O}$ produces an enhanced input $\mathcal{O}(\mathbf{x})$ such that $\mathbf{y}^*(\mathcal{O}(\mathbf{x}))$ can be computed more efficiently.

Simulation Verifying an Oracle in the real world is often too expensive, either in financial terms or because of legal or other risk issues. Simulation allows us to train and verify the Oracle without having to run those risks. More formally, a simulation produces realistic inputs $\mathbf{x}$ (following the same distribution).

3.3 Pipeline for Verification of Hypothesis 1

Input generation Step Generate infrastructure Generate schedule Generate disturbance Full Schedule (w/o disturbance) Hypothesis Validation

How big is the speed up if we know Delta? Step Re-Scheduling full w/o restrictions Determine Delta w.r. to Full Schedule Re-Scheduling Delta

4 Formal Problem Definition

4.1 FLATland Formal Definition

Gridworld and Semantics. I'd like to formalize this. I think it would be worth to describe the FLATland semantics (synchronization, entering Grid, malfunctions, speed model). This will help us to generalize/abstract.

4.2 Scheduling Problem

Our formulation is based on [2], but we slightly modify it in order to make the equivalence of microscopic halting sections explicit in the mathematical formulation. We will need this explicit property in the definition the Re-Scheduling Problem.[1] The naming is based on [3].

A *Scheduling Problem* is a triple (N, T, C) consisting of

- a *Railway Network* $N = (V, E, R, m, a, b)$ where
  - *global route graph* (V, E), which is a directed graph; the edges are also called *route sections*
  - R is a set of resources
  - *minimum travel times* per route section $m : E \to \mathbb{N}$
  - *resource allocations* on route sections $a : R \to 2^E$
  - *blocking or release times* per resource $R \to \mathbb{N}$
- *trains* T of the form $T \ni t = (V^t, E^t, e^t, l^t, w^t)$ where
  - *route graph* (V^t, E^t) is an acyclic sub-graph of (V, E)
  - *earliest passing time* of train t at a vertex: $e^t : V^t \to \mathbb{N}$
  - *latest passing time* of train t at a vertex $l^t : V^t \to \mathbb{N}$
  - *waiting times* of train t on edge $w^t : E^t \to \mathbb{N}$
  - a partition $W^t = \{W_1^t, \ldots, W_n^t\} \subseteq 2^{E^t}$ of equivalent microscopic edges such that every path $p = (v_0, \ldots, v_k) \subseteq V^t$ for $(v_i, v_{i+1}) \in E^t$ for $i = 1, \ldots, n-1$ [2] such that $in(v_0) = 0$ and $out(v_n) = 0$ contains exactly one element of each partitioning element W_i, $i = 1, ..., n$ (i.e. there is exactly one edge $w(W_i^t) = (v_j, v_{j+1}) \in E^t$ for $W_i^t \in W$ and $v_j, v_{j+1} \in p$ and such that all $w(W_i^t)$ are different (i.e. w is injective)).
- *connections* $C \subseteq T \times E \times T \times E \times \mathbb{N}$ such that $(t_1, e_1, t_2, e_2, c) \in C \implies e_1 \in E^{t_1}, e_2 \in E^{t_2}$

[1] TODO: consistency of connections is not enforced - do we need it?

[2] We use the following notation: $(v_0, \ldots, v_n) \subseteq V^t$ denotes an ordered subset of elements of V^t. TODO: Should we formally introduced the set of paths in V^t, relating to E^t to simplify notation?

Note that (V, E) needs not be connected, i.e. each train may have its own (disjoint) route sub-graph, which is linked to other trains' route graph only through shared resources in R, and the union of of all train-specific route graphs needs not cover (V, E). Notice that the some consistency conditions of the real-world problem are not part of this formal definition, for instance we would have to ensure that every path can produce the commercial stops on different platform tracks.

A solution (P, A) of the Scheduling Problem consists of the

- the selected paths $P : T \to 2^V$ where $p(t) = (t_1, \ldots, t_n) \subseteq V^t$ (we use $\ell(p(t)) = n$ for the length of the path) is an ordered set satisfying

$$(v_i, v_{i+1}) \in E^t \text{ for } i \in \{1, \ldots, n-1\} \tag{1}$$

$$in(v_1) = 0 \text{ and } out(v_n) = 0 \tag{2}$$

  where $in : V \to \mathbb{N}$ and $out : V \to \mathbb{N}$ associate in and out degrees of vertices.

- partial allocation $A : T \times V \to \mathbb{N}$ where $A(t, v)$ is defined for $v \in P(t)$ satisfying

$$A(t, v) \geq e^t(v) \text{ for } v \in P(t) \tag{3}$$

$$A(t, v) \leq l^t(v) \text{ for } v \in P(t) \tag{4}$$

$$A(t, v_i) + m((v_i, v_{i+1}) + w^t((v_i, v_{i+1})) \leq A(t, v_{i+1}) \tag{5}$$

$$\begin{aligned} A(t_1, v') + b(r) \leq A(t_2, u) \oplus A(t_2, u') + b(r) \leq A(t_1, v) \\ \text{for } v, v' in \in P(t_1), u, u' \in P(t_2), (v, v'), (u, u') \in a(r) \end{aligned} \tag{6}$$

  (where $\oplus$ denotes exclusive or)

$$A(t_1, v) + c \leq A(t_2, u) \text{ for } (t_1, (v, v'), t_2, (u, u'), c) \in C \tag{7}$$

4.3 FLATland as Scheduling Problem

How are the route graphs derived from FLATland in general?

4.4 Commercial and Operational Schedule

The result of the above (offline) Scheduling Approach is a conflict-free (microscopic) solution. We therefore call it an *Operational Schedule*. If everything runs smoothly, it contains all the information required to run trains.

What we publish to customers needs not be as details and may not even be conflict-free in the microscopic sense. For instance, we might publish two trains to start at the same time and decide online which one goes first whichever is

ready first. Formally, given an Operational Schedule (P, A), a *Commercial or Published Schedule* captures the elements published to consumers[3]. Formally,

- $S(t, v)$ is the published schedule, partially defined for $v_{i+1} \in P(t) = (v_1, \ldots, v_n)$ where $w^t(v_i) > 0$ and satisfying $S(t, v) \leq A(t, v)$ and $S(t, v_j) \leq S(t, v_k)$ for $j < k$ and $v_j, v_k \in P(t)$

i.e. we allow for conflicts in the commercial schedule, but guarantee that we never depart earlier than published in S.

4.5 Re-Scheduling Problem

Given a

- Operational Schedule (P, A),
- Commercial Schedule S
- the current time $H \in \mathbb{N}$
- operational interventions in the form of a scheduling problem (N^*, T^*, C^*)
- an event $A^{\lightning}(t^{\lightning}, v^{\lightning}) \geq H$, $A^{\lightning}(t, v) \neq A^{(}t, v)$, $A(t, v) \geq H$

The interpretation of H is that we have received all information up to H, i.e. all times $A(v, t) \leq H$ are fixed, i.e. they are already included included in our current operational schedule (either because they worked according to our initial plan or because we have already updated our current operational schedule). We require $A^{\lightning}(t, v) \geq H$; the interpretation is that $A^{\lightning}(t^{t}\!\!\lightning, v^{\lightning})$ is a prognosis when train t will pass at v for $A(t, v) \geq H$. Examples:

- If $A^{\lightning}(t^{\lightning}, v^{\lightning}) = H$ and we know that we will be ready only later on at $A^{\lightning}(t^{\lightning}, v^{\lightning}) \neq A^{(}t^{\lightning}, v^{\lightning})$, $A(t, v) \geq H$.
- If $A(t^{\lightning}, v^{\lightning}) > H$, we might know that we will be ready earlier at $A^{\lightning}(t^{\lightning}, v^{\lightning}) < A(t^{\lightning}, v^{\lightning})$

A *Re-Scheduling Oracle* is a pair (F_1, F_2)

- $F_1 \subseteq T^*$ and $\mathcal{G}(t) = (\bar{V}^t, \bar{E}^t)$ is an acyclic connected sub-graph of (V, E) for $t \in F_1$
- $F_2 \subseteq T^* \times V$ where $(t, v) \in F_2 \implies A(t, v) \geq H, v \in P(t), t \notin F_1$.
- $T(F_2) \cap F_1 = \emptyset$ where $T(F_2) = \{t : (t, v) \in F_2)$

The situation is depicted in Figure 2: there are trains with no flexibility, trains with time flexibility and trains with routing and time flexibility. Notice that the route graphs $\mathcal{G}(t)$ may grow or shrink with respect to the initial Scheduling Problem, i.e. we may allow for route alternatives that were not allowed for planning. We even allow for trains and connections to be added or dropped.

[3]TODO: Should the connections be part of the Commercial Schedule as well?

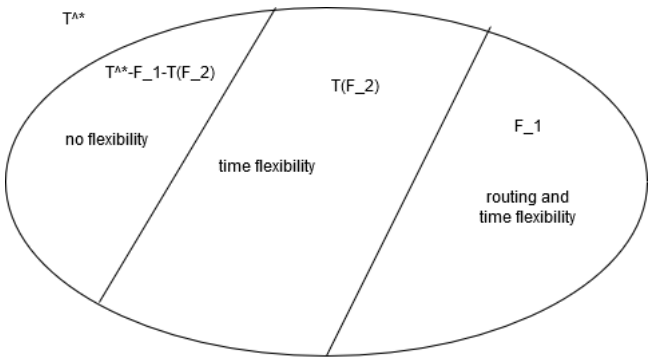

Figure 2: Degrees of flexibility for the Re-Scheduling-Problem.

A Solution $(\bar{P}, \bar{A})$ of the Re-Scheduling Problem is a solution to the Scheduling Problem (N^*, T^*, C^*), further satisfying

$$\bar{P}(t) = P(t) \text{ for } t \notin F_1, \tag{8}$$

$$\bar{A}(t, v) = A(t, v) \text{ for } t \notin F_1, t \notin T(F_2), \tag{9}$$

$$\bar{A}(t, v') \geq S(t, v) \text{ for all } S(t, v) \text{ and some } v' \in W(\_, v) \tag{10}$$

as well as equations (1), (2), (5), (6), (7) above and minimizing

$$\sum_{(t,v) \in \mathrm{dom}(S)} \bar{A}(t, v) - S(t, v). \tag{11}$$

The Re-Scheduling Oracle defines the degrees of freedom on two levels, allowing path alternatives for trains $t \in F_1$ and allowing for modified departure times for the departures in F_2, but we require trains to respect the Commercial Schedule, i.e. we do not allow for trains or stops to be dropped.

5 Results Hypothesis 1

5.1 Implementation Details

k shortest paths, constraints in the model above, best-case Oracle

Our setting for Re-Scheduling for step XX in [1]:

- $N^* = N$
- T = shortest path or all routes
- $C^* = C$
- F_1, F_2?

6 Results Hypothesis 2

6.1 DL approaches

6.1.1 Graph-based Neighborhood Search (GNN)

Approach Christian B. + Mayra

6.1.2 Probabilistic Schedule (GNN)

Erik

6.1.3 Reinforcement Learning

Erik with Jakab/Levent?

6.2 Heuristics Oracle

6.2.1 Deadlock Avoidance

Extending Adrian's Deadlock-Avoidance Algorithm? Christian E. and Adrian?

6.2.2 Job Insertion

Blocking Job-Shop-Scheduling Fribourg (Reinhard Bürgy)? Christian E. with Reinhard?

6.3 Exploiting the Oracle in OR

Christian E.

7 Conclusion

What have we achieved?

8 Future Work

Describe the Research Plan and the Challenge for other Researcher.

To Discuss

- definition of operational schedule should be independent of the scheduling problem based on time windows, definition is not very readable
- - not use t for train but for time? Confusing for physicists....
- use real numbers for event times

- which cost function (classes) do we want to consider?
- use different mathematical problem formulation (alternative/disjunctive graph job shop scheduling,....?)
- Further Research: not only consider isolated events?
- Further Research: drop trains, drop halts, ...
- Further Research distributions or time windows as prognosis not only fixed event time

References

[1] Eric Larsen, Sébastien Lachapelle, Yoshua Bengio, Emma Frejinger, Simon Lacoste-Julien, and Andrea Lodi. Predicting solution summaries to integer linear programs under imperfect information with machine learning. *CoRR*, abs/1807.11876, 2018.

[2] Dirk Abels, Julian Jordi, Max Ostrowski, Torsten Schaub, Ambra Toletti, and Philipp Wanko. Train scheduling with hybrid ASP. In *LPNMR*, volume 11481 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2019.

[3] starter-kit repo for the sbb train schedule optimisation challenge on crowdai: Input Data Model. `https://github.com/crowdAI/train-schedule-optimisation-challenge-starter-kit/blob/master/documentation/input_data_model.md`. Accessed: 2019-11-04.