MILP approaches to practical real-time train scheduling: the Iron Ore Line case

Lukas Bach
SINTEF
Oslo, Norway
lukas.bach@sintef.no

Carlo Mannino
SINTEF
Oslo, Norway
carlo.mannino@sintef.no

Giorgio Sartor
SINTEF
Oslo, Norway
giorgio.sartor@sintef.no

ABSTRACT

Real-time train scheduling is a complex network optimization problem, which is receiving increased attention from scientists and practitioners. Despite a vast literature on optimization algorithms for train dispatching, there are very few examples of real-life implementations of such algorithms. Indeed, the transition from theory to practice poses several critical issues, and many simplifying assumptions must be dropped. MILP models become more involved and hard to solve in the short time available. Here we describe how we successfully tackled these issues for dispatching trains on a railway in the north of Norway and Sweden.

1 INTRODUCTION

Railway infrastructure is increasingly congested: passenger traffic is expected to grow by 3.2% yearly for the next 8 years, while freight traffic by 1.4% ([13]). Increasing pressure results in poorer punctuality. In principle, one could augment capacity by building more infrastructure, but this requires large investments and the benefits will only be available after some years. A quicker and cheaper way to increase capacity is to improve traffic management by network optimization. Several recent studies have shown improvements in the punctuality ranging from 10% to 100% [3, 6, 9, 10, 12]. In these and all other papers presented in a large literature (for recent reviews, see [4, 8]), the train scheduling problem is represented by means of *event graphs*. The seminal example is probably Balas' disjunctive graph introduced in [1] (where each node represents the starting of an operation), later extended to cope with blocking, no-wait job-shop scheduling problems by Mascis and Pacciarelli [11].

Despite this huge body of academic studies and successful stories, there have been only a few implementations of real-time train scheduling algorithms in real life [2, 9, 10]. Things are rapidly changing now, thanks to an increased interest by infrastructure managers worldwide in automatic train traffic control systems capable of maximizing punctuality or average velocity[1]. In this paper, we describe one such implementation, focusing on the modelling and algorithmic challenges we had to tackle when moving from theory to practice. First, standard simplifying assumptions must be discarded in order to produce solutions which are practically viable. Next, new solution approaches must be developed and implemented in order keep the computation time of the optimal solution in the range of few seconds. Indeed, Fischetti and Monaci [5] showed that state-of-the-art solvers are already unable to tackle rather small instances of the MILP models derived from the event graphs of these type of scheduling problems.

The implementation we discuss in this paper is applied to a critical part of the railway network that runs from Sweden to the coast of Norway, also known as the "Iron Ore Line". This single track line, well within the arctic circle, was originally built to transport iron ore from northern Sweden to the ice free waters of Narvik in northern Norway. The line is also used by a few passenger trains per day. In recent years, the increased number of iron ore trains as well as other types of freight trains has challenged the capacity of the line. An optimized dispatching could help ensuring that the physical capacity is used to its full extent.

One peculiar challenge of this piece of railway comes from its incline. This affects the speed of the heavy trains and may also affect their ability to stop in some of the stations. For example, fully loaded freight trains travelling from the iron ore mines in Sweden to Norway have constraints regarding where they are allowed to stop, while lighter freight trains travelling towards Sweden are allowed higher speed and flexibility. Moreover, some freight trains are also too long for some of the side tracks in certain small stations. Passenger trains may have limitations too. In fact, most of the stations do not have passenger platforms in all their internal tracks, constraining the number of passenger trains that are able to meet in the station. Instead, for all type of trains, another important aspect is the variability of their travel times. Indeed, moving from a stopped condition requires some time to accelerate; similarly, stopping a train requires a deceleration and thus extended running times.

All these practical constraints are usually ignored in theoretical works but they are crucial in real-world applications, requiring more refined models. In this work, we mainly focus on two aspects: a) being able to define a more diverse set of constraints within each station; b) model travel times of each train based on its stopping pattern.

Our starting point is the recent Benders' like decomposition approach to train rescheduling presented in [7, 9], extended to cope with all new physical and logical constraints. In this paper, we discuss the new features and the decomposition approach for this MILP problem. Furthermore we describe the actual implementation which has been tested by dispatchers on the iron-ore line.

2 A MILP FORMULATION

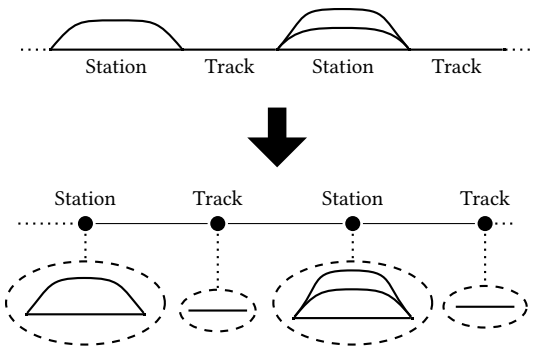
We start our description by considering a slightly simplified version of our problem, where the travel times of trains are fixed and do not depend on whether they stop.

The Iron Ore Line consists of a sequence of small stations and single-tracks. We follow here the micro-macro decomposition approach proposed in [7]. The macro problem is associated with the railway line, considered as a sequence of capacitated resources,

[1] In [2], a large North-American railway company claims that a 1% increase in average velocity of their freight trains brings to the company $200 million savings.

Figure 1: Iron Ore Line

Figure 2: Line decomposition



alternating stations and tracks (see Figure 2). Observe that at this macro level, we avoid the detailed description of the movements (i.e. routing and scheduling) of each train in every station which is instead represented by the expected total time spent in the station. The micro problem is associated with the routing and scheduling of trains within each station and track, according to the arrival and departure times established by the macro problem. The decomposition allows us to treat the constraints generated in the micro level (e.g., track assignment, capacity) independently of each other. In Section 4 we will see how this decomposition can be exploited to solve the MILP model in a master-slave fashion. Instead, in this section we focus on describing how to formulate the constraints arising both at the macro and micro level, describing train movements.

For the line (or macro) problem, each train $a \in A$ is assigned a route, i.e., an ordered sequence $n_a^{r_1}, n_a^{r_2}, \ldots, n_a^{r_q}$ of *route nodes*, where $r_i \in R, i = 1, \ldots, q$ is a line resource, either a track or a station[2], and r_1, r_q are the origin and destination station, respectively. In this aggregation scheme, between each pair of nodes that represent two adjacent stations, there is always a track node.

Let N be the set of all route nodes for all trains in A, $N^O \subset N$ be the set of all nodes associated with origin stations, and $N^D \subset N$ the set of all nodes associated with destinations. We associate a scheduling variable $t_a^r \in \mathbb{R}$ with each route node $n_a^r \in N$, representing the time train a enters the resource r. There is also a fictitious variable $t^o \in \mathbb{R}$, which serves as a reference time for all trains (typically, but not necessarily, we have $t^o = 0$). Thus, we have

$$t_a^r - t^o \geq \Gamma_a, \quad n_a^r \in N^O, \tag{1}$$

where Γ_a is the earliest time train a can enter the network. Now let $n_a^r, n_a^{r+1} \in N$ be two consecutive route nodes in a particular train route. Note that the time a train exits a resource is precisely the time the train enters the subsequent resource in its route. Therefore, the following constraints hold:

$$t_a^{r+1} - t_a^r \geq \Lambda_a^r, \quad n_a^r \in N \setminus N^D, \tag{2}$$

where Λ_a^r is the minimum time it takes train a to traverse resource r. Moreover, for the destination nodes we have:

$$t_a^{\text{out}} - t_a^r \geq \Lambda_a^r, \quad n_a^r \in N^D, \tag{3}$$

where the fictitious $t_a^{\text{out}}, a \in A$, represents the time train a "leaves" the railway network, i.e., it concludes its journey at the arrival station. Constraints (1), (2), and (3) are usually called *precedence constraints*, and they model the *free running* of a train, i.e., the minimum time required by a train to travel along its route without obstacles from other trains. Incidentally, even if we will not make explicit use of the underlying event graph, it is worth mentioning here that this is built by associating a node with every time variable and a directed edge with every constraint (1), (2), and (3).

In general, one has to consider the interactions between trains travelling in the same network. Observe that, for a pair of distinct trains a, b traversing a resource r, exactly one of the following three conditions must occur:

(1) train a and b meet in resource r
(2) train a traverses resource r before train b
(3) train b traverses resource r before train a

Consider now a set of distinct trains $A(r) \subseteq A$ traversing a resource r. For each ordered pair of distinct trains $(a, b) \in A(r) \times A(r)$, we define y_{ab}^r to be equal to 1 if a exits r before b enters, and 0 otherwise. Furthermore, for each pair of trains $\{a, b\} \subseteq A(r)$, we introduce the binary variable x_{ab}^r, which is 1 if and only if a and b are simultaneously (i.e., they *meet*) in resource r. Then, we have that

$$y_{ba}^r + y_{ab}^r + x_{ab}^r = 1, \quad \{a, b\} \subseteq A(r), r \in R. \tag{4}$$

Accordingly, for every $\{a, b\} \subseteq A(r), r \in R$, the schedule t will satisfy a family of (indicator) *disjunctive constraints* as follows[3]:

$$\begin{aligned}
&(i) \quad y_{ab}^r = 1 \implies t_b^r - t_a^{r+1} \geq 0, \\
&(ii) \quad y_{ba}^r = 1 \implies t_a^r - t_b^{r+1} \geq 0, \\
&(iii) \quad x_{ab}^r = 1 \implies \begin{cases} t_b^{r+1} - t_a^r \geq 0 \\ t_a^{r+1} - t_b^r \geq 0 \end{cases}, \\
&\qquad y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0, 1\}.
\end{aligned} \tag{5}$$

Indeed, $y_{ab}^r = 1$ implies that a exits r before b enters r and, similarly, $y_{ba}^r = 1$ implies that b exits r before a enters. On the other hand, when $x_{ab}^r = 1$, then both a and b exit the sector r after the other train enters it (i.e., they meet in r). Exploiting the

[2] Other decomposition schemes are possible, for instance by collapsing entire railway region in a single node of our master line problem.

[3] Constraints (5) are associated with special entities of the event graph called *disjunctive (alternative) edges*, see for instance [11].

big-M trick, the family of disjunctive constraints in (5) can be easily linearized as follows:

$$\begin{aligned}
&(i) && t_b^r - t_a^{r+1} \geq -M(1 - y_{ab}^r),\\
&(ii) && t_a^r - t_b^{r+1} \geq -M(1 - y_{ba}^r),\\
&(iii) && t_b^{r+1} - t_a^r \geq -M(1 - x_{ab}^r),\\
&(iv) && t_a^{r+1} - t_b^r \geq -M(1 - x_{ab}^r),\\
& && y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0,1\},
\end{aligned} \tag{6}$$

A final set of constraints in the macro program will be used to represent the infeasibility of the micro problems, which in turn is associated to the resources in which the railway is decomposed.

Now, let t^* be a schedule that satisfies constraints (1), (2), (3), and (6), and suppose t^* minimizes a given objective function $c(t)$.

If the timetable t^* is feasible for every micro problem (i.e., for every station and every track section between successive stations), then it is feasible and optimal also for the overall problem. Otherwise, at least for one station or one track section, the time schedule decided by the macro problem cannot be attained. There may be several reasons for such infeasibility. Here we will describe the case where feasibility depends only on the set of trains simultaneously in the resource. For example, two passenger trains are not able to meet in a station where there are two internal tracks but only one passenger platform, but two freight trains may meet. On a single track no two trains can meet. Two short trains may pass each other in a siding, but not two long trains. Etc.

So, let $r \in R$ be a set of trains $Q \subseteq A$ *minimally infeasible for* r, if the trains in Q cannot meet simultaneously in r, but all proper subsets of trains in Q can meet. We define the set of $\mathcal{A}(r) \subset 2^A$ as the family of minimally infeasible set of trains for r. Clearly, for any $Q \in \mathcal{A}(r)$, at least two trains[4] in Q cannot meet in r. Note that if, according to a solution (t^*, x^*, y^*), all trains in Q meet in r, then we have $\sum_{\{a,b\}\subseteq Q} x^{*r}_{ab} = \binom{|Q|}{2}$ (namely the number of pairwise meetings in r of trains in Q is precisely $\binom{|Q|}{2}$). To prevent this to happen when the set Q is minimally infeasible, we can thus write the constraint:

$$\sum_{\{a,b\}\subseteq Q} x_{ab}^r \leq \binom{|Q|}{2} - 1, \quad Q \in \mathcal{A}(r), r \in R. \tag{7}$$

In conclusion, a complete MILP formulation can be obtained by considering as objective function $c(t)$ the sum of the arrival times at destination of the trains, subject to constraints (1), (2), and (3) for all routes, and constraints (4), (6), and (7) for all resources $r \in R$ and all the minimally infeasible sets $Q \in \mathcal{A}(r)$ of trains.

[4]Observe that, by Helly's property, if every pair of trains in Q meet in r, then there exist a point in time where all trains in Q are simultaneously in r.

The full model can be written as follow:

$$\begin{aligned}
&\min\ c(t)\\
&\text{subject to:}\\
&t_a^r - t^o \geq \Gamma_a, && n_a^r \in N^O\\
&t_a^{r+1} - t_a^r \geq \Lambda_a^r, && n_a^r \in N \setminus N^D\\
&t_a^{\text{out}} - t_a^r \geq \Lambda_a^r, && n_a^r \in N^D\\
&y_{ba}^r + y_{ab}^r + x_{ab}^r = 1, && \{a,b\} \subseteq A(r), r \in R\\
&t_b^r - t_a^{r+1} \geq -M(1 - y_{ab}^r), && \{a,b\} \subseteq A(r), r \in R\\
&t_a^r - t_b^{r+1} \geq -M(1 - y_{ba}^r), && \{a,b\} \subseteq A(r), r \in R\\
&t_b^{r+1} - t_a^r \geq -M(1 - x_{ab}^r), && \{a,b\} \subseteq A(r), r \in R\\
&t_a^{r+1} - t_b^r \geq -M(1 - x_{ab}^r), && \{a,b\} \subseteq A(r), r \in R\\
&\sum_{\{a,b\}\subseteq Q} x_{ab}^r \leq \binom{|Q|}{2} - 1, && Q \in \mathcal{A}(r), r \in R\\
&y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0,1\}, && \{a,b\} \subseteq A(r), r \in R\\
&t_a^r \in \mathbb{R}, && n_a^r \in N\\
&t_a^{\text{out}} \in \mathbb{R}, && a \in A\\
&t^o \in \mathbb{R}.
\end{aligned} \tag{8}$$

As mentioned above, the cost function $c(t)$ in (8) usually consists of the sum of the weighted arrival time at destination of all trains, that is $c(t) = \sum_{a\in A} w_a t_a^{\text{out}}$, where w_a is the weight of train a. However, this can be generalized to more complex functions. For example, an objective function commonly used in the railway industry is a piece-wise linear one, where the delay of a train is taken into account only if it is greater than few minutes[5].

3 AN EXTENDED MILP FORMULATION

As discussed in the previous section, the model in (8) assumes that travel times in tracks are constant and do not depend on the fact the train has stopped at the previous station or that it is going to stop in the next station. Similarly with travel times in stations. While this assumption is usually tolerated for passenger trains, it cannot be applied to heavy freight trains. In fact, they usually need a very long time to reach the nominal speed after they stopped, and to reach a full stop when travelling at nominal speed. Based on theese observations, we identified four different running times for each track and two different running times for each station.

We define by $R^T, R^S \subset R$ as the sets of resources that represent tracks and stations, respectively. Now, for each track $r \in R^T$ and for each train $a \in A$ we have:

- Λ_a^r: the minimum running time if a does not stop at the previous or next station;
- $\Lambda_a^r + \bar{\Lambda}_a^r$: the minimum running time if a stops at the next station but not at the previous one;
- $\Lambda_a^r + \underline{\Lambda}_a^r$: the minimum running time if a stops at the previous station but not at the next one;
- $\Lambda_a^r + \bar{\Lambda}_a^r + \underline{\Lambda}_a^r$: the minimum running time if a stops both at the previous and next station.

Instead, for each station $r \in R^S$ and for each train $a \in A$ we have:

- Λ_a^r: the minimum running time if a does not stop in this station;
- $\Lambda_a^r + \Delta_a^r$: the minimum running time if a stops in this station.

In order to include this flexible running times into model (8), we introduce binary variables $z_a^r, a \in A, r \in R^S$ that are equal to

[5]Note that the delay of a train $a \in A$ can be computed by subtracting the originally scheduled arrival time at destination from t_a^{out}.

1 if train a stops in station r, 0 otherwise. In other words, we introduce this additional (indicator) constraint:

$$z_a^r = 0 \qquad \Longrightarrow \qquad t_a^{r+1} - t_a^r = \Lambda_a^r. \tag{9}$$

Similarly to what done in (6), we can linearize this constraint by using the big-M trick:

$$t_a^{r+1} - t_a^r \leq \Lambda_a^r + M z_a^r. \tag{10}$$

Indeed, when z_a^r is equal to 0, then this constraint together with (2) imply that the travel time of train a in r is exactly Λ_a^r, which means that the train did not stop and traversed the station at its nominal speed.

With these "stopping" binary variables at hand, we can now define the constraints that model the travel times in stations and tracks more accurately.

For each station $r \in R^S$, and each train $a \in A$, we introduce the following set of constraints:

$$t_a^{r+1} - t_a^r \geq \Lambda_a^r + \Delta_a^r z_a^r. \tag{11}$$

Similarly, for each track $r \in R^T$, and each train $a \in A$, we have:

$$t_a^{r+1} - t_a^r \geq \Lambda_a^r + \bar{\Delta}_a^r z_a^{r-1} + \underline{\Delta}_a^r z_a^{r+1}, \tag{12}$$

where z_a^{r-1}, z_a^{r+1} represent the stopping variables at the previous and next stations, respectively.

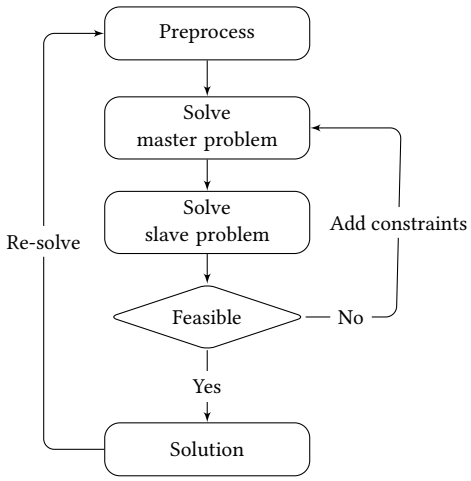
We can now substitute constraints (2) with constraints (10), (11), and (12) to obtain the final MILP model:

$$\begin{aligned}
&\min\ c(t)\\
&\text{subject to:}\\
&t_a^r - t^o \geq \Gamma_a, && n_a^r \in N^O\\
&t_a^{r+1} - t_a^r \geq \Lambda_a^r + \Delta_a^r z_a^r, && n_a^r \in N \setminus N^D, r \in R^S\\
&t_a^{r+1} - t_a^r \geq \Lambda_a^r + \bar{\Delta}_a^r z_a^{r-1} + \underline{\Delta}_a^r z_a^{r+1}, && n_a^r \in N \setminus N^D, r \in R^T\\
&t_a^{r+1} - t_a^r \leq \Lambda_a^r + M z_a^r, && n_a^r \in N \setminus N^D, r \in R^S\\
&t_a^{\text{out}} - t_a^r \geq \Lambda_a^r, && n_a^r \in N^D\\
&y_{ba}^r + y_{ab}^r + x_{ab}^r = 1, && \{a,b\} \subseteq A(r), r \in R\\
&t_b^r - t_a^{r+1} \geq -M(1 - y_{ab}^r), && \{a,b\} \subseteq A(r), r \in R\\
&t_a^r - t_b^{r+1} \geq -M(1 - y_{ba}^r), && \{a,b\} \subseteq A(r), r \in R\\
&t_b^{r+1} - t_a^r \geq -M(1 - x_{ab}^r), && \{a,b\} \subseteq A(r), r \in R\\
&t_a^{r+1} - t_b^r \geq -M(1 - x_{ab}^r), && \{a,b\} \subseteq A(r), r \in R\\
&\sum_{\{a,b\} \subseteq Q} x_{ab}^r \leq \tbinom{|Q|}{2} - 1, && Q \in \mathcal{A}(r), r \in R\\
&y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0,1\}, && \{a,b\} \subseteq A(r), r \in R\\
&z_a^r \in \{0,1\}, && n_a^r \in N, r \in R^S\\
&t_a^r \in \mathbb{R}, && n_a^r \in N\\
&t_a^{\text{out}} \in \mathbb{R}, && a \in A\\
&t^o \in \mathbb{R}.
\end{aligned} \tag{13}$$

4 SOLUTION APPROACH

In our real-life implementation of train rescheduling, Problem (13) is solved iteratively every 10 seconds. Each time, the current status of the trains (i.e. position, speed, etc.) is gathered from the field, along with the current status of the rail network. The associated initial event graph is built. This is the pre-processing phase. Then, the MILP associated to the current instance is solved. The solution method is based on the decomposition in the macro problem and the micro problems described in Section 2 and shown in Figure 2. Indeed, this can be seen a master-slave approach, as described in [7]. If the master (or macro) problem is infeasible, then there is no solution to the scheduling problem (and we are in a *deadlock* situation). Otherwise, it produces an optimal *tentative* schedule t^*. Recall that the schedule variables in the macro problem are associated with the times each train enters a macro railway resource, in our case a station or a track between two stations in the line. Thus, the schedule t^* may be interpreted as a tentative (*disposition*) timetable. Next, this timetable is used by all micro (or slave) problems in order to solve the routing/scheduling within each station or track (even though for tracks this is trivial). If they are all feasible, then we have found the optimal solution. Otherwise, we generate the constraints that invalidate the current schedule in at least on piece of the railway, forcing the master problem to find a new tentative schedule.

Figure 3: Solution Algorithm



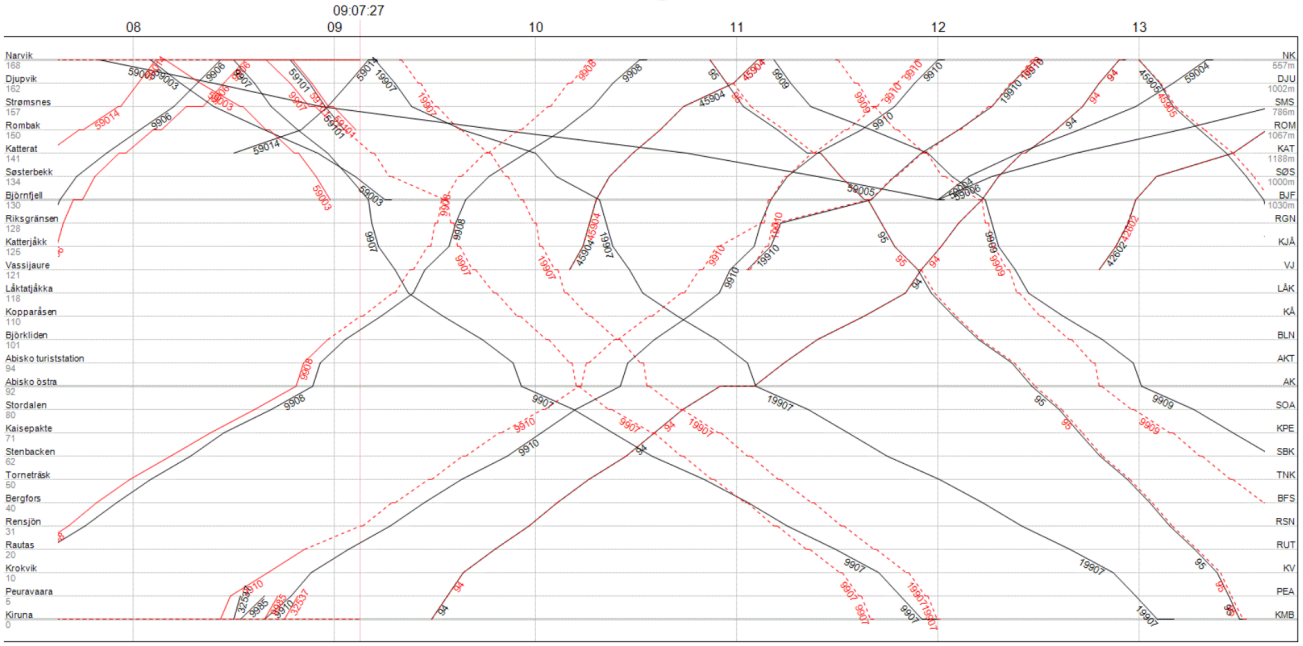
In Figure 3 we give a schematic representation of the overall algorithm. First, the real-time data are pre-processed and the event graph is updated (pre-processing). Second, the MILP Master Problem is solved considering only a subset (initially empty) of constraints (7): this MILP is called *restricted master*. The current master solution is then checked for feasibility by solving the slave problems. Namely, we check if any constraint (7) not included in the current restricted master is violated by the current solution - we call such violation a *conflict*. Observe that, while checking conflicts on tracks is in general a simple exercise, a similar check for stations can be indeed hard (some polynomial cases of practical interest are discussed in [7]). If this is the case, the violated constraints are added to the master problem and the process is iterated until no violated constraints exists.

Note that this delayed row-generation approach usually generates models that are much smaller than the ones generated by the full MILP formulation (see [7]). This helps to drastically reduce the computation time.

5 RESULTS & CONCLUSIONS

Implementing an algorithm of this sort in real-life poses, as described earlier, some additional challenges both with respect to removing theoretical assumptions and adapting formulations to problem specific peculiarities. Another major challenge is to interface with real-time systems in order to get the correct data in real-time. Collaboration with the dispatchers is very important as they have the final word on whether the decisions suggested by the algorithm is accepted or not. Here eliciting why they make their decisions should not be underestimated. The interaction

Figure 4: Train graph



with the dispatchers does also make it difficult to compare the algorithm to the current as we do simply not know what would have happened had the dispatchers accepted all dispatching suggestions.

The algorithm has been implemented into a user interface where we are able to show the dispatchers a classical train graph representing the line, see Figure 4. On the x-axis we have time, both past and present separated by a red line. The stations along the line are placed on the y-axis. Each train is shown as a line in the graph with its train number, the black line is the schedule, the full red line represent the past real-time data. Where the dashed red line is the future dispatching suggestions. This train graph has been tested over a period by the dispatchers in Narvik operational constrol center in real-time.

When running in a real-time setting the input data in the algorithm is updated every 10 seconds before it is executed again. Testing on real-time data over an extended period the approach presented in this paper has been able to provide solutions within (the wanted) 2 seconds. The planning horizon covers the following 2 hours, and the solution returned must be conflict free. The computing speed is extremely important as solutions which are constantly updated on the status of the trains and of the railway must be presented to dispatchers. Hence, with longer solution times the dispatching suggestions might already be out of sync with the real-time data.

The implementation was funded by the Norwegian National Research Council, and the system was indeed operative only on the Norwegian side of the line, supporting the Norwegian dispatchers sitting at Narvik. However, for the Swedish part of the line, there is a second control center located in Boden (not far from the Gulf of Bothnia where the line ends). It is worth noticing here that the limited coordination between the two brains of the line generates various problems. The dispatchers at Narvik may become aware of scheduling decisions taken at Boden only when the trains are approaching the border and in any case they cannot affect such decisions (if not occasionally through some laborious negotiations on the phones). It should be apparent that having a single optimization tool on both sides of the border would significantly increase the coordination, the quality of the overall solutions and the awareness of both teams of dispatchers.

REFERENCES

[1] Egon Balas. 1969. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations research* 17, 6 (1969), 941–957.

[2] Srinivas Bollapragada, Randall Markley, Heath Morgan, Erdem Telatar, Scott Wills, Mason Samuels, Jerod Bieringer, Marc Garbiras, Giampaolo Orrigo, Fred Ehlers, et al. 2018. A Novel Movement Planner System for Dispatching Trains. *Interfaces* 48, 1 (2018), 57–69.

[3] Quentin Cappart and Pierre Schaus. 2017. Rescheduling railway traffic on real time situations using time-interval variables. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, June 5–8, 2017, Padua, Italy*. Springer, Cham, 312–327.

[4] Francesco Corman and Lingyun Meng. 2015. A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems* 16, 3 (2015), 1274–1284.

[5] Matteo Fischetti and Michele Monaci. 2017. Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research* 263, 1 (2017), 258–264.

[6] Pavle Kecman, Francesco Corman, Andrea D'Ariano, and Rob MP Goverde. 2013. Rescheduling models for railway traffic management in large-scale networks. *Public Transport* 5, 1-2 (2013), 95–123.

[7] Leonardo Lamorgese and Carlo Mannino. 2015. An exact decomposition approach for the real-time train dispatching problem. *Operations Research* 63, 1 (2015), 48–64.

[8] Leonardo Lamorgese, Carlo Mannino, Dario Pacciarelli, and Johanna Törnquist Krasemann. 2018. Train Dispatching. In *Handbook of Optimization in the Railway Industry*. Springer, Cham, 265–283.

[9] Leonardo Lamorgese, Carlo Mannino, and Mauro Piacentini. 2016. Optimal train dispatching by Benders'-like reformulation. *Transportation Science* 50, 3 (2016), 910–925.

[10] Carlo Mannino and Alessandro Mascis. 2009. Optimal real-time traffic control in metro stations. *Operations Research* 57, 4 (2009), 1026–1039.

[11] Alessandro Mascis and Dario Pacciarelli. 2002. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* 143, 3 (2002), 498–517.

[12] Paola Pellegrini, Grégory Marlière, and Joaquin Rodriguez. 2016. A detailed analysis of the actual impact of real-time railway traffic management optimization. *Journal of Rail Transport Planning & Management* 6, 1 (2016), 13–31.

[13] SCI-Verkher. 2017. Rail transport markets - global market trends 2016-2025.