SINTEF

SINTEF A26384 - Unrestricted

Report

Prioritization of Trains: mathematical model

Author(s)
Leonardo Lamorgese, Arnt-Gunnar Lium,
Carlo Mannino

Foto: Jernbaneverket/Njål Svingheim

SINTEF Technology and Society
Applied Economics
10th October 2014

SINTEF Teknologi og samfunn
SINTEF Technology and Society

Address:
Postboks 4760 Sluppen
7465 Trondheim
NORWAY

Telephone: +47 73590300

ts@sintef.no
www.sintef.no
Enterprise Number:
NO 948007029 MVA

KEYWORDS:
Train dispatching, optimization, mathematical programming, rail

Report

Prioritization of Trains: mathematical model

VERSION
1.0

DATE
10th October 2014

AUTHOR(S)
Leonardo Lamorgese, Arnt-Gunnar Lium, Carlo Mannino

CLIENT(S)
Jernbaneverket

CLIENT'S REFERENCE
Tone Norløff

PROJECT
102001009 TS AØ Priorization of trains

NUMBER OF PAGES AND ATTACHMENTS
16

ABSTRACT
We present a mathematical optimization model suitable for providing decision support on train dispatching on small and medium sized regional lines.

An implementation of the model has undergone extensive real life testing at the Stavanger train dispatching central on the Stavanger - Moi line from February to October 2014. During this test period the model has been solved about 40 000 times, typically finding the optimal solution within two-four seconds.

PREPARED BY
Arnt-Gunnar Lium

SIGNATURE
Arnt-Gunnar Lium

CHECKED BY
Thor Bjørkvoll

SIGNATURE
Thor Bjørkvoll

APPROVED BY
Frode Rømo

SIGNATURE
Frode Rømo

REPORT NUMBER
SINTEF A26384

ISBN
9788214057027

CLASSIFICATION
Unrestricted

CLASSIFICATION THIS PAGE
Unrestricted

SINTEF

Contents

SINTEF

1 Background

This document provides a description of a mathematical optimization model able to solve the train dispatching problem for the Stavanger-Moi line to optimality in seconds. The model that was implemented cover, to the best of our knowledge, all relevant issues for dispatching on the Stavanger - Moi line. The model contains most of the required features for decision support train dispatchers on small and medium sized regional lines.
The model was developed by SINTEF as part of the research project Prioritization of trains and has been tested/used in actual operations by the dispatchers at Stavanger train dispatching central in the period February to October 2014. We thank them for their many good suggestions helping to improve the model.
The project was supported by The Research Council of Norway, Jernbaneverket, NSB Persontog, Flytoget AS, CargoNet AS and SINTEF. We are most greatful for their support.

2 Problem description

A railway network is generally a complex, interconnected system with different types of infrastructure which can be represented with growing detail. Moving from the macroscopic to the microscopic level inevitably increases the number and complexity of resources to be taken in account, turning even simple representation into a non trivial task. Although most details of the microscopic level are relevant in a practical implementation, they are omitted in this description for sake of clarity.
In this simplified picture, we start by modeling the *Railway Network* as a set S of stations and a set C of connections between pairs of stations. In the general case, adjacent stations may be connected by an arbitrary number of parallel tracks. In this report, we will consider at most two tracks (*double track*). We now introduce definitions for the elements of a railway network on the macroscopic and (in part) microscopic infrastructural level.

Stations. A station is represented for our purpose by a directed graph, with nodes associated with tracks where the trains can stop to execute some operations (*stopping points*) and directed edges associated with *interlocking-routes*, i.e. the tracks connecting stopping points. Special stopping points are the *entry/exit* points, where trains enter or exit the station, and the *platforms*, where trains may embark and alight passengers, or simply wait before departure. Each train running through a station s is associated with a directed path in the station graph. In the common case of passenger trains, such path contains a *platform* node. Trains running through a station s without a scheduled stop (ex. freight trains) may or may not have platforms in their path. Also, a path through a station will always contain an entrance (exit) node, unless the train originates (or terminates) in the station.

Connections. One or more tracks which connect two stations of the line are a *connection*. Tracks may be partitioned into *sections* and, generally, trains running on a track in the same direction must be separated by a minimum number of sections. This model allows for any number of *fixed* block sections. Also, we assume that rolling stock running on a double-track connection may use each track exclusively in one direction (*uni-directional traffic*), unless in the case of failure of one of the two tracks (*interruption*). Should this occur, trains running in opposite direction will alternate on the available track, as for the case of single-track connections.

Trains. Let T be the set of controlled trains. Controlled trains are trains for which, at time zero (the observed time), information is available and re-scheduling and re-routing decisions can, to some extent, be taken. Note that many of such trains may still have to enter the controlled line. Moreover, Train Dispatching is a real-time activity, which means that a controlled train can be anywhere in the network when the solution process is started, or can be simply expected to originate somewhere at a given time in the future.

Train Routes. A train route corresponds to the sequence of tracks, interlocking routes and stopping points encountered by a train in its run through the railway network. At a macroscopic level, the route of a train $i \in T$ is an alternating (ordered) sequence of stations and connections. At a given time, we denote by $S(i)$ the set of stations on the route of i. There is a natural ordering $(S(i), \succ_i)$, where for any distinct $r, s \in S(i)$ we let $s \succ_i r$ if s is encountered after r on the route of train i. We then define the set $C(i)$ of connections in the route of train i, i.e. $C(i)$ contains a connection for each pair of consecutive stations in $S(i)$, plus possibly the initial connection leading to the first station (if the current position of i is on a track). Also the connections in $C(i)$ are naturally ordered and for $c, d \in C(i)$ we write $d \succ_i c$ if d is encountered after c on the route of train i. Also, if the train is currently outside the controlled region we add to $C(i)$ a fictitious connection to represent it, which is the first one in the ordering of $C(i)$. Similarly, if the train will end outside the controlled region we add a second fictitious connection which will be the last one in the ordering of $C(i)$.
Let us consider now two distinct trains i and j and let $S(i,j) = S(i) \cap S(j)$ be the set of stations common to the routes of i and j. We assume that $S(i,j)$ is a set of consecutive stations both for i and for j, and we order the stations in $S(i,j)$ as they are ordered in $S(i)$. Trains i and j are called *followers* if train i visits the stations in $S(i,j)$ in the same order as train j; otherwise they are called *opposite trains*. We denote by Opp the set of unordered pairs of opposite trains and by $Foll$ the set of unordered pairs of follower trains. Let $C(i,j)$ be the set of connections between consecutive stations in $S(i,j)$. Also the connections in $C(i,j)$ are ordered as encountered by train i. Each connection can be single or double-track, and we denote by $DC(i,j)$ the set of double tracks between $C(i,j)$, ordered in the usual way.
On the microscopic level, routes also include the complete trajectories of trains across the stations, defined as alternating sequences of nodes and arcs of the station graph.

Official and real-time timetable. The *schedule* is a real vector t which specifies the exact timing of all the movements of the train's route. A given component t_k of the schedule t is associated with a train i and with an non-sharable railway resource r, and denotes the time in which i enters r. Non-sharable railway resources include track sections, station platforms or stopping points, converging interlocking routes, etc. A resource is occupied by a train until it enters the next resource on its route. A *timetable* is the sub-vector of a schedule specifying when trains arrive and depart from each station in their route. The *official timetable* is the wanted one, typically planned long in advance but may be updated from one day to the next. The *real-time timetable*, instead, is the outcome of real-time dispatching decisions.

Conflicts and Meet and Pass events. The model described in this report allows only opposite trains to cross each other on double track connections. Follower trains can meet and pass each other in stations several times but cannot overtake on double-tracks. When running on tracks, followers must maintain a safety distance of at least one block section.
Conflicts are associated with schedules. We say that a schedule t *generates a conflict* if, according to t, two trains occupy simultaneously a non-sharable railway resource. So, in order to be implemented,

a schedule must be conflict-free.

Dispatching horizon. The dispatching horizon is the time window which is taken in account for decision making. The real-time nature of the problem implies that the origin of the time window is the current instant in time. There are several, conflicting factors which have an impact on defining its duration. Since, in principle, each re-scheduling or re-routing decision may have an impact on future events, considering a wider time window may have its advantages. On the other hand, the need for fast computation and the growing uncertainty related to events lying in the future suggest that choosing a narrower time window may be more appropriate. In other words, the dispatching horizon is a delicate and open topic which, in any case, must be thoroughly discussed and agreed with the infrastructure manager.

Objective Function Conformity to the official timetable is generally identified as the main factor in determining the quality of the real-time schedule t. In this case, the cost function $c(t)$ is defined as a sum of terms associated with trains, where each term is convex and piece-wise linear with the delay of the associated train. Furthermore, each train may be assigned a specific weight in the objective, based on its relative priority. Typical examples of trains which are assigned high priority are airport trains or freight trains carrying special goods. Many other, alternative objectives can also be taken into account. Some examples are:

- To minimize the number (or cost) of lost connections (as in [14]).
- To minimize the average travel times of passengers.
- To minimize the (estimated) socio-economic cost of delays.
- To minimize the number (or cost) of cancellations.

The Train Dispatching problem (TD) We are now able to state the TD problem:

Problem 2.1 *Given a railway infrastructure and its current status, a set of trains and their current position, find a route for every train from its current position to the destination, and find an associated real-time timetable so that the cost function $c(t)$ is minimized.*

Observe that, in order to solve the TD problem, we need to solve both a routing and a scheduling problem. In this model, the routing problem is restricted to stations, as connections are either single-track or double-track with a fixed direction. The TD problem can be easily modeled by Mixed Integer Linear Programming (MILP) formulations (as in [9] or [4]). However, TD instances of practical interest are typically too large to be attacked directly by state-of-art commercial solvers, and most authors resort to heuristic approaches or to simplified versions of the problem.
We have followed a different path, developing a decomposition technique which makes it possible to apply classical MILP techniques and solve to optimality instances of the TD problem of practical interest. To do so, we identify two major sub-problems: the real-time *Line Dispatching* problem, which amounts to establishing a schedule for the trains so that they only meet in stations or in double track regions (or they do not meet at all), minimizing a given cost function; and the real-time *Station Dispatching* problem, a feasibility problem which amounts to finding suitable routes in the station and

a schedule which matches the given timetable. In the next sections we will show how the TD problem can be formulated as a MILP where the 0,1 variables (say x) are associated with decisions of type "where train a and train b shall meet" or "which platform is to be assigned to train i in station s". The schedule is represented by a real vector t. Now, the MILP associated with a TD problem typically exhibits a block structure as follows:

$$\begin{array}{lll} \min c(t) & & \\ s.t. & & \\ (i) \quad Ax^L + & Bt & \geq b, \\ (ii) & Dt + Ex^S & \geq q \\ (iii) & t \text{ real}, \quad x^L, x^S \text{ binary} & \end{array} \tag{1}$$

where the first block (i) (plus (iii)) is associated with the Line Dispatching problem and the second block (ii) (and (iii)) is associated with Station Dispatching problem. Remarkably, the two blocks "communicate" through a small subset of the continuous variables t, namely those associated with the timetable. In other words, fixing trains arrival and departure times at and from the stations, perfectly decomposes the original MILP into (many) smaller programs. Our methodology can be described as an adaptation of the classical Benders' reformulation to cope with integer slave problems - as discussed in [16]. First, block (ii) is neglected, the problem restricted to (i) and (iii) is solved to optimality, and a schedule t^* is found. If t^* can be extended to a solution satisfying (ii) and (iii), then we are done. Otherwise a suitable feasibility cut - a combinatorial analogue to the classical feasibility Benders' cut in the Benders' decomposition approach (see [10]) - is generated, added to the restricted problem and the process is iterated. Combinatorial Benders' cuts for the general case are introduced and discussed in [3]; in this paper, however, the slave problem is a linear program and the combinatorial cut may be viewed as a strengthening of the standard Benders' feasibility cut.
In the next sections we describe in detail the two problems in this decomposition and how combinatorial feasibility cuts are generated.

3 The Line Dispatching problem

First we discuss the Line Dispatching problem. In this problem, every station is modeled as a black box and the precise movements of trains within each station are ignored, except for the (minimum) time necessary to the train to cross the station. What matters at this stage is only when trains enter and exit the station, namely the real-time timetable. So, for each train $i \in T$ and each station $s \in S(i)$, we introduce two continuous variables a_s^i and d_s^i, representing, respectively, the arrival and departure time of i at and from s. The vector $(a, d) \in \mathbb{R}_+^{2|T|}$ is the real-time timetable.

Single Train Simple Precedence Constraints. Denoting by W_s^i the minimum time necessary for i to cross station s (including the time needed to embark and alight passengers)[1], we have

$$d_s^i - a_s^i \geq W_s^i \tag{2}$$

[1]To simplify the description we assume here that the arrival time is the time the train enters the station, and the departure time is the time the train leaves the station

Also, denoting by Q^i_k the minimum running time for train i to run the track from station s_k to station s_{k+1}, we have:

$$a^i_{s_{k+1}} - d^i_{s_k} \geq Q^i_k \tag{3}$$

Modeling meet and pass events. For any pair of trains $\{i,j\} \in Opp \cup Foll$ and every station $s \in S(i,j)$ we introduce a binary variable y^{ij}_s which is 1 if and only if i and j meet in s. Meeting in $s \in S(i,j)$ implies that i enters s before j leaves s and vice versa. This can be expressed by the following pair of constraints:

$$\begin{aligned} &(i) \quad d^i_s - a^j_s \geq (y^{ij}_s - 1)M \qquad \{i,j\} \in Opp \cup Foll, s \in S(i,j) \\ &(ii) \quad d^j_s - a^i_s \geq (y^{ij}_s - 1)M \qquad \{i,j\} \in Opp \cup Foll, s \in S(i,j) \end{aligned} \tag{4}$$

where M is a suitably large constant so that the constraints become redundant when $y^{ij}_s = 0$.

Meet vectors and meet graphs. The vector y plays a crucial role in our decomposition approach. The subvector y_s corresponding to the components of y associated with a given station s is called *station s meet vector*. We associate with y_s an undirected graph $G(y_s)$, the *station s meet graph*, with vertex set $V = T$ and edge set $E(y_s) = \{\{i,j\} \subseteq V : y^{ij}_s = 1\}$, i.e. there is an edge connecting node i and node j in $G(y_s)$ if and only if the corresponding trains meet in s according to y_s. It is not difficult to see that $G(y_s)$ is an interval graph and y_s is then the incidence vector of the edges of an interval graph (see for example [8]).

Opposite Trains. Opposite trains can also cross in double-track connections. For each pair of opposite trains $\{i,j\} \in Opp$ and every double track $d \in DT(i,j)$ we introduce a binary variable z^{ij}_d which is 1 if and only if i and j cross in d. Denoting by $f(d)$ and $s(d)$ the first and the second station (according to the ordering of $S(i,j)$) at the ends of the double track connection d, we have a pair of constraints similar to its station counterpart (4) but "centered" on track d:

$$\begin{aligned} &(i) \quad a^i_{s(d)} - d^j_{s(d)} \geq (z^{ij}_d - 1)M \qquad \{i,j\} \in Opp, d \in DC(i,j) \\ &(ii) \quad a^j_{f(d)} - d^i_{f(d)} \geq (z^{ij}_d - 1)M \qquad \{i,j\} \in Opp, d \in DC(i,j) \end{aligned} \tag{5}$$

The above pair of constraints ensure that when i and j meet in d, i enters d before j leaves it and viceversa.
Note that since $S(i,j)$ also contains the external, uncontrolled regions, any pair of opposite trains must actually meet somewhere, and we have

$$\sum_{s \in S(i,j)} y^{ij}_s + \sum_{d \in DC(i,j)} z^{ij}_d = 1 \qquad \{i,j\} \in Opp \tag{6}$$

Next, we have to ensure that if trains i and j *do not meet* in station s, then the leading train exits s before the trailing one enters s. It is not difficult to see that this condition is always satisfied by opposite trains for any real-time timetable satisfying (5) and (6). The situation is different for followers.

Followers. For any $\{i,j\} \in Foll$, the above condition must be enforced somehow on the real-time timetable. First, we assume without loss of generality that i is the leading train when the trains first appear in the common region. Then, for each station $s \in S(i,j)$ we introduce a binary variable o_s^{ij}. $o_s^{ij} = 1$ if i enters s before j and $o_s^{ij} = 0$ if j enters s before i. Now, let stations $s, q \in S(i,j)$ with q next to s in the routes of i and j. Then the relative order when entering q can differ from the relative order when entering s only if the trains meet in s. This fact can be easily expressed by the following pair of constraints:

$$\begin{aligned} &(i) \quad o_q^{ij} \le o_s^{ij} + y_s^{ij} \qquad \{i,j\} \in Foll, \quad s \in \bar{S}(i,j), \quad q = succ_i(s) \\ &(ii) \quad o_s^{ij} \le o_q^{ij} + y_s^{ij} \qquad \{i,j\} \in Foll, \quad s \in \bar{S}(i,j), \quad q = succ_i(s) \end{aligned} \tag{7}$$

where $\bar{S}(i,j)$ is $S(i,j)$ without its last station, and $succ_i(s)$ is the station immediately after s according to $\succ_i$.
Now if i and j do not meet in s ($y_s^{ij} = 0$) **and** i is ahead of j in s ($o_s^{ij} = 1$), then i exits s before j enters s:

$$a_s^j - d_s^i \ge (o_s^{ij} - y_s^{ij} - 1)M \qquad \{i,j\} \in Foll, s \in S(i,j) \tag{8}$$

In all other cases the above constraint becomes redundant.
Similarly, if i and j do not meet in s **and** j is ahead of i in s, then j exits s before i enters s:

$$a_s^i - d_s^j \ge (-o_s^{ij} - y_s^{ij})M \qquad \{i,j\} \in Foll, s \in S(i,j) \tag{9}$$

Finally, followers' timetable must respect headway safety rules. This requires the leading train always to be at least one section ahead of the follower. So, let $d \in C(i,j)$ be a connection common to followers i and j and let as before $f(d)$ and $s(d)$ denote the first and the second station at the extremes of connection d, respectively. Since we only consider single section tracks, the leading train must leave d, so entering station $s(d)$, before the follower enters d, so exiting $f(d)$. Note that the leading train in station $s(d)$ is also leading in track d. Then we have:

$$d_{f(d)}^j - a_{s(d)}^i \ge (o_{s(d)}^{ij} - 1)M \qquad \{i,j\} \in Foll, d \in C(i,j) \tag{10}$$

and:

$$d_{f(d)}^i - a_{s(d)}^j \ge -M \cdot o_{s(d)}^{ij} \qquad \{i,j\} \in Foll, d \in C(i,j) \tag{11}$$

3.1 Enhancing the model

The model so far introduced contains in general a large number of binary variables and, even worse, Big_M constraints. As mentioned in the introduction and discussed in detail in a next section, we tackle this problem by delayed variable and constraint generation. The binary variables are needed to model decisions in order to avoid conflicts between pairs of trains and to pick among precedence constraints in disjunctions. Now, simple tests immediately reveal that most difficulties in the solution process are caused by pairs of opposite trains. In fact, a slight delay from the official plan for a given train is not likely to produce effects on many other trains running in the same direction. Indeed the number of take-overs is typically very small. On the other hand, a small deviation may result in many potential conflicts with trains running in the opposite direction. Next, we introduce a new modeling

idea to tackle conflicts for opposite trains which effectively contributes to reducing the computational burden.
If we consider a pair of opposite trains $\{i,j\}$ and a connection $c \in C(i,j)$, then i and j may either cross (i) *before* c, i.e. in a station or in a double track encountered by i before c, or (ii) *after* c. If c is a double-track connection, then case (ii) also includes the possibility that i and j meet in c. In order to model these two possible options, we introduce a binary variable w_c^{ij} for each pair of opposite trains $\{i,j\} \in Opp$ and each connection $c \in C(i,j)$, with $w_c^{ij} = 1$ if i and j meet *before* c and $w_c^{ij} = 0$ otherwise (i and j meet *after* c or *in* c). Denoting by $f(c)$ ($s(c)$) the station which immediately precedes (follows) connection c in the usual ordering, then we have:

- Case $w_c^{ij} = 1$ (i and j meet *before* track c) implies that train j enters $f(c)$ before i exits $f(c)$, which is modeled by the following constraint:

$$d^i_{f(c)} \geq a^j_{f(c)} + (w_c^{ij} - 1)M \qquad \{i,j\} \in Opp, \quad c \in C(i,j) \tag{12}$$

- Case $w_c^{ij} = 0$ (i and j meet in track c or after). If c is not double-track then i and j meet after c, and train i enters $s(c)$ before j exits $s(c)$. This can be modeled by the following Big_M constraint:

$$d^j_{s(c)} \geq a^i_{s(c)} - w_c^{ij}M \qquad \{i,j\} \in Opp, \quad c \in C(i,j) \setminus DC(i,j) \tag{13}$$

  If c is double-track, then i and j can also meet in c and the above constraint must be amended as follows:

$$d^j_{s(c)} \geq a^i_{s(c)} - (w_c^{ij} + z_c^{ij})M \qquad \{i,j\} \in Opp, \quad c \in DC(i,j) \tag{14}$$

  In fact, if i and j meet in c ($z_c^{ij} = 1$), then $d^j_{s(c)} < a^i_{s(c)}$ holds, and (14) becomes redundant. In other words, $d^j_{s(c)} \geq a^i_{s(c)}$ holds only if i and j do not meet "before" c (i.e. $w_c^{ij} = 1$) or in c (i.e. $z_c^{ij} = 1$).

Adding these new variables and constraints to the model allows us to neglect the y variables and drop the associated constraints (4) and (6). Also, we can easily strengthen the model by observing that, if $c, d \in C(i,j)$ and track d is after c, then, if two opposite trains i and j meet before c, they meet before d as well. So we have

$$w_d^{ij} \geq w_c^{ij}, \qquad \{i,j\} \in Opp, \quad c,d \in C(i,j), \quad d \succ_i c \tag{15}$$

A final observation concerns the meet vector y, which plays a crucial role in the Station Dispatching problem to be discussed in the next section. Even if the y variables are neglected for opposite trains, they can be easily derived from the w and the z variables.

4 Modeling the real-time Station Dispatching problem

In our decomposition scheme, a solution to the Line Dispatching problem provides a timetable for the trains in T, namely the arrival and departure time of every train at and from each station of its route. We now need to verify if such timetable is actually achievable, that is, for any station $s \in S$ and every train $i \in T$ through s, we need to find a routing and a scheduling of all of the movements of i through s so that all arrival and departure times match with the timetable and no conflicts in the use of resources arise. This problem closely resembles its off-line version, the *Train Platforming* problem, see [1]. Actually, in most small/medium sized stations, there is only one path from the entry point to a given platform and from the given platform to the exit point. So, there is a unique route from the entry point to the exit point through a given platform. This assumption is acceptable for the Stavanger-Moi line, but may not yield good decision support on other lines (with larger stations).
We state now the Station Dispatching problem for a station $s \in S$. Remark that the solution to the Line Dispatching problem provides us, besides a timetable, also the associated meet vector y_s, which is the incidence vector of the train pairs meeting in s, and the corresponding meet graph $G(y_s) = (T, E(y_s))$. Actually, the timetable for s immediately provides an interval representation of $G(y_s)$, where for each train the extremes of the associated interval are the arrival and departure time, respectively.

Problem 4.1 (Train Platforming) *Let P be the set of platforms of station s, let T be the set of controlled trains and let y_s be a meet vector in station s. For every train $i \in T$ denote by $P(i) \subseteq P$ the set of platforms that can accommodate i. Assign to each $i \in T$ a platform in $P(i)$ so that i and j receive a different platform whenever $y_s^{ij} = 1$ (i.e. $\{i,j\} \in E(y_s)$), or prove that no such assignment exists.*

It can easily be shown that the above problem is NP complete by reduction from list coloring of interval graphs (see [7]). The problem can be formulated as a MILP by introducing a binary variable q_{ip} for each $i \in T$ and each $p \in P(i)$ which is 1 if and only if platform p is assigned to train i. Then q is a feasible assignment if and only if it satisfies:

$$\begin{aligned} &(i) \quad q_{ip} + q_{jp} \leq 1, \quad \{i,j\} \in E(y_s), p \in P(i) \cap P(j) \\ &(ii) \quad q_{ip} \in \{0,1\}, \quad i \in T, p \in P(i) \end{aligned} \tag{16}$$

However, the problem becomes easy ([8]) when $P(i) = P$ for all trains i, i.e. every train can be accommodated in any of the platforms of the station (we call this case the *all-good* Station Dispatching problem). In fact, an assignment exists if and only if $G(y_s)$ can be colored with $|P|$ colors, which in turn can be done easily since $G(y_s)$ is interval. Namely, $G(y)$ can be colored with $|P|$ colors if and only if $G(y_s)$ does not contain a clique K with $|K| > |P|$. Since $G(y_s)$ is interval, this test can be performed in polynomial time. When $P(i) \neq P$ holds for some $i \in T$, then the condition is not sufficient anymore, but stays necessary, that is a feasible assignment exists *only if* $G(y_s)$ does not contain a clique K with $|K| > |P|$. We will use this necessary condition in our solution algorithm.
Finally, observe that if $G(y_s)$ is not connected, then we can decompose the original problem into independent sub-problems, each associated with a connected component. This situation often occurs in small stations, as it occurs if there are periods of the dispatching horizon when the station is empty of trains.

4.1 An algorithm for the Station Dispatching problem.

We first remark once more that the Station Dispatching problem is a feasibility problem, which means that any feasible assignment is a solution to the problem. Next, we summarize the solution algorithm:

1. Find all connected components in $G(y_s)$.
2. For each connected component H do:
   (a) Find a maximum cardinality clique K in H. If $|K| > |P|$ return FALSE
   (b) Find an assignment q_H by applying a heuristic method.
   (c) If q_H not found
   Solve system (16) associated with H. If infeasible, return FALSE.
3. A feasible assignment q is found. Return TRUE.

We discuss now in detail each step of the above algorithm.

Find all connected components in $G(y_s)$. Since $G(y_s)$ is interval and an interval representation is at hand, this problem can be solved in linear time in $|T|$.

Find a maximum cardinality clique K in H. Since H is an interval graph, also this step can be performed in linear time in $|T|$, see [13].

A heuristic search for Station Dispatching. The Station Dispatching problem is a list coloring problem, and thus any algorithm for list-coloring on (interval) graphs may be applied. In our current implementation, we simply resorted to a greedy algorithm. Namely, we visit trains by ascending cardinality of available platforms, assign to the current train the first available platform and adjust all other available platforms accordingly.

Solve the 0,1 program associated with the problem. To solve this problem we use IBM ILOG CPLEX (R) as commercial solver. The solver implements a branch&cut algorithm. Since we deal with a feasibility problem, the search can be halted as soon as a feasible solution is found. In any case, the real-life instances of our test-bed are easily tackled by CPLEX, as the number of platforms is relatively small.

Combinatorial cuts. Since the feasibility of program (16) only depends on trains meeting in s, if (16) is infeasible for some meet vector $\bar{y}_s$ then at least two of the trains meeting in s (according to $\bar{y}_s$) cannot meet there. In other words, any feasible meet vector y_s must satisfy the following linear constraint:

$$\sum_{\{i,j\}\in E} y_s^{ij} \leq |E| - 1 \tag{17}$$

where E is the set of edges of $G(\bar{y}_s)$.

In principle, the above constraint may contain a large number of variables, since many pairs of trains may meet in a station s during the control horizon. However, in many practical situations, the infeasibility is caused by a small set of trains. This is the case, for example, when the infeasibility arises because $G(\bar{y}_s)$ contains a clique K with $|K| > |P|$, which can be tested efficiently. In this case we may let E in (17) be the edges in clique K and we may rewrite (17) as:

$$\sum_{\{i,j\}\subseteq K} y_s^{ij} \leq \binom{|K|}{2} - 1 \tag{18}$$

In many cases, typically arising in small stations $s \in S$ for regional or minor lines, the graph $G(\bar{y}_s)$ is the union of several connected components, namely there are periods when the station is empty of trains. Then, if the problem is infeasible, for at least one connected component H a feasible assignment does not exist. In this case, we may restrict (17) to the edges in H.

5 Solution Algorithm

We apply the classical master-slave decomposition approach to tackle the Train Dispatching problem. The master role is played by the Line Dispatching problem, more precisely by the MILP problem defined by the inequalities from (2) to (15) and by the combinatorial cuts generated by the slave. In turn, the slave corresponds to the solution of $|S|$ Station Dispatching problems, one for each station: if not feasible, the slave will return one or more new combinatorial cuts to be included into the master problem. The master-slave pair is solved in sequence several times, until the last slave problem will be feasible - in which case the current solution is the optimal one - or until the current master turns out to be infeasible, which implies that there exists no solution to the overall TD problem. At iteration k the master program also contains a family E^k of constraints of type (17): each constraint in E^k is associated with a set of train pairs E. We denote[2] by $E^k = \{E_1^k, E_2^k, \dots, \}$ the family of sets associated with the combinatorial cuts at iteration k and we have $\emptyset = E^0 \subseteq E^1 \subseteq \dots$.

Solving the current master problem. A major difficulty arises in solution of the master problem. Indeed, the pairs of opposite ($|Opp|$) and follower trains ($|Foll|$) grow quadratically with the number of trains. Consequently, when dealing with a large number of trains, the number of Big_M constraints from (4) to (15) and the number of related variables y, w, z grows very large. The relaxations of the resulting MILPs tend to be very weak making them very hard to solve in practice. To cope with this, we apply delayed row and column generation. Namely, we solve a sequence of restricted MILPs $M^0, M^1, \dots$ containing only subsets of the variables and constraints of the problem. In order to represent the programs in the sequence, observe that each of the constraints from (4) to (15) is completely identified either by a set of opposite train pairs Opp, or by a set of follower pairs $Foll$. On the other hand, constraints (17) are identified

So, we denote a restricted master program by $M(O, F, E)$, where $O \subseteq Opp$, $F \subseteq Foll$, and E is a family of sets of train pairs. The MILP $M(O, F, E)$ is defined by including all the inequalities (2) and (3) along with the corresponding variables, all combinatorial cuts associated with the sets in E and the subset of inequalities (and variables) (4) to (15) associated with sets O and F.

[2]With a slight abuse of notation, we let E^k denote both the family of constraints and the family of sets of train pairs in one-to-one correspondence with the constraints.

At each iteration of our algorithm we solve the current restricted master $M = M(O, F, E)$. If M is infeasible, then it is easy to see that the original TD problem has no solution Otherwise, let $t^* = (a^*, d^*)$ be the current optimal schedule. By inspection one can easily verify if t^* generates conflicts for some pairs of opposite trains $\bar{O}$. Similarly, we can identify a subset $\bar{F} \in Foll$ of follower pairs involved in conflicts generated by t^*. If at least one of the two sets $\bar{O}, \bar{F}$ is non-empty, we add $\bar{O}$ to O, $\bar{F}$ to F, construct the new restricted master problem associated with such sets (and the former E) and iterate. Otherwise, t^* is conflict-free for the Line Dispatching problem and we need to check if it is possible to fulfill the associated timetable for every train and station by solving the slave problem. Namely, for each station $s \in S$, a timetable[3] t^s is extracted from t^* and $|S|$ Station Dispatching problems are solved to test the feasibility of each individual t^s. If one or more timetables are not feasible, the slave problem returns a set $\bar{E}$ of combinatorial cuts violated by the current solution. The set $\bar{E}$ is added to E, a new restricted master is generated and the method iterates. Next, we summarize our Master-Slave approach to the TD problem:

OptimalTrainDispatching

0. Set i = 0; Set $O = \emptyset, F = \emptyset, E = \emptyset$
1. Solve $M(O, F, E)$ to optimality. If M is infeasible, **STOP** (the TD problem is infeasible)
2. Let t^* be the current optimal schedule. Find all line conflicts generated by t^*.
3. If no conflicts exist
   - 3.a Solve the slave problem associated with t^*. Let $\bar{E}$ be the violated combinatorial cuts.
   - 3.b If $\bar{E}$ is empty, **STOP** (t^* is the global optimal schedule)
   - 3.c Add $\bar{E}$ to E
4. If some line conflicts exist
   Let $\bar{O}$ and $\bar{F}$ be the set of opposite and follower train pairs involved in the current line conflicts.
5. Add $\bar{O}$ to O and $\bar{F}$ to F. GoTo 1

In order to provide an initial upper bound to the master-slave algorithm we developed a simple, heuristic algorithm. In a similar fashion, we decompose the problem into a Line Dispatching problem and a Station Dispatching problem: however the master and the slave problems are solved by a heuristic procedure briefly described in [7]. A practical implementation of our heuristic algorithm is currently operating several regional railway lines in Italy[4].
A final interesting remark is that our decomposition and row generation approach mimics, in some sense, the actual behaviour of human dispatchers which detect potential conflicts and prevent them by establishing a suitable meeting point for the conflicting pairs. Dispatchers then make the drivers follow their decisions by switching traffic signals. Adding a violated constraint to the master is the mathematical equivalent of activating a red signal.

[3]The timetable associated with a station s is simply the vector t^s of all arrival and departure times of the trains arriving and leaving the station.

[4]In the current implementation the solutions generated are displayed to the dispatchers, which can accept them or refuse them.

6 An implementation on the Stavanger-Moi line (Jærbanen) in Norway

The first operative system based on the ideas presented in this article was implemented in Norway in February 2014, backed by the network operator (Jernbaneverket[6]) and train operating companies (NSB[11], Flytoget[5], CargoNet[2]). Such system was developed at SINTEF([15]) for the purpose of providing decision support to Norwegian dispatchers. After a first, positive test campaign in the Trondheim area[5], the main line in the Stavanger region (*Jærbanen*[6]) was agreed by all stakeholders to be a suitable candidate for the first real-life implementation in a Norwegian dispatching central. Figure 1 is a graphical representation of the Jærbane taken from the 2013 OpenTrack infrastructure model.

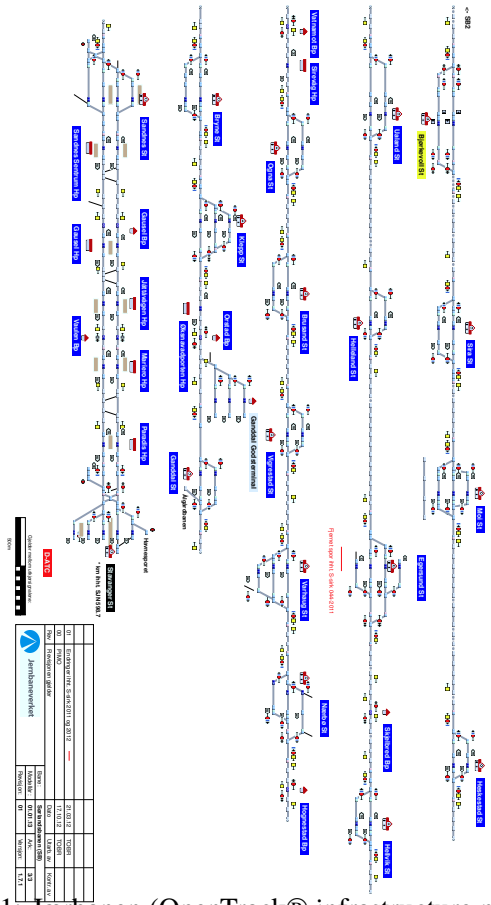

Figure 1: Jærbanen (OpenTrack® infrastructure model 2013)

The Jærbane (123 km, 16 stations) is one of the few lines in Norway with both single- and double-tracks and is among the most trafficked lines in Norway. This is due mostly to the number of local passenger trains which connect Stavanger and Sandnes, which amounts to around 40% of the line's traffic.
We present a brief description of the current setting: first of all, a server continuously acquires real-time data from the Traffic Management System. Each time an event occurs on the line (train reaches

[5]Namely, the Dovrebane, line which connects Trondheim to Dombås, northern gateway to Oslo

[6]Actually, the Jærbane technically comprises the region between Stavanger and Egersund, while the system also controls the part of the network which extends past Egersund to Moi

signaling point, delays are registered, etc.) the algorithm is run to identify a solution to the current problem, which is then displayed on a screen in the Stavanger control center. Figures in Table 1 show that, in most cases, the solution found is optimal. Furthermore, we have noticed that, in the cases when this does not occur, corrupt real-time data in input is generally identified as main cause. While techniques and tools, critical for providing sound advice to the dispatchers, has been developed for this in the Prioritization of trains-project, a thorough description of these are beyond the scope of this report. Dispatchers interact with the system by confirming or modifying solutions and updating parameter settings such as slowdowns, interruptions, delays, cancellations, etc.
In Table 1 we present figures regarding the actual runs of the algorithm in March 2014, which show how most instances are solved to proven optimality within the time limit ("% Optimal"). Figures are presented for different days and include the number of runs considered[7], the mean number of controlled trains ("# Trains"), mean and standard deviation of computation times ("Time", expressed in seconds). Last column ("Objective") indicates average objective value of the identified solutions. Optimization time horizon is set to 12 hours as agreed with the operator and train companies. Time limit for each run was set to 60 seconds by default.

Day	**# Runs**	**% Optimal**	**# Trains**	**Time(s)**		**Objective**
				Mean	Std Dev	
26	3588	85.5%	70	6.54	7.59	2511.27
27	3659	79.0%	76	10.73	12.29	6453.03
28	4986	89.1%	63	3.41	3.30	1259.09
29	3461	99.6%	30	0.45	0.29	1117.72
30	2859	99.9%	34	1.25	1.23	1255.72

Table 1: Figures for system performance on 5 days in March 2014.

References

[1] A. Caprara, L. Galli, P. Toth. *Solution of the Train Platforming Problem*, Transportation Science, 45 (2), pp. 246-257, 2011.

[2] CargoNet, Freight operating company. www.cargonet.no.

[3] G. Codato, M. Fischetti, *Combinatorial Benders' Cuts for Mixed-Integer Linear Programming*, Operations Research, 54 (4), pp. 756-766, 2006.

[4] T. Dollevoet, D. Huisman, L. Kroon, M. Schmidt, and A. Schöbel. *Delay management including capacities of stations*. Transportation Science, to appear.

[5] Flytoget, Norwegian high-speed airport rail link. www.flytoget.no.

[6] Jernbaneverket, the Norwegian government's agency for railway network services. www.jernbaneverket.no.

[7] Runs are triggered when events occur on the line

[7] L. Lamorgese, C. Mannino, *An exact decomposition approach for the real-time train dispatching problem*, Technical Report N. A23274, SINTEF ICT, Norway, 2012.

[8] C. Mannino, *Real-time traffic control in railway systems*, Proceedings of Atmos'11, A. Caprara and S. Kontogiannis (Eds.), OASICS Vol. 20, 2011.

[9] C. Mannino, A. Mascis, *Real-time Traffic Control in Metro Stations*, Operations Research, 57 (4), pp. 1026-1039, 2009

[10] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience, 1999.

[11] Norwegian StatsBaner (Norwegian State Railways). www.nsb.no.

[12] Rete Ferroviaria Italiana, http://www.rfi.it/.

[13] A. Schrijver, *Combinatorial Optimization*, Springer, 2003.

[14] M. Schachtebeck and A. Schöbel, *To Wait or Not to Wait - And Who Goes First? Delay Management with Priority Decisions*, Transportation Science, 44 (3), pp. 307–321, 2010.

[15] Stiftelsen for INdustriell og TEknisk Forskning. www.sintef.no.

[16] F. Vanderbeck, L.A. Wolsey, *Reformulation anad decomposition of integer programs*, in 50 Years of Integer Programming, Eds. Jünger et al., Springer, pp. 431–502, 2010.