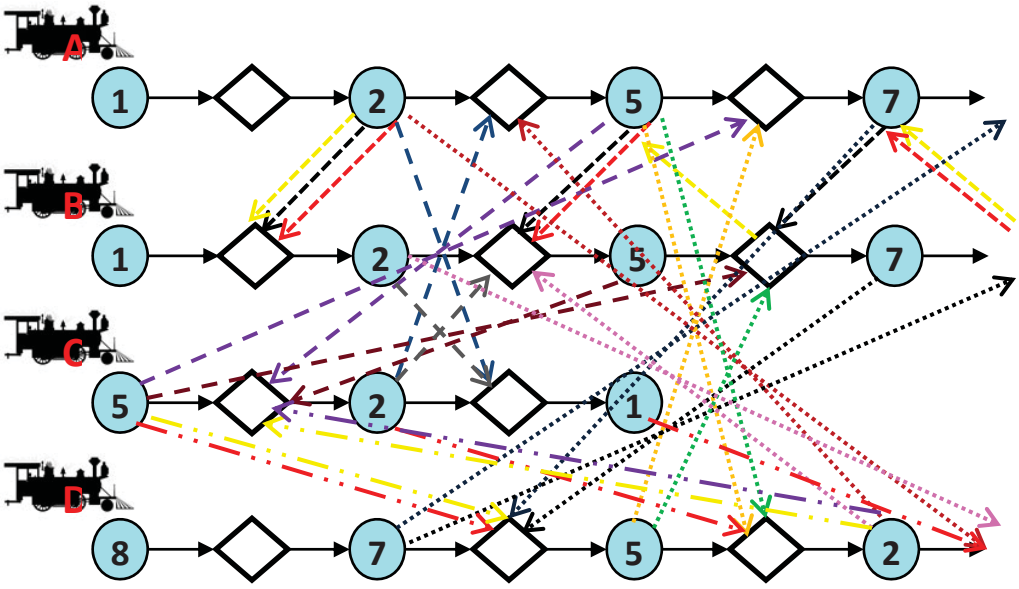
SINTEF

A23274- Unrestricted

Report

An exact decomposition approach for the real-time train dispatching problem

Author(s)
Leonardo Lamorgese
Carlo Mannino



SINTEF ICT
Applied Mathematics
2012-08-09

SINTEF IKT
SINTEF ICT
Address:

NO-
NORWAY
Telephone:+47 73593000
Telefax:+47 22067350
postmottak.IKT@sintef.no
www.sintef.no
Enterprise /VAT No:
NO 948 007 029 MVA

KEYWORDS:
(see \cite{MaMa09})

Report

An exact decomposition approach for the real-time train dispatching problem

Subtitle

VERSION
1

DATE
2012-08-09

AUTHOR(S)
Leonardo Lamorgese
Carlo Mannino

CLIENT(S)

CLIENT'S REF.

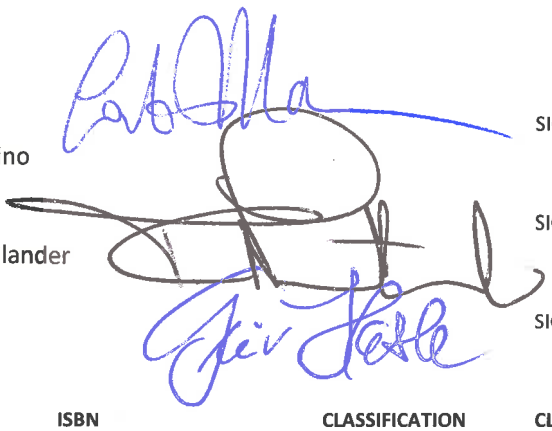
PROJECT NO.
60A14011

NUMBER OF PAGES/APPENDICES:
26

ABSTRACT

Abstract heading

Trains movement on a railway network are regulated by the official timetables. Deviations and delays occur quite often in practice, asking for fast rescheduling and rerouting decisions in order to avoid conflicts and minimize overall delay. This is the real-time train dispatching problem. In contrast with the classic "holistic" approach, we show how to decompose the problem into smaller subproblems associated with the line and the stations. This decomposition allows for the application of suitable simplified models, which in turn makes it possible to apply Mixed Integer Linear Programming to quickly find optimal or near-optimal solutions to a number of real-life instances from single-track lines in Italy.



PREPARED BY Carlo Mannino		SIGNATURE
CHECKED BY Tomas Nordlander		SIGNATURE
APPROVED BY Geir Hasle		SIGNATURE

REPORT NO.	**ISBN**	**CLASSIFICATION**	**CLASSIFICATION THIS PAGE**
A23274	978-82-14-05299-2	Unrestricted	Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
1	2012-08-08	

An exact decomposition approach for the real-time train dispatching problem

Leonardo Lamorgese * Carlo Mannino *

Abstract

Trains movement on a railway network are regulated by the official timetables. Deviations and delays occur quite often in practice, asking for fast rescheduling and rerouting decisions in order to avoid conflicts and minimize overall delay. This is the real-time train dispatching problem. In contrast with the classic "holistic" approach, we show how to decompose the problem into smaller subproblems associated with the line and the stations. This decomposition allows for the application of suitable simplified models, which in turn makes it possible to apply Mixed Integer Linear Programming to quickly find optimal or near-optimal solutions to a number of real-life instances from single-track lines in Italy.

1 Introduction

In a first and gross picture, a rail network may be viewed as a set of stations connected by tracks. Each train runs through an alternating sequence of stations and tracks. Each route also includes the (possibly complicated) movements performed by the train within each station. The trains run along their routes according to the *production plan*; the latter specifies the movements (*routing*) and the times when a train should enter and leave the various segments of its route (*schedule*), including stations arrival and departure times which define the *official timetable*. The generation of the production plan is typically decomposed into two successive phases. In the first phase a tentative *offficial timetable* is established and the arrival and departure times are fixed. In the second phase, called *train platforming* or *track allocation* (see [7], [11]) the complete routes (including station movements) for the trains are established, sometimes by allowing moderate deviations from the tentative timetable.

In principle, the production plan ensures that no two trains will occupy simultaneously incompatible railway resources. In other words, a production plan is a *conflict free* schedule. The problem to design optimal production plans is of crucial relevance for railway operators. As pointed out in [23] *optimum resource allocation can make a*

*SINTEF ICT, Oslo, e-mail: leonardo.lamorgese@sintef.no, carlo.mannino@sintef.no

difference between profit and loss for a railway transport company. Even though the official timetable is conflict free, one or more trains can actually be delayed when running and potential conflicts in the use of resources may arise. As a consequence, re-routing and re-scheduling decisions must be taken in real-time. These decisions are still, in most cases, made by human operators (*dispatchers*), and implemented by re-orienting switches and by controlling the signals status (i.e. setting signalling lights to green or to red), or even by telephone communications with the drivers. The dispatchers take their decisions trying to moderate delays, typically having in mind some ranking of the trains or simply following prescribed operating rules. What the dispatchers are actually doing without being aware of it is solving an optimization problem (and of a very tough nature). Following [30], we call this problem the *real-time Train Dispatching* problem (RTD).

In short, the RTD problem amounts to establish, for each controlled train and *in real-time*, a route and a schedule so that no conflicts occur among trains and some measure of the deviation from the official timetable is minimized. As such, the RTD problem falls into the class of *job-shop scheduling problems* where trains correspond to *jobs* and the occupation of a railway resource is an *operation*. Two alternative classes of formulations have been extensively studied in the literature for job-shop scheduling problems and consequently also applied to train scheduling and routing problems, namely the *time indexed formulations* [17] and the *disjunctive formulations* [4].

In time indexed formulations (TI) the time horizon is discretized, and a binary variable is associated with every operation and every period in the time horizon. Conflicts between operations are prevented by simple packing constraints. Examples of applications of (TI) to train optimization can be found in [7], [8], [10], [11], [20], [30], [34]: actually the literature is much wider, and we refer to [14], [23] and [31] for extensive surveys. To our knowledge, basically all these works deal with the track allocation problem, which is solved off-line and where the feasible time periods associated with train routes are strongly limited by the tentative timetable. In contrast, in the RTD problem the actual arrival and departure times may differ substantially from the wanted ones. Consequently, the number of feasible time periods grows too large to be handled effectively by time-indexed formulations within the stringent times imposed by the application, as extensively discussed in [26]. Another problem with (TI) formulations is that, if the time step is not chosen carefully, they may easily lead to solutions which are practically unrealizable (see [20]).

In disjunctive formulations, continuous variables are associated with the starting times of the operations, whereas a conflict is represented by a disjunctive precedence constraint, namely, a pair of standard precedence constraints at least one of which must be satisfied by any feasible schedule. The *disjunctive graph* ([3]), where disjunctions are represented by pairs of directed arcs, can be associated to any disjunctive formulation and exploited in solution algorithms. This type of disjunctive formulations can be easily transformed into mixed integer linear programs (MILPs) by associating a binary variable with every pair of (potentially) conflicting operations and, for any such

variables, a pair of *big-M precedence constraints* representing the original disjunction. These constraints contain a very large coefficient and they tend to weaken the overall formulation, which is the main reason why (TI) formulations were introduced.

The connection between railway traffic control problems, job-shop scheduling and corresponding disjunctive formulations was observed quite early in the literature. However, a systematic and comprehensive model able to capture all the relevant aspects of the RTD was described and studied only in the late 90s ([25]) and further developed in [27]. In these works, the authors also introduce a generalization of the disjunctive graph that they call *alternative graph*, which we refer to here simply as disjunctive graph. After these early works there has been a flourishing of papers representing the RTD by means of disjunctive formulations and exploiting the associated disjunctive graph. Recent examples can be found in [13], [14], [15], [32]. A comprehensive list of bibliographic references is out of our scope and again we refer to the above mentioned surveys. The great majority of these papers, however, only use the disjunctive formulation as a descriptive tool and resort to purely combinatorial heuristics to solve the corresponding RTD problems. The explicit use of the disjunctive formulation or their reformulations to compute bounds is quite rare, and typically limited to small or simplified instances. Examples are [26], which handles small-scale metro instances, and [32], which introduces several major simplifications, drastically reducing the instances size.

So, methods based on mathematical programming techniques are rarely applied to solve real-life instances of the RTD problem: time-indexed formulations tend to be too large and often cannot even generate a solution within the time limit; big-M formulations tend to be too weak and they can fail to produce feasible solutions within the time limit. Actually, the lack of real-life implementations of theoretical studies regards all known approaches, exact or approximated, as recently observed in [20]. However, in the same paper the author conjectures that the application of optimization to regular dispatching activities is imminent: in this paper we somehow confirm his conjecture. We introduce a new modeling approach to the RTD and a solution methodology which allow to overcome some of the limitations of the standard big-M formulations and solve to optimality (or near) a number of real-life instances in single-track railways within the stringent time limits imposed by the application. The methodology is based on a structured decomposition of the RTD into two sub-problems: the *Line Traffic Control Problem* (LTC) and the *Station Traffic Control Problem* (STC). The LTC amounts to establishing a timetable so that trains never occupy incompatible railway resources. The STC problem is the problem of routing and scheduling trains in a station according to a given timetable. The LTC problem and the STC problem give raise to distinct sets of variables and constraints, which are then solved jointly by using row and column generation.

The decomposition has two major advantages. First, the number of variables and big-M constraints is drastically reduced with respect to the standard big-M formulations. Second, depending on the specific infrastructure, we may choose different models

to represent stations in the STC problem. How we will show in Section 4, the (general) STC problem is NP-hard. However, in some cases of practical impact, simpler models can be considered, leading to polynomial cases. One such case is described in Section 4 along with two different solution approaches. Actually, since the lines may contain quite different station layouts, different models can be applied simultaneously. Also, one can start using the simplified models in every station of the line, and refine it only if constraints violation occur (in a row generation fashion).

Interestingly, this decomposition resembles the normal practice of railway engineers to distinguish between *station tracks* and *line tracks* (see, e.g. [13]) and of actually tackling the two problems separately.

The decomposition principle presented in this paper has been already applied to design a semi-automated traffic control system operating a number of single and double tracks line in Italy. The corresponding solution algorithms have been incorporated into a commercial software developed by Bombardier Transportation, one of the largest multinational companies in the sector. However, the current implementation only makes use of simple greedy heuristics to solve the subproblems. Also, the final decisions are still in the hands of the dispatchers, which may accept or refuse the solutions proposed by the system. On the other hand, the exact algorithm presented in this paper has already been successfully tested on the above mentioned lines.

Summarizing the major contribution of this paper to the current practice:

- We introduce an exact decomposition approach to the real-time Train Dispatching problem.
- We give some complexity results on some variants of the subproblems in the decomposition which are *relevant in the practice*.
- We show how to effectively model the subproblems by mixed integer programs and how to apply delayed row generation to couple them.
- The new approach is able to solve real-life instances of single-track lines from various regions of Italy within the stringent time limits imposed by the application.

2 Problem description

In this section we introduce the main ingredients of the RTD problem. For seek of brevity, we omit here and in the subsequent modeling sections a number of details which are necessary in a practical implementation but irrelevant to describe the crucial modeling and algorithmic issues.

A *Railway Network* is a set S of stations and a set B of tracks (called *blocks*) connecting pairs of stations. Blocks are often partitioned into sections, and, for safety reasons, trains running in a same direction on the same block will be separated by (at least) a fixed number of such sections. We neglect sections in the remainder of

the paper but extending the model to handle such case is immediate. We also neglect other railway infrastructures, such as sidings and cross-overs, but again the extension is straightforward. Next, we examine the elements of the railway network.

Stations. A station can be viewed as a set of *track segments*, the minimal controllable rail units, which in turn may be distinguished into *stopping points* and *interlocking-routes*. A stopping point is a track segment in which a train can stop to execute a service. Two special stopping points are those associated with the entrance and the exit to the station. An *interlocking-route* is the rail track between two stopping points, and is actually formed by a sequence of track segments. For our purposes, a station $s \in S$ is represented by means of a directed graph $G(s) = (N_s, E_s)$ where N_s is the set of *stopping nodes* (corresponding to points) and $E_s \subseteq N_s \times N_s$ is the set of *interlocking arcs* (corresponding to routes). A train going through a station s is running a directed path in the station graph. The path usually contains a *platform* node, where a train can, if required, embark or alight passengers. Also, if the train enters (exits) the station, the path will contain an entrance (exit) node.

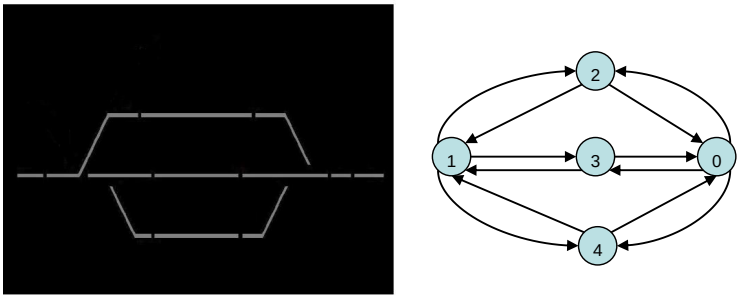


Figure 1: From the station scheme to the oriented graph. Nodes 2, 3, and 4 correspond to platforms

Trains and routes. Our purpose is to model the real-time movements of the trains T along the line. Part of the trains will still be out of the line, while the others will be in stations or running on tracks between stations. A train $i \in T$ running through the line from an initial position to the destination station will traverse all the intermediate stations and blocks. We represent this movement by a graph $R(i) = (V^i, A^i)$ called *train route*. The nodes of $R(i)$ are associated with the blocks, with the (station) stopping nodes and with the (station) interlocking arcs traversed by train i. The graph $R(i)$ is a directed simple path. Every arc $(u, v) \in A^i$ has a weight W_{uv}, and represents a simple precedence constrain, i.e. v is encountered by train i right after u, with W_{uv} being the minimum time to move from u to v. So, if u is block connecting station A to station B

then v is the entrance node of station B and W_{uv} is the minimum running time. If u is a platform in a station then v is interlocking arc going out from u and W_{uv} is the time spent to embark and alight passengers, etc. Since $R(i)$ is a directed path, its nodes are naturally ordered and we let $V^i = \{v_1^i, v_2^i, \dots\}$. Every route will also include an artificial node (the last) representing the out-of-line state.

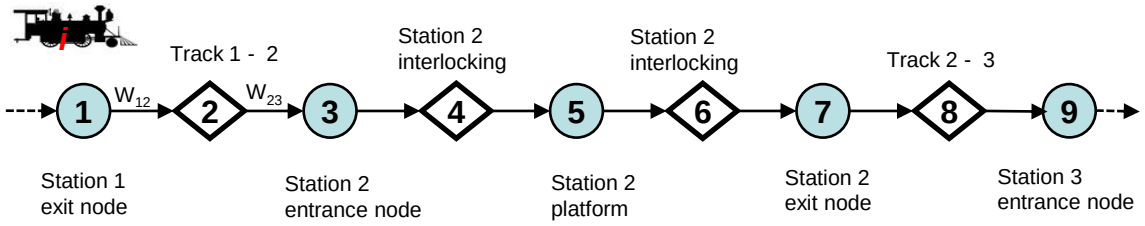


Figure 2: A train route (for train A). Circle nodes correspond to tracks preceded by signals, whereas diamond nodes are interlocking routes or tracks between stations.

The real-time schedule. We now consider a new graph $R = (V, A)$ which is the union of all route graphs $R(i)$, $i \in T$ plus an additional vertex O (the *Origin*) and a directed arc from O to the first node v_1^i of each route $i \in T$. The weight $W_{Ov_1^i}$ of these arcs equals the expected time for the train i to start its route for trains still outside the line, and $W_{Ov_1^i} = 0$ for trains in line.

Every node v in graph R (except the origin) represents the occupation of a rail resource by some train. With every node v we associate a non-negative continuous variable t_v. For $v \in V \setminus \{O\}$ and $v = v_k^i$, the quantity t_v represents the (earliest) time in which train i can enter the $k-th$ node in its route, i.e. the time when the corresponding rail resource can be occupied by train i. Also, we let $t_O = 0$: in other words, node O represents the planning starting time. The vector $t \in \mathbb{R}_+^V$ is called *real-time schedule*. Clearly, every feasible schedule must satisfy the following set of precedence constraints:

$$t_v - t_u \geq W_{uv} \qquad (u, v) \in A \tag{1}$$

Also, any feasible scheduling is such that no two trains occupy simultaneously the same rail resource or incompatible ones. So let $i, j \in T$ be distinct trains, and let $v_k^i, v_l^j \in V$ represent the occupation of incompatible (or same) rail resources. So, either train i enters next rail resource v_{k+1}^i on its route before j enters v_l^j, **or** train j enters next rail resource v_{l+1}^j before i enters v_k^i. This can be expressed by the following disjunctive constraint:

$$(t_{i,k+1} - t_{j,l} \geq 0) \bigvee (t_{j,l+1} - t_{i,k} \geq 0) \tag{2}$$

where, to simplify the notation, we let $t_{x,y} = t_{v_y^x}$. There is one such constraint for every pair of incompatible rail resources visited by any two distinct trains. Disjunctions of precedence constraints can be visualized on the graph R by pairs of dotted arcs.

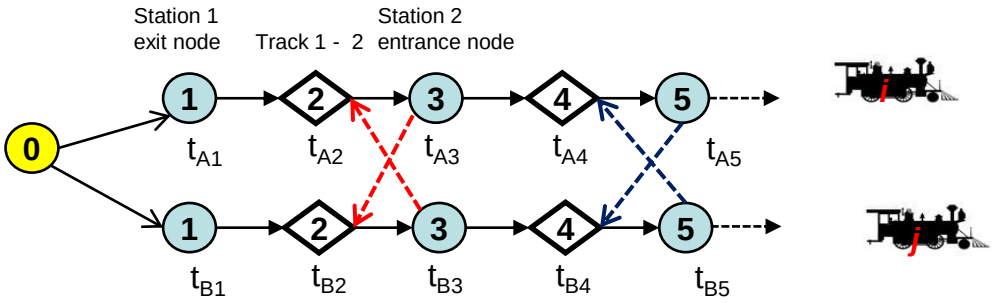


Figure 3: The graph R with disjunctive constraints. Train A cannot enter the track between station 1 and 2 before the train B has entered Station 2, or viceversa

Objective Function The quality of the real time schedule t depends on its conformity to the official timetable. With engineers from Bombardier Transportation and from the Italian railway network operator we have considered here a convex, piece-wise linear function. Let a_s^i be the arrival time of train $i \in T$ in station $s \in S$. We associate with a_s^i a convex piece-wise linear function c_s^i as depicted in Picture 4. The overall cost will be the sum of the costs associated with all trains. Of course, more complicated measures can be considered. The cost for a train is obtained by summing up the delay costs in every station of its route and the overall cost $c(t)$ is the sum of the costs of all trains. Consequently, the cost function $c(t)$ is also convex and piece-wise linear.

The railway traffic optimal real-time traffic control problem We are now able to state the RTD 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 and an associated real-time schedule satisfying all of the (simple) precedence constraints (1) and all of the disjunctive precedence constraints (2) so that the cost function $c(t)$ is minimized.*

Remark that in order to solve the RTD problem we need to solve jointly a routing and a scheduling problem. The RTD problem can be easily modeled by Mixed Integer Linear Programming (as in [26]) or some other techniques to tackle disjunctive programs. However, the RTD instances of some practical interests are typically so large that the corresponding MILP programs cannot be solved by the direct application of some

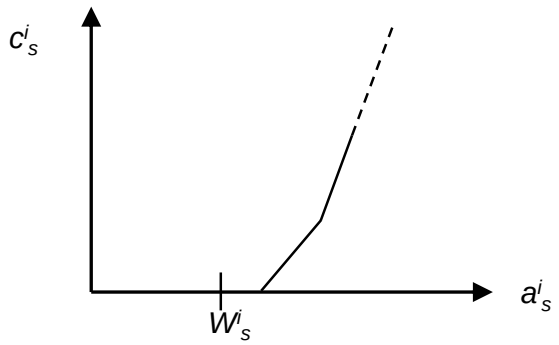


Figure 4: Cost function agreed with practitioners. For train i and station s, W^i_s is the official arrival time, whereas a^i_s is the actual arrival time.

commercial solver or of standard solution techniques. For this reason most authors resort to heuristic approaches or to simplified versions of the problem.

We have followed a different path, namely we developed a decomposition technique which makes it possible to apply classical MILP techniques and solve to optimality instances of the RTD problem of practical interest. In what follows, also driven by our real-life application, we focus our attention on single-track lines. Nevertheless, the decomposition here discussed can be applied to any instance of the RTD problem and constitute a valid scheme for any solution technique.

Single-Track lines We will consider here the case of single-line, single-track railways. "Single-track" means that there is only one track between any two stations; single-line means that the infrastructure graph, with vertices corresponding to stations and edges to the tracks between stations, is a simple undirected path. This choice is motivated as follows:

- The generalization to multi-track railways is straightforward and does not increase significantly the computational burden. In contrast, the presentation of the models and of the algorithms is substantially simplified.

- Also the generalization to multi-line railways is straightforward. However, in this case the computational effort increases (depending on the number of lines). Still, exact or approximated decompositions may be applied (see, e.g., [16]).

- A large share of the railway system is still single-track. For example, in December 2011 in Italy there were 9.218 km of single-track lines against 16723 km in total ([29]): there are more single-track lines than any other type of lines. In Norway in 2009 this ratio is even more extreme ([22]): 3919 km of single-track lines against 4170 in total!

- The methodology here presented will be actually implemented to manage single-track lines of the Italian railway system.

In single-track lines the stations in $S = \{1, \ldots, q\}$ are connected by single tracks (blocks), with block i joining station $i-1$ and station i. Observe that in this situation the routing problem is only limited within the stations as there is only one way to go from a station to another. Also, if two trains meet somewhere on the line, this must be in a station (or some similar facility).

We decompose the RTD problem into two sub-problems. The first is the real time *Line Traffic Control* (LTC) problem and amounts to establishing a schedule for the trains so that they only meet in stations (or they do not meet at all), minimizing a given cost function. The second problem, namely the real time *Station Traffic Control* (STC) problem, is a feasibility problem and amounts to finding suitable routes in the station according to a given schedule. Again driven by the application, we will consider only small sized stations, with some important consequences on the adopted models. The two problems must be solved jointly. In fact, a solution to the LTC problem may result in an inadmissible configuration for the STC problem, as we may not be able to assign routes to trains as scheduled by the LTC (for example when the number of trains simultaneously in the station exceeds the number of platforms available).

3 The real-time Line Traffic Control problem.

The first problem we discuss is the real time Line Traffic Control problem (LTC). We conventionally extend the line with two additional fictitious stations, one for each side, able to accommodate any number of trains. Trains meeting in one of these stations are actually meeting outside the line, or, equivalently, not meeting on the line. As in this subproblem we are neglecting (for the moment) everything that happens in the stations, we may consider simplified routes for the trains. In particular, we assume that for each train i, its route is an alternating sequence of stations and blocks and can be represented by the simple directed path $R(i) = \{v^i_1, (v^i_1, v^i_2) \ldots, (v^i_{l(i)-1}, v^i_{l(i)}), v^i_{l(i)}\}$ where node $v^i_k \in R$ for $1 \leq k \leq l(i)$ is either a station or a block. In particular, the last node $v^i_{l(i)}$ is always the destination station, whereas the first node v^i_1 may be a station or a block, depending on the initial position of the train in the line. If v^i_k is a station then v^i_{k+1} is the next block on the route of train i, and the weight of the arc (v^i_k, v^i_{k+1}) is the minimum time the train is supposed to stay in the station. If v^i_k is a block then v^i_{k+1} is the next station on the route of train i, and the weight of the arc (v^i_k, v^i_{k+1}) is the minimum running time of the train through the block. A particular care must be taken for the first arc (v^1_k, v^i_2), where the weight represents a *residual time*.

Once again we can consider a set of trains T running through the line, the corresponding graph of the routes $R = (V, A)$ obtained as described in Section 2 and the associated schedule $t \in \mathbb{R}^V_+$. The schedule t approximates the behaviour of the trains along the line. In this simplified setting, if v is the node representing station s on the

route of train i, then t_v is the (predicted) arrival time of train i in station s. Similarly, if v is the node representing the block outgoing a station s on the route of train i, then t_v is the (predicted) exit time of train i from station s. Since we are dealing with small stations, this time closely approximates the departure time of the train from the station[1]. Official departure times are of course lower bounds on actual departure times, and can be immediately represented by weighted arcs from the origin to the nodes representing the stations.

Consider now two distinct trains i and j and let $R(i)$ and $R(j)$ be their respective routes. Assume that the trains meet in station $s \in S$ and let v^i_k and v^j_m the vertices representing station s on route $R(i)$ and $R(j)$, respectively. To simplify the following discussion, we may assume neither of these nodes is the last on its route.

Now, since the two trains meet in s then train i exits station s after train j arrives in s and viceversa, i.e. train j exits station s after train i arrives in s. In other words, the schedule t must satisfy the following (conjunctive) pair of constraints:

$$t_{i,k+1} - t_{j,m} \geq 0 \tag{3}$$

$$t_{j,m+1} - t_{i,k} \geq 0 \tag{4}$$

Observe that the above precedence constraints correspond to adding to graph R the arcs (v^i_{k+1}, v^j_m) and (v^j_{m+1}, v^i_k), with 0 weight. This is depicted in Figure **??**, where we consider the case of two trains running in opposite directions and meeting in station s_5; the two precedence constraints are represented by oriented arcs.

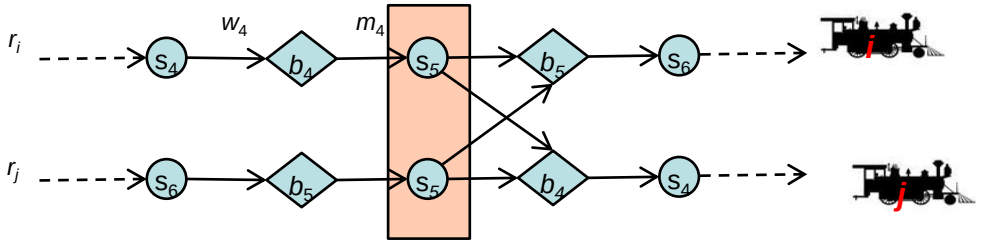


Figure 5: Two trains running in opposite direction and meeting in station $s5$ may satisfy two additional precedence constraints, represented on the graph by two directed arcs.

In the following, trains i and j running in the same direction are *followers*, otherwise they are *crossing trains*. To simplify the discussion we assume now that trains will meet at most once on the line. This is obvious for crossing trains, but not true in general for

[1]By slightly complicating the model these times can be made more exact

a pair of followers, even though in almost all practical cases it is respected. However, the assumption can be easily dropped by a straightforward extension of the model. Another assumption we make for followers is that when the following train catches up with the other train, it becomes the leading train after the meeting (the so called *pass event*). This is what typically happens in practice, where the catching up train is a faster one; again this assumption can be easily dropped in a slightly extended model.

Consider now a pair of followers i and j and assume that i precedes j before they meet in s and j precedes i after the meet. Since we are considering single section tracks, the following train cannot enter a given block before the leading train has left it, i.e. it has entered the next station on the block. This can be trivially expressed by a family of precedence constraints, which, in turn, corresponds to adding to R a family of directed arcs of 0 weight.

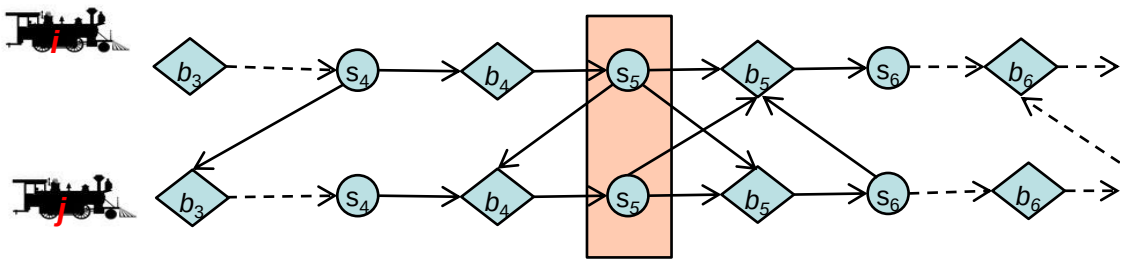


Figure 6: Precedence constraints for two followers meeting in $s5$.

So, in general, the meeting condition of train i and train j in station s translates into a family precedence constraints on the schedule variables, which, in turn, corresponds to a family A_s^{ij} of arcs in R.

The LTC problem amounts to finding a minimum cost schedule t such that all pair of trains only meet in stations. For every $\{i, j\} \subseteq T$, and every $s \in S$, we introduce a binary variable y_s^{ij} and we let $y_s^{ij} = 1$ if i and j meet in s, and 0 otherwise.

Then the LTC can be immediately formulated as follows:

$$
\begin{array}{lll}
\min & c(t) & \\
s.t. & & \\
(i) & t_v - t_u \geq W_{uv}, & (u,v) \in A \\
(ii) & t_v - t_u \geq -M(1 - y_s^{ij}), & (u,v) \in A_s^{ij}, s \in S, \{i,j\} \subseteq T \\
(iii) & \sum_{s \in S} y_s^{ij} = 1, & \{i,j\} \subseteq T \\
 & t \in \mathbb{R}_+^V, \quad y \text{ binary} &
\end{array}
\tag{5}
$$

where M is a large suitable constant. Also, since $c(t)$ is convex and piece-wise linear, it can be easily linearized by adding suitable variables and constraints, and (5) can be turned into a MILP.

Let $(\bar{t}, \bar{y})$ be a feasible solution to (5). Then the binary vector $\bar{y}$ is called a *meeting*.

We discuss now a property of meetings with crucial consequences on the solution algorithms. We recall here that an undirected graph $G = (V, E)$ is called an*interval graph* if it is the intersection graph of intervals of the real line, i.e. the nodes of G correspond to intervals and there is an edge between two nodes if and only if the corresponding intervals overlap.

Lemma 3.1 *Let $\bar{y}$ be a meeting and let $\bar{y}_s \in \{0,1\}^{\binom{|T|}{2}}$ be the subvector of y associated with station s. Then $\bar{y}_s$ is the incidence vector of the edges of an interval graph.*

Proof. Since $\bar{y}$ is a meeting, there exists $\bar{t} \in \mathbb{R}^V_+$ such that $(\bar{t}, \bar{y})$ is feasible to (5). For a train $i \in T$, let Q^i_s be the time interval (possibly empty) in which the train is in station s according to the schedule $\bar{t}$. Let $G_s = (T, E_s)$ be the interval graph associated with the time intervals $\{Q^1_s, \ldots, Q^{|T|}_s\}$. Now, $\bar{y}^{ij}_s = 1$ if and only if $Q^i_s \cap Q^j_s \neq \emptyset$, that is if and only if $\{i, j\} \in E_s$. □

In what follows we let $G(y)$ be the interval graph associated with the meeting y (recall that the union of interval graphs is an interval graph). It is also not difficult to see that, given an interval graph $H = (V, E)$ it is possible to build an instance of the LTC problem with a single station and a solution (t, y) (so that $G(y) = H$, i.e. y is the incidence vector of the edges of H.

A solution (t, y) to the LTC problem cannot in general be extended to a solution to the RTD problem. Indeed, it may be impossible to accommodate the trains in the stations according to the schedule t (which establishes when trains enter and leave the station). The corresponding feasibility problem is discussed in the next section.

4 Modeling the real-time Station Traffic Control problem.

We focus now our attention on a station s. A solution to the LTC problem provides a timetable for s, that is the time in which each train enters and leaves s. In its more general version, the real-time Station Traffic Control (STC) asks for finding, for each train entering or leaving s, a route in s and a scheduling of the movements of the train along its route so that input and exit times from the station agree with a given timetable. This general STC problem closely resembles its off-line version (the *Train Platforming Problem*, see [9]). However, in most practical contexts and in particular in our specific setting we can make reasonable assumptions that make the problem a lot simpler.

First, in most single-track lines and in particular in those considered in our test-bed, the stations are very small, like the one in Figure 1. Basically, for a given platform, there is only one route going through it (two, if you consider the different directions). In other words, there is a one-to-one correspondence between platforms and routes for a given train, and if we choose a platform for train i, then we also establish the station route for i. [2] A second assumption is that the running time from the entrance stopping point to the platform is (approximatively) the same for all trains and all platforms. So, we do not add further delay to a train by selecting, say, platform b instead of platform a. Dropping these assumptions, however, does not complicate the model too much.

Thanks to the above assumptions, the STC problem reduces to decide whether the platforms in the station suffice to accommodate all the trains that arrive at the station, which, in turn, only depends on the meeting vector y.

We state now more formally the (no routing) STC problem.

Problem 4.1 (STC) *Let P be the set of platforms, let T be the set of controlled trains and let y_s be a feasible meeting in the station. For every train $i \in T$ denote by $P(i) \subseteq P$ the set of platforms that can accommodate train i. Then the STC problem is the problem to assign to each $i \in T$ a platform in $P(i)$ so that if $y_s^{ij} = 1$ then i and j receive a different platform.*

Given a undirected graph $G = (V, E)$, a coloring is a function $c : V \to N$ such that $c(i) \neq c(j)$ for all $ij \in E$. A k-coloring is a coloring such that $c(i) \leq k$ for all $i \in V$. Given sets $L(i) \subseteq \{1, \ldots, k\}$ for $i \in V$, a *list coloring* of G is a coloring c with $c(i) \in L(i)$. Consider a function $\mu : V \to N$. A μ-coloring is a coloring c of G with $c(i) \leq \mu(i)$ for every $i \in V$. The coloring, k-coloring, list-coloring and μ-coloring problem amount to establish if a graph G admits a coloring, a k-coloring, a list coloring and a μ-coloring, respectively. The following complexity results are surveyed in [5]: on interval graphs, the coloring problem and the k-coloring problems are easy, the list coloring and the μ-coloring problems are NP-complete.

It is not difficult to see that the STC problem amounts to finding a list coloring of $G(y_s)$, with $L(i) = P(i)$ for every node i. In the previous section we have seen that $G(y_s)$ can actually be any interval graph. It immediately follows the next

Theorem 4.2 *The STC problem is NP-hard.*

Proof. Reduction from list-coloring in interval graphs.

However, for most stations in a single-track line, a more treatable situation occurs, namely $P(i) = P$ for all i and every train can be accommodated in any of the platforms of the station. We call this case the *all good* STC problem.

Lemma 4.3 *The all good STC problem is easy.*

[2] Clearly, this does not hold for larger stations, where several routes go through the same platform.

Proof. When all color lists are equal, the list coloring problem is the standard graph coloring problem. The graph coloring problem is easy for interval graphs.

Finally, there is an intermediate case which occurs in practice. Namely, when platforms have different lengths as well as trains and a train can only be accommodated on a platform which is at least as long. We call this the *progressive* STC problem. We have that:

Lemma 4.4 *The progressive STC problem is NP-complete.*

Proof. Reduction from μ-coloring on unit interval-graphs. A unit interval graph is the intersection graph of unit length intervals. Observe that every μ-coloring uses at most $k_\mu = \max_{i \in V} \mu(i)$ colors. So, given the function μ, and a unit interval graph $H = (V, E)$ we construct an instance of the progressive STC problem in the following way. We consider a single station line. We let the set of trains $T = V$, the platforms $P = \{1, \ldots, k_\mu\}$ and the meeting y be the incidence vector of the edges of H (i.e. $G(y) = H$). Next, for each train $i \in T$, we define its length as $l_T(i) = M - \mu(i)$, where M is a large real number; similarly, for each platform $p \in P$ we let its length be $l_P(p) = M - p$. Suppose that the associated progressive STC problem is feasible, and let $c : T \to P$ be an assignment of platforms to trains. Then c is also a μ-coloring of H. In fact, since $c(i) \neq c(j)$ for all $ij \in E$, c is a coloring of H. Also, for each $i \in T$ we have $l_P(c(i)) \geq l_T(i)$, which implies $M - c(i) \geq M - \mu(i)$, which becomes $c(i) \leq \mu(i)$, and c is a μ-coloring of H. □

Basically all the instances considered in our (real-life) test-bed belong to the all-good STC problem, with exceptions which can be handled easily. In what follows we discuss two different approaches to the solution of the all good STC problem. The first leads to a compact formulation whereas the second to a non-compact one, with a number of constraints which can grow exponentially with the number of trains and the number of platforms. Nevertheless, the non-compact approach has proven to be more effective in practice, as we will show in Section 6. In the sequel the number c_s of platforms of station s will be called the *station capacity*.

A compact, flow based representation of the all good STC problem. Let (t, y) be a solution to the LTC problem, and let $s \in S$ be a station and $i, j \in T$ be two distinct trains going through s. We say that j is a successor of i in s (according to (t, y)) if i leaves s before j enters s. We now introduce for every ordered pair (i, j) of distinct trains and every station $s \in \{1, \ldots, |S|\}$, the quantity x_s^{ij} which is 1 if and only if j is a successor of i in station s. It is not difficult to see that x can be easily obtained from y. For instance, if i runs from station 1 to station $|S|$ and j from $|S|$ to 1 (so they run in opposite directions), and they meet in station $1 \leq k \leq |S|$, then i is a successor of j in every station $s > k$ and j follows i in every station $s < k$. Assuming $i < j$, the above conditions can be expressed by the following constraints:

$$x_s^{ji} = \sum_{q<s} y_q^{ij}, \qquad s \in S$$

. and

$$x_s^{ij} = 1 - \sum_{q<=s} y_q^{ij}, \qquad s \in S$$

Similar transformations may be derived for pair of followers. In general, there exists an affine transformation from y to x, i.e.

$$x = Qy + q \tag{6}$$

with Q and q are suitable matrices.

So let $Su(i, s, x)$ be the set of successors of i in station s. Now, we can think at station platforms as unit resources that can be supplied to trains. Then a train j receives the platform either "directly" from the station s, or from a train i such that $j \in Su(i, s, x)$, which in turn received its platform at an earlier stage. Following this interpretation, we can represent the STC as a network flow problem. Informally, the station s can be represented by a supply node (it supplies up to c_s units of resource) and every train i can act both as a demand node and a supply node, since it can supply 1 unit of resource to successive trains.

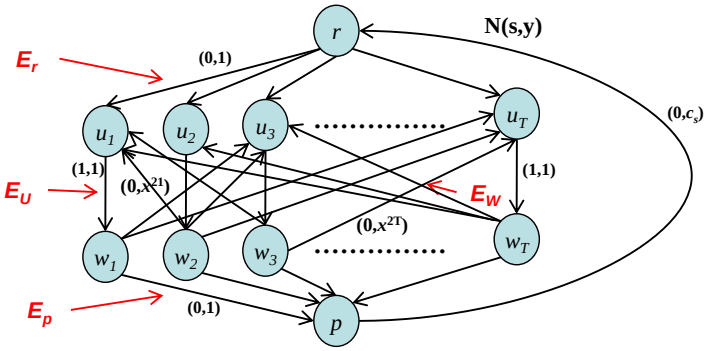


Figure 7: The support network. Lower and upper bounds are shown between brackets for some representative arcs

We consider now a given station s and a meeting $\bar{y}$, along with the corresponding successors vector $\bar{x}$. For sake of simplicity, we assume that every train in T goes through s. Since both s and $\bar{x}$ are fixed, we simply denote $Su(j, s, \bar{x})$ as $Su(j)$. Also, we assume that trains are ordered by their arrival times in station s. So, $i \in Su(j)$ implies $i > j$. Let us introduce a support graph $N(s, \bar{x}) = (\{r, p\} \cup U \cup W, E)$, where $U = \{u_1, \ldots, u_{|T|}\}$, $W = \{w_1, \ldots, w_{|T|}\}$. Let the arc set $E = E_r \cup E_U \cup E_W \cup E_p \cup \{(p, r)\}$, where $E_r = \{(r, u) : u \in U\}$, $E_U = \{(u_j, w_j) : j \in T\}$, $E_W = \{(w_i, u_j) : i, j \in T, i \neq j\}$,

$E_p = \{(w, p) : w \in W\}$. With each arc $e \in E$ we associate lower bound l_e and upper bound f_e. In particular, $l_e = 1$ for $e \in E_U$ and $l_e = 0$ for $e \in E \setminus E_U$. Also, $f_e = 1$ for $e \in E_r \cup E_U \cup E_p$, $f_{(w_i,u_j)} = \bar{x}_s^{ij}$ for $(w_i, u_j) \in E_W$ and $f_{pr} = c_s$.

We have the following

Theorem 4.5 *The all good STC problem has a solution if and only if the graph $N(s, \bar{x})$ has a circulation satisfying all lower and upper bounds.*

We give the necessity proof of this theorem in the Appendix. The sufficiency proof is simpler and is omitted.

Since circulation problems can be expressed as linear programs ([1]), the above result shows that the (all good) STC problem for a station $s \in S$ can also be formulated as a linear program. Coupling this linear program with the MILP associated with (5) immediately provides us a MILP formulation for the RTD problem. However, as we will show in the computational section, the approach discussed next has proven to be more effective in finding optimal solutions to the instance of the RTD problem in our test-bed.

A non-compact formulation for station capacity constraints. Consider a station $s \in S$ with capacity c_s and let (t, y) be a solution to the LTC problem. The station capacity will be violated if and only if there exists a set of trains $K \subseteq T$ such that $|K| = c_s + 1$ and all pairs of trains in C meet in s. If this last condition is verified, then $y_s^{ij} = 1$ for all $\{i, j\} \subseteq K$. Since there are $\binom{c_s+1}{2} = (c_s + 1)c_s/2$ pairs of distinct trains in K, the condition is equivalent to $\sum_{\{i,j\}\subseteq K} y_s^{ij} = \frac{1}{2}(c_s + 1)c_s$.

In other words, the meeting y does not violate any station capacity constraint if and only if, for all $s \in S$, we have:

$$\sum_{\{i,j\}\subseteq K} y_s^{ij} <= \frac{1}{2}(c_s + 1)c_s - 1 \tag{7}$$

for all $K \subseteq T$ with $|K| = c_s + 1$.

5 Solution Algorithm

We are finally able to formulate the RTD problem as a Mixed Integer Linear Program by coupling constraints (7) and program (5) and linearizing the objective function:

$$
\begin{array}{lll}
\min & c(t) & \\
s.t. & & \\
(i) & t_v - t_u \geq W_{uv}, & (u,v) \in A \\
(ii) & t_v - t_u \geq M(y_s^{ij} - 1), & (u,v) \in A_s^{ij}, s \in S, \{i,j\} \subseteq T \\
(iii) & \sum_{\{i,j\} \subseteq K} y_s^{ij} <= \frac{1}{2}(c_s+1)c_s - 1, & s \in S, K \subseteq T, |K| = c_s + 1 \\
(iv) & \sum_{s \in S} y_s^{ij} = 1, & \{i,j\} \subseteq T \\
 & t \in \mathbb{R}_+^V, \quad y \text{ binary} &
\end{array}
\tag{8}
$$

An alternative (compact) formulation is obtained by replacing constraints (8.iii) with the inequalities defining the circulation problem on the network $N(s,x)$ for all s (plus the transformation (6)).

One major drawback of the above formulation is that the number of constraints (8).iii can grow exponentially with the number of trains and the capacity of the stations. Also the number of constraints (8).ii can grow very large in practice, even in our instances with small stations and a relative small number of trains. For this reason we resort to the delayed row generation approach ([2]) which we summarize next. We start by selecting a initial subset of constraints. Then, in each node of the branching tree, we (i) solve the current linear relaxation (ii) check if the current fractional solution violates any of the neglected constraints (*separation*) (iii) add the violated constraints to the current program and iterate. Following this scheme, our initial formulation contains only (all) constraints (8.i) and (8.iv). A major variant of the standard scheme has proven to be more effective in our application. Namely, rather than looking for violated inequalities in every node of the branching tree, we do it only when the associated solution is integral. This approach tends to generate larger branching trees, but this negative effect is more than counterbalanced by the saving in computing times. Another related advantage is that the separation becomes easy. Indeed, whereas the separation of a violated inequality (8.ii) can be done by inspection and is easy, we do not have at hand a polynomial time algorithm for separating inequalities (8.iii) when y is fractional. Actually, if y can assume *any* fractional value, then the separation problem for (8.iii) reduces to the Maximum Edge-Weighted Clique problem in undirected graphs. The latter is known to be an NP-hard problem (see [24]), leaving very little hope to solve the separation efficiently. When y is binary and no constraints of type (8).ii are violated, the situation changes drastically. Indeed, thanks to Lemma 3.1, we know that in this case the graph $G(y_s)$ is interval. Then finding an inequality of type (8.iii) violated by y amounts to finding, for each s, a maximum cardinality clique in the interval graph $G(y_s)$, which can be done in $|T| \log |T|$ (see [19]).

A final interesting remark is that our row generation approach mimics, in some sense, the actual behavior of human dispatchers. A violated constraint (8.ii) corresponds to a so called *line conflict*, that is a situation in which two trains occupy simultaneously

incompatible track sections. Line conflicts are detected by dispatchers and prevented by establishing a correct meeting station for the conflicting trains. The dispatchers then force drivers to follow their decisions by switching suitable traffic signals to red light. Adding a constraint of type (8.ii) is the mathematical equivalent to activating a red signal.

6 Computational Results

As already mentioned, an implementation of our decomposition approach equipped with very simple local optimization heuristics has been already developed in cooperation with Bombardier Transportation and placed into service on several single-track lines in Italy. Besides being unable to certify the quality of the solutions found, a major drawback of the current heuristic implementation is that in some practical situations it may be unable to find feasible solutions even when detectable by expert dispatchers. This is not a big issue in the current practice, as the system is only designed to support the dispatchers by presenting possible solutions, but the operative decisions are still taken by them. On the other hand, this faults would not be acceptable by a fully automated conflict resolution and route setting system.

The major purpose of our test campaign was to demonstrate that an exact algorithm can indeed be implemented in practice, and that optimal solutions can be found within the stringent time limits imposed by the application. Our experiments confirm that an optimal solution can be found very quickly in almost all instances of our real-life test bed.

The real-life instances of our test-bed are from railways in North-Eastern Italy and Sicily, regions with considerably different topography and network status quo.

Namely these railways were:

1. *Trento - Bassano del Grappa (T-BG)*: 22 stations
2. *Gela - Siracusa (G-S)*: 22 stations
3. *Lentini - Canicattì (L-C)*: 19 stations
4. *Roccapalumba - Agrigento (R-A)*: 14 stations
5. *Piraineto - Trapani (P-T)*: 12 stations
6. *Alcamo - Trapani (A-T)*: 14 stations

A second purpose was to compare the flow-based formulation with the non-compact formulation for the all-good STC problem. Our experiments show that the non-compact formulation clearly outperforms the compact one.

Implementation details. All tests were run using CPLEX 12.3 on a shared server with 1.87 GHz Intel® Core CPUs. To implement Row Generation within branching nodes, CPLEX offers different strategies. An extension of the LazyConstraintCallback class was used: the cut separator is called only if the current solution is integer. The recommended settings for cplex were: (i) Turning off presolve, (ii) Avoiding multiple threads (iii) Setting the MIP search to Primal (iv) Turning off dual reduction. In addition we also disabled the heuristic search (*HeurFreq* = -1) and increased integer tolerance.

Results. In our computations, we confronted the performances of the two formulations by running tests using the real-life instances described above. A summary of computed results is shown in table 1.

The time limits for the non-compact version was set to 15 seconds. In most instances, and in particular on the harder ones, and on average, the non-compact formulation outperforms the compact, flow based one. In all cases above reported, the algorithm based on the non-compact formulation found optimal solutions within a few seconds, an acceptable time for dispatchers.

Next, we show the evolution of the integral solutions found by CPLEX and of the gap between the incumbent solution and the best available LB for two instances of the Alcamo - Trapani line:

Name	Line	# trains	Late	Initial Rows		Rows Added		Computation Time(s)	
				C	NC	C	NC	C	NC
1	T-BG	30	3	9483	1401	307	96	20,70	0,95
2	T-BG	27	3	8234	1302	246	130	19,28	0,78
3	T-BG	21	3	4324	908	625	215	18,45	9,35
4	T-BG	28	3	9507	1633	583	964	15,41	14,04
5	T-BG	32	3	4858	918	688	453	16,89	2,69
6	G-S	17	0	2736	747	8	0	0,25	0,08
7	G-S	18	0	2908	771	12	0	0,17	0,08
8	G-S	18	0	2612	725	12	0	0,12	0,09
9	G-S	17	0	2731	742	12	0	0,11	0,2
10	G-S	16	0	2612	725	12	0	0,09	0,08
11	L-C	7	0	596	261	0	0	0,01	0,03
12	L-C	7	0	563	250	0	0	0,05	0,05
13	L-C	7	0	507	234	0	0	0,32	0,21
14	L-C	6	0	480	225	0	0	0,32	0,22
15	L-C	6	0	470	223	0	0	0,03	0,05
16	P-T	26	3	3669	904	399	189	6,81	1,09
17	P-T	24	1	3781	898	741	259	2,98	0,98
18	P-T	24	0	3530	873	790	323	8,10	0,84
19	P-T	24	0	3553	882	311	516	1,38	1,26
20	P-T	23	1	3417	862	313	412	1,31	1,09
21	R-A	14	0	1609	458	325	95	1,69	0,21
22	R-A	16	1	1722	487	228	100	2,95	0,21
23	R-A	14	0	1576	465	251	100	2,40	0,20
24	R-A	14	0	1580	460	228	102	1,73	0,19
25	R-A	14	0	1508	447	175	95	2,19	0,17
26	A-T	20	2	4031	832	975	435	32,57	0,97
27	A-T	20	1	4250	851	661	386	5,74	11,04
28	A-T	19	1	3764	832	707	743	10,14	5,81
29	A-T	19	1	3729	796	590	763	22,25	6,13
30	A-T	18	0	3057	755	830	786	28,08	6,41
Average								**7,56**	**2,18**

Table 1: Computational results. *C* stands for *Compact*, *NC* stands for *Non-Compact*, referring to the method used in the algorithm to ensure that no station capacity constraint is violated

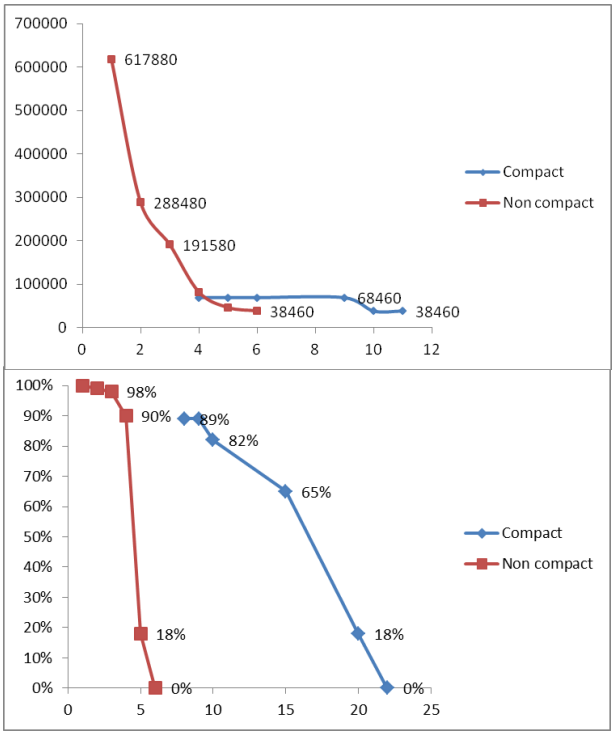


Figure 8: Instance 28: Above, evolution of feasible solutions; Below, evolution of the gap between the best current feasible solution and lower bound

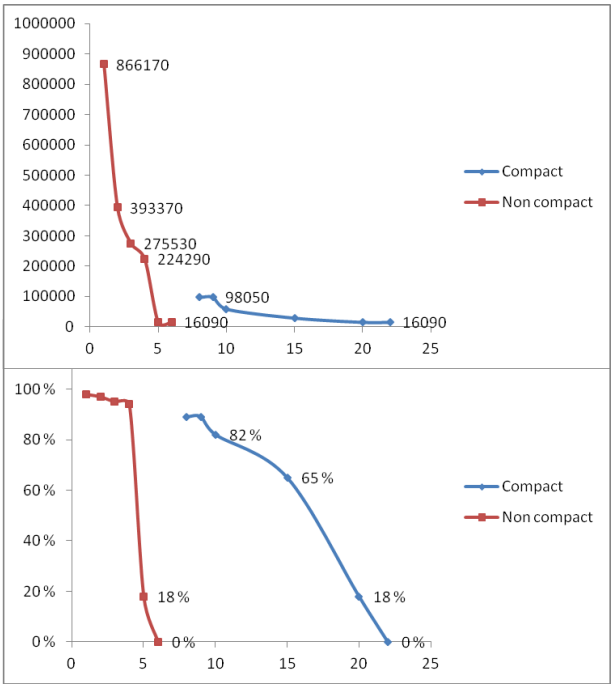


Figure 9: Instance 29: Above, evolution of feasible solutions; Below, evolution of the gap between the best current feasible solution and lower bound

In both cases, solutions of acceptable quality were found quite early in the branching process. In a very few instances of the Trento - Bassano line, the process did not terminate, but good feasible solutions were found within the time limits and the final gap was small.

7 Current developments

A number of research directions are currently being explored. We are investigating the possibility to strengthen our relaxation by identifying stronger valid inequalities. We are looking for new branching strategies and investigating heuristic approaches to initialize CPLEX. We are also studying alternative ways to represent meetings, which appear very promising. Finally, we are extending the approach to deal with double-track lines and multiple routes.

References

[1] Ahuja, R.K., T.L. Magnanti, J.B. Orlin, *Network Flows*, Prentice-Hall, 1993.

[2] Alvras D. and Padberg M.W., *Linear Optimization and Extensions: Problems and Soluzions*. Springer-Verlag, Berlin, Germany, 2001.

[3] Balas, E., Machine sequencing via disjunctive graphs, Operations Research 17 (1969) pp. 941–957.

[4] Balas, E., *Disjunctive programming*, Annals of Discrete Mathematics, **5**, pp. 3–51, 1979.

[5] F. Bonomo, G. Duran, J. Marenco, *Exploring the complexity boundary between coloring and list-coloring*, Annals of Operations Research 169(1), pp. 3–16, 2009.

[6] F. Bonomo, Y. Faenza, G. Oriolo *On coloring problems with local constraints*, Discrete Mathematics, to appear.

[7] Borndörfer R., T. Schlechte, *Models for Railway Track Allocation*, ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, Eds. Christian Liebchen and Ravindra K. Ahuja and Juan A. Mesa.

[8] Brännlund, U., P.O. Lindberg, A. Nou, J.-E Nilsson, *Railway Timetabling using Lagrangian Relaxation*, Transportation Science, Vol 32 (4), pp. 358-369, 1998.

[9] A.Caprara, L.Kroon, M.Monaci, M.Peters, P.Toth, *Passenger Railway Optimization*, in Discrete Optimization, Eds. Aardal, Nemhauser, Weismantel. 129-187 (2007).

[10] Caprara, A., M. Fischetti, P. Toth, *Modeling and solving the train timetabling problem*, Operations Research,50 (5) 292, pp. 851-861, 2002.

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

[12] A. Caprara, L. Kroon, M. Monaci, M. Peeters, P. Toth, "Passenger Railway Optimization", in C. Barnhart, G. Laporte (eds.), *Transportation*, Handbooks in Operations Research and Management Science 14, Elsevier (2007) 129–187.

[13] Conte C, *Identifying dependencies among dealys*, Ph. D. Thesis, University of Göttingen, 2007.

[14] Corman F., *Rail-time railway traffic management: dispatching in complex, large and busy railway networks*, Ph.D. Thesis, TRAIL Thesis Seriess T2010/14, the Netherlands TRAIL Research School.

[15] F. Corman, A. D'Ariano, D. Pacciarelli, M. Pranzo *A tabu search algorithm for rerouting trains during rail operations*, Transportation Research Part B 44 (2010) 175–192

[16] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M. *Optimal inter-area coordination of train rescheduling decisions.* Transportation Research E, Logistics and Transportation Review, 48 (1), 71-88.

[17] Dyer., M., L. Wolsey, *Formulating the single machine sequencing problem with release dates as a mixed integer program*, Discrete Applied Mathematics, no. 26 (2-3), pp. 255-270, 1990.

[18] Goldberg, A.V., R. Tarjan, *Finding minimum cost circulation by successive approximation*, Math. of Op. Res., 15, pp. 430-466, 1990.

[19] U.I. Gupta, D.T. Lee, J.Y.T. Leung, *Efficient algorithms for interval graphs and circular-arc graphs*, Networks 12, pp.459–467, 1982.

[20] S. Harrod, *Modeling Network Transition Constraints with Hypergraphs*, Transportation Science 45 (1), pp. 81-97, 2011

[21] L. Kroon, D. Huisman, E. Abbink, P.-J. Fioole, M. Fischetti, G. Maroti, A. Schrijver, A. Steenbeek, R. Ybema, *The New Dutch Timetable: The O.R. Revolution*, Interfaces 39 (1), pp. 6-17, 2009

[22] Jernbaneverket Presentation 2009, http://www.jernbaneverket.no/PageFiles/7535/JBV-presentasjon_web_2009_05_07.pdf

[23] Luthi M., *Improving the Efficiency of Heavily Used Railway Networks through Integrated Real-Time Rescheduling*, Ph. D. Thesis, ETH Zurich, 2009.

[24] E.M. Macambira, C.C. de Souza, *The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations*, European Journal on Operational Research, 123 (2), pp. 346–371, 2000.

[25] Mascis A., *Optimization and simulation models applied to railway traffic.* Ph.D. thesis, University of Rome "La Sapienza", Italy, 1997. (In Italian).

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

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

[28] M. Montigel, *Semi-Automatic Train Traffic Control in the New Swiss Lötschberg Base Tunnel*, IRSA-Apect 2006,
www.systransis.ch/fileadmin/2006_Paper_MM.pdf

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

[30] G. Sahin, R.K. Ahuja and C.B. Cunha, *Integer Programming Based Approached for the Train Dispatching Problem*, Tech. Rep. Sabanci Univerisity, 2010.

[31] J. Törnquist, *Railway Traffic Disturbance Management*, Ph.D. Thesis, Blekinge Institute of Technology, 2006

[32] J. Törnquist, J. A. Persson, *N-tracked railway traffic re-scheduling during disturbances*, Transportation Research Part B 41, 342–362, 2007.

[33] Zwaneveld, P.J., Railway Planning. Ph.D. Thesis, Rotterdam School of Managemente, TRAIL, Rotterdam, 1997.

[34] Zwaneveld, P.J., L.G.S. Kroon, H.E. Romeijn, M. Salomon, S. Dauzere-Peres, S.P.M. Van Hoesel, H.W. Ambergen, *Routing trains through railway stations: model formulation and algorithms*, Transportation Science, 30 (3), pp. 181-194, 1996.

8 Appendix.

Necessity proof of Theorem 4.5.
Proof.

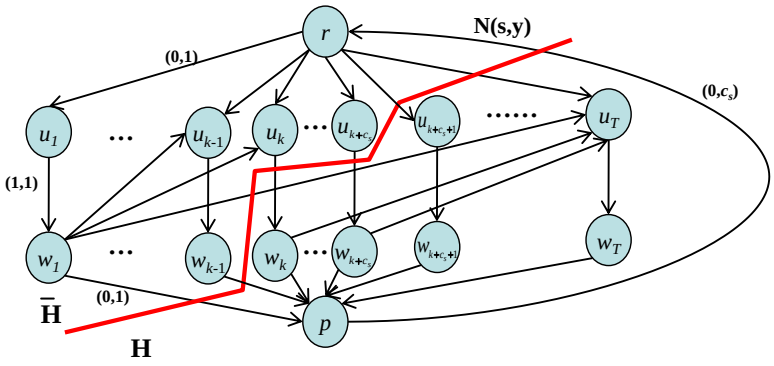


Figure 10: The cut in the necessity proof of Theorem 4.5.

We show that if there does not exist a platform assignment for some station s, then $N(s, \bar{y})$ does not admit a circulation. We use Hoffman's circulation theorem, which states that N does not have a circulation if and only if there exists a set of nodes H such that $\sum_{e \in \delta^-(H)} l_e > \sum_{e \in \delta^+(H)} f_e$. To simplify the explanation, we drop from the support network all arcs with 0 lower and upper bounds (i.e. the arcs in the set $(w_i, u_j) \in E_W$ with j not a successor of i). So, assume that a platform assignment does not exist, then there exist $c_s + 1$ trains, say $Q = \{k, \ldots, k + c_s\} \subseteq T$, which

are simultaneously in station s. We construct a cut by letting $H = \{p\} \cup \{w_j : j = \{k, \ldots, |T|\} \cup \{u_j : j = \{k + c_s, \ldots, |T|\}$.

We then have $\sum_{e \in \delta^-(H)} l_e = |Q| = c_s + 1$ since the only arcs with positive lower bound (the arcs in E_U) entering H are precisely the arcs $(u_k, w_k), \ldots, (u_{k+c_s}, w_{k+c_s})$ (all other arcs in E_U are either completely contained in H, for $j > k + c_s$, or in the complement $\bar{H}$ of H, for $j < k$).

On the other hand, it is easy to see that the only arc with positive upper bound outgoing from H is (p, r), which implies $\sum_{e \in \delta^+(H)} f_e = c_s < c_s + 1 = \sum_{e \in \delta^-(H)} l_e$.In fact:

1. $E_r \cap \delta^+(H) = \emptyset$. Indeed, all arcs in E_r are outgoing from r and $r \in \bar{H}$.

2. $E_U \cap \delta^+(H) = \emptyset$. Indeed, $u_j \in H$ for$j = k + c_s, \ldots, |T|$. But then also $w_j \in H$ for $j = k + c_s, \ldots, |T|$.

3. $E_W \cap \delta^+(H) = \emptyset$. We must show that $(w_i, u_j) \notin E_W$ for $i \geq k$ and $j \leq k + c_s$. That is, we show that for $j \in \{1, \ldots, k, \ldots, k + c_s\}$ and $i \geq k$, then $j \notin Su(i)$. This is trivial for $i > k + c_s$ since $j \notin Su(i)$ for all $i > j$. Also, by assumption, the trains in $Q = \{k, \ldots, k + c_s\}$ are simultaneously in the station, which implies that $j \notin Su(i)$ for all $j, i \in Q$.

4. $E_p \cap \delta^+(H) = \{(p, r)\}$. Trivial, since $p \in H$ and all arcs in $E_p \setminus \{(p, r)\}$ are incoming in p.

□

Technology for a better society

www.sintef.no