Real-Time Traffic Control in Railway Systems

Carlo Mannino

Dipartimento di Informatica e Sistemistica, Università di Roma "Sapienza", e-mail: mannino@dis.uniroma1.it.

Abstract

Despite the constantly increasing demand of passengers and goods transport in Europe, the share of railway traffic is decreasing. One major reason appears to be congestion, which in turn results in frequent delays and in a general unreliability of the system. This fact has triggered the study of efficient ways to manage railway traffic, both off-line and real-time, by means of optimization and mathematical programming techniques. And yet, to our knowledge, there are only a few fully automated real-time traffic control systems which are actually in operation in the European railway system; in most cases such systems only control very simple lines and actually they only support the activity of human dispatchers. We describe here two recent optimization based applications to real-time traffic control which have actually been put into operation in the Italian railways. One such system has been able to fully control the trains in the terminal stations of Milano metro system. The other one will be fully operative by the end of 2012, when it will control the trains on several Italian single-track railways. Both systems heavily rely on mixed integer programming techniques to elaborate good quality timetables in real time.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems; G.1.6 Optimization; I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Train Timetabling, Real-Time Traffic Control, Integer Linear Programming

Digital Object Identifier 10.4230/OASIcs.ATMOS.2011.1

1 Introduction

The demand of people and freight transportation in Europe is increasing at a rate of 2% per year: in contrast, the share of railway traffic is decreasing (from 11% in 2000 to an expected 8% in 2020) (see [5]). Apparently, the major reason for such decrease is a general unreliability of railway systems when compared with other transport modes. In recent years this fact triggered the investigation of new mathematical models and approaches to manage railway traffic, both off-line and real-time. Off-line optimization approaches devoted to timetabling, routing and train platforming have been implemented and applied successfully to tackle real-life problems (e.g. [3, 4, 8]). In contrast, and maybe quite surprisingly, there are very few examples of optimization systems actually in operation to manage railway traffic in real-time and such systems typically control very simple lines, with their tasks restricted only to support human dispatchers. One such system ([13]) is managing the Lötschberg Base Tunnel (operated by the Swiss BLS).

Due to the relevance of the problem, in the past decade there has been a flourishing of studies and experimental implementations; the literature is quite ample and we refer to [5] for a recent survey. Nevertheless, most of the algorithms presented in the literature never went beyond a laboratory implementation and, to our knowledge, they are not yet operative. This is maybe a consequence of the widespread reluctance of network operators to rely on automatic systems, also due to a number of unsuccessful attempts to tackle the

11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems.
Editors: Alberto Caprara & Spyros Kontogiannis; pp. 1–14

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

problem with different techniques, such us expert systems, rule based systems, etc. One other major obstacle is that real-time systems must be able to suitably model even the smallest and utmost specific detail of the real network, and, at the same time, response very quickly when invoked. So, besides representing an extremely challenging mathematical problem, real-time train traffic control also requires the implementation of intricate software packages, totally out of the interest (and skill) of scientists and academic staff.

In this paper we describe the basic mathematical ingredients of real-time railway traffic control systems which have been, or currently are, or will soon be, put in operation in different Italian railway lines. In particular, the first example concerns the metro terminal stations of the city of Milano, whereas the second example refers to a number of single-track railways in different Italian districts, from the very northern line Trento-Bassano to the multi-line of the Sicilian district. Three major actors were involved in the development of the systems: the network operator bringing the problem and, in some sense, an entire railway; the academia, providing the mathematical tools and implementing efficient solution algorithms; a (large) company of the transport sector, capable to design and implement the hardware and software components interfacing the mathematics to the real-world. In my opinion, only this blend of skills allowed for the practical achievement of the tools whose basic ingredients we describe next.

Stations and railway lines may be seen as sets of track segments, each accommodating at most one train, which can be accessed from other track segments either directly (as the two segments are adjacent on a same track) or through switches. The structure of stations and lines can be suitably represented by the *infrastructure* digraph, in which both arcs and nodes represent specific track segments, as we will show in detail in a next section. A train runs through a specific sequence of track segments, called *train route*. The *official timetable* associates a time with specific track segments of each train-route, namely with the track entering (*arrival time*) and leaving (*departure time*) each station. The (real-time) Railway Traffic Control problem (*RTC*) consists in finding a minimum cost *real-time plan* which is a suitable route for each train and the time in which the train enters each segment in its route. The cost of the real-time plan is a function of the deviation from the official timetable. Observe that trains compete to access the track segments and decisions must be taken in order to establish which train precedes which on possible conflicting segments.

The (RTC) has long been recognized as a particular *job-shop scheduling problem*, where trains correspond to *jobs*, tracks to *machines* with unit capacity, and the use of a track segment by a train corresponds to an *operation* (see, e.g., [9, 12]). Two major mathematical models were adopted to represent job-shop scheduling problems and railway traffic control problems. In the first one (see, e.g., [2]), the scheduling variables are continuous real variables, each representing the time in which a given operation is started. In the second model (introduced in [6]), often referred to as *time-indexed formulation*, the time horizon is discretized into a finite set of time periods, and the main decision variables are 0,1 variables associated with a given operation and a specific time period.

The second approach has several, relevant advantages w.r.t. the first. The most important one lies in the way we express the fact that some track segments (distinct or not) cannot be occupied simultaneously by two distinct trains, such as a passenger platform and the track segment (*interlocking route*) to access it. In particular, in time-indexed formulations, such incompatibility constraints are expressed by simple cardinality inequalities in which at most one variable can be one. In the time-continuous approach, in contrast, an incompatibility is typically expressed by means of the so called Big-M constraint which requires the introduction of an additional 0,1 variable and of a large coefficient M. It is a well known

fact that time-indexed formulations provide much stronger relaxations, and, in turn, better bounds and smaller search trees. In addition, handling additional and heterogeneous constraints is in general a much easier task. Also, and quite important, routing and scheduling can be immediately handled in a unique framework, i.e. by associating the time-indexed variables to route-segments rather than track segments.

However, time-indexed formulations present a serious inconvenience, that is that the number of variables and constraints grows very quickly with the time horizon and the discretization step. This inconvenience appeared to be deadly when solving real-time traffic control problems, at least in our experience, where the optimization stage must return a reasonably good solution, possibly optimal, within at most one second. For this reason we decided to resort to the first and more classical continuous representation.

2 Real-time traffic control in metro stations

In year 2001 the municipal transport company of the City of Milano, Azienda Trasporti Milanesi (ATM), recognized the potential of applying optimization techniques to control in real-time the running trains in the terminal stations of the Milano metro system. The task of implementing the overall software was assigned to a large multinational of the transport sector, namely Bombardier Transportation, and later the University of Rome Sapienza came on board to develop the core optimization algorithms.

The main challenge of such algorithms was to generate a real-time plan that optimized a specific performance indicator, such as punctuality or regularity. In practice, human dispatchers solve several instances of the (RTC) every minute. To generate an effective plan, an optimization algorithm (referred to as *Optplan*) for the (RTC) needed to be embedded in the *traffic control loop*(see Fig. 1): The position of the trains and the status of the switches and of the signals were captured by remote control equipment and input to Optplan, which returned a real-time plan. The system then signalled to the trains the next move to make on their assigned routes.

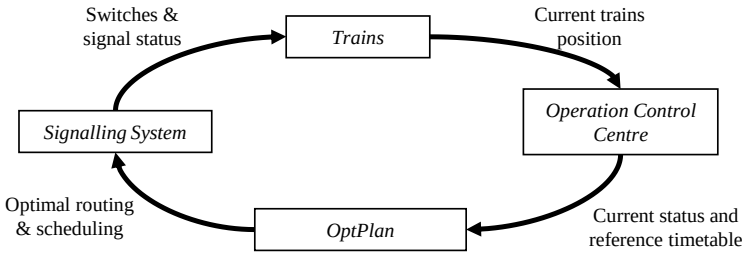

Figure 1 The control loop

An upper limit of five seconds was established for the execution of a complete control loop, which left fewer than one second available to Optplan. Indeed, the execution of the traffic control loop had to be designed in a way so as no additional delays to the standard traffic-management decision process be added. The headway between trains in peak hours is exactly 90 seconds at Sesto F.S., the Milan metro network main terminal station. This tight

schedule stretches the station capacity to its limits. Consequently, even a few additional seconds result in an unrecoverable delay. Second, Optplan needed to be able to quickly re-compute a new plan whenever a dispatcher intervened in real-time. In the case of train failures, dispatchers can reroute trains, edit the official timetable and modify the available network infrastructure. Consequently, they need Optplan to do the same: re-compute plans accordingly and show the new plans immediately on their monitors. The need for quick and effective re-routing is especially crucial in peak hours when dispatchers are under severe pressure as they simultaneously control several monitors, interact with other operators, make radio calls to drivers, and so on. There is no time for a slow system.

After a rigorous and extensive test-campaign, the system proved not only to be able to control in real-time the trains in the terminal stations, but also to produce better real-time plans than human operators, significantly improving over all performance indicators. Only thanks to such positive comparison, as by contractual clause, the automatic route setting system was accepted by ATM and put into operation on July 2007. In what follows we give a brief description of the model and the algorithm developed to solve the (RTC). A comprehensive description of the methodology and of the computational tests can be found in [10]. We start with some basic definitions.

Stations. A (terminal metro) station is a facility where passengers may board and alight from trains, and in which trains can reverse direction or perform a number of additional operations. Such operations are called *train services*. A metro 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 while 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 metro station is represented by means of a directed graph $M = (P, I)$ where P is the set of *stopping nodes* (points) and $I \subseteq P \times P$ is the set of *interlocking arcs* (routes). A performable service is associated with every stopping node $p \in P$.

Trains. Trains enter terminal stations in order to execute a sequence of services; thus trains are defined as an ordered list of services along with an origin, a destination and a planned departure time (according to a given master timetable). The set of trains to be scheduled will be denoted by $T = \{1, \dots, |T|\}$, while D_j is the planned departure time of $j \in T$. Finally, for all $i, j \in T$, we assume $D_i \leq D_j$ whenever $i < j$, i.e. trains are ordered by increasing departure times.

Routes. Train movements within a station may be viewed as ordered sequences of stopping points and interlocking-routes, which in turn correspond to directed paths of M. Such paths are called *(train) routes*. Observe that every route r corresponds to an ordered list of services (each associated with a node of r). Therefore, a route r will be called *feasible* for a train $j \in T$ if the ordered list of services associated with j is contained in the ordered list of services associated with r. A *feasible routing* for $T = \{1, \dots, |T|\}$ is a family $R = \{r_1, \dots, r_{|T|}\}$ of routes such that, for every $j \in T$, r_j is feasible for j. The set of the feasible routings of a station M for a set of trains T will be denoted by $\mathcal{R}(M, T)$. Let $R \in \mathcal{R}(M, T)$, let $r_j \in R$, and let $p \in P$ be any stopping point of r_j. We associate with p a *duration* $d_p(j)$ which depends on the service available in p and on the train j associated with r_j. In addition, with every interlocking arc $a \in r_j$ we associate a *travel time* $d_a(j)$.

Scheduling Nodes and arcs of a route r correspond to rail tracks. In order to provide a complete description of the movements of a train along its route r, we need to establish the exact time when the train enters each track, or, equivalently, a *starting time* for all of the nodes and arcs of r. Now, let $a = (u, v) \in r$, and let t_u, t_v, t_a denote the starting times

of nodes u, v and arc a, respectively: then, since the train enters stopping point u *before* running interlocking-route a, it must be $t_a - t_u \geq d_u$ (*precedence constraint*). Also, since a train cannot be stopped while running through an interlocking-route (*no-wait constraint*), we have $t_v - t_a = d_a$. If $R \in \mathcal{R}(M,T)$ is a feasible routing, an assignment of starting times to all nodes and arcs of all routes in R is called a *schedule* for R.

The problem of computing a schedule for $R \in \mathcal{R}(M,T)$ falls into the class of *job-shop scheduling* problems where trains can be viewed as *jobs*, tracks are *machines* and train movements at stopping nodes and through interlocking arcs are *operations*. Also, observe that a train cannot move away from a stopping point if the next one on its route is occupied by another train (*blocking constraints*). Blocking constraints can be expressed by a disjunction of linear constraints on the starting times. Suppose routes $r_1, r_2 \in R$ share a common stopping node u and let $a_1 = (u,v) \in r_1$ and $a_2 = (u,w) \in r_2$ and let t_{u1}, t_{a1} (t_{u2}, t_{a2}) be the starting times of Train 1 (Train 2) associated to u (u) and to a_1 (a_2). If Train 1 precedes Train 2 in u, then Train 2 can enter u only when Train 1 has already moved to a_1, i.e. $t_{u2} - t_{a1} \geq 0$. Analogously, if Train 2 precedes Train 1 in u, then $t_{u1} - t_{a2} \geq 0$. Therefore, $t_{u1}, t_{a1}, t_{u2}, t_{a2}$ satisfy the following *disjunctive constraint*:

$$(t_{u2} - t_{a1} \geq \epsilon) \bigvee (t_{u1} - t_{a2} \geq \epsilon) \tag{1}$$

where $\bigvee$ denotes that at least one of the two constraints of the disjunction must be satisfied. Observe that the disjunctive constraint (1) generalizes the standard one for job-shop scheduling, because distinct machines (tracks) may be involved.

Schedule costs. Costs represent deviations of the actual schedule from the master timetable. Clearly, early and late trains must be penalized. This is done by introducing a convex, piecewise linear function $g_j(s_j)$, for $j = 1, \ldots, |T|$, where s_j is the departure time of train j. Also, the time-lag between the departures of two consecutive trains $j-1$ and j must equal the planned one (*regularity lag*). The corresponding cost $f_j(s_j - s_{j-1})$, $j = 2, \ldots, T$ is again a convex, piecewise linear function.

The overall schedule cost $c'(s)$ is computed by summing up the two cost functions, and only depends upon departure times s_j, for $j \in T$:

$$c'(s) = \sum_{j=1}^{|T|} g_j(s_j) + \sum_{j=1}^{|T|} f_j(s_j - s_{j-1}), \tag{2}$$

where s_0 is the last departure time. We are finally able to state the Metro-Station Traffic Control Problem (m-RTC).

▶ **Problem 2.1.** [Metro Station Traffic Control Problem] Given a set of trains T, a metro-station $M(P,I)$ and earliness-tardiness and regularity costs g_j and f_j, for $j \in T$, find a feasible routing $R^* \in \mathcal{R}(M,T)$ and a schedule t^* for R^* such that the sum of the earliness-tardiness and regularity costs is minimized.

In short, the (m-RTC) is tackled by enumerating all feasible routings in $\mathcal{R}(M,T)$ and then by solving, for each $R \in \mathcal{R}(M,T)$, the associated job-shop scheduling problem. Therefore, for any $R \in \mathcal{R}(M,T)$, we have a set of operations $N = N(R) = \{0, \ldots, n\}$, where 0 is a dummy operation (called *start*), while the operations $\{1, \ldots, n\}$ correspond to the stopping nodes and the interlocking arcs of all of the routes in R. With every $i \in N$ we associate a *starting time* $t_i \in \mathbb{R}$. The vector $t \in \mathbb{R}^{n+1}$ is called a *schedule* of N, and we assume

$t_i - t_0 \geq 0$, for all $i \in N$. The departure time s_j of Train $j \in T$ is related to the starting time of the exit node $d(r_j)$ of r_j through the equation $s_j = t_{d(r_j)} - t_0$, for $j \in T$.

Feasible schedules must satisfy a number of *precedence constraints* between pairs $i, j \in N$ of the type $t_j - t_i \geq l_{ij}$, where $l_{ij} \in \mathbb{R}$ is a *time-lag*. We indicate the precedence constraint $t_j - t_i \geq l_{ij}$ by $\{i, j, l_{ij}\}$, or simply by (i, j) if the time-lag is omitted.

A (unordered) pair of precedence constraints $(\{i, j, l_{ij}\}, \{h, k, l_{hk}\})$ is a *disjunctive precedence pair* for N if every feasible schedule t satisfies either $t_j - t_i \geq l_{ij}$ or $t_k - t_h \geq l_{hk}$.

▶ **Problem 2.2.** [Job-shop Scheduling Problem] Given a set of operations $N = \{0, \ldots, n\}$, a set of precedence constraints F, a set of disjunctive precedence constraints A and a cost function $c : R^{n+1} \to R$, find a (feasible) schedule $t \in R^{n+1}$ such that all constraints are satisfied and $c(t)$ is minimized.

The job-shop scheduling problem is NP-hard and can be formulated as the following *disjunctive program*:

▶ **Problem 2.3.**

$$\begin{array}{lll} \min & c(t) & \\ s.t. & t_j - t_i \geq l_{ij} & (i,j) \in F \\ & (t_j - t_i \geq l_{ij}) \bigvee (t_k - t_h \geq l_{hk}) & ((i,j),(h,k)) \in A \\ & t \in \mathbb{R}^{n+1} & . \end{array}$$

The set of feasible schedules of an instance of the blocking, no-wait job-shop scheduling problem can be represented by means of the so called *disjunctive graph* $D(N, F, A)$, where N is a set of nodes, F a set of directed arcs, A a set of (unordered) pairs of directed arcs. The arcs in F are called *fixed arcs*. The arc pairs in A are called *disjunctive arcs*. Finally, denoting by $Z(A) = \{(i, j) : ((i, j), (h, k)) \in A\}$ the set of all directed arcs in (the pairs of) A, a length $l_{ij} \in \mathbb{R}$ is associated with every $(i, j) \in F \cup Z(A)$. An instance of the job-shop scheduling problem is thus represented by a triple (D, l, c), where $D = D(N, F, A)$ is a disjunctive graph, l a weight vector and $c : \mathbb{R}^{n+1} \to \mathbb{R}$ a cost function.

A *selection* $S \subseteq Z(A)$ is a set of arcs obtained from A by choosing at most one arc from each pair. The selection is *complete* if exactly one arc from each pair is chosen. Every selection S of $D(N, F, A)$ naturally defines a new disjunctive graph $D[S] = (N, F_S, A_S)$, where $F_S = F \cup S$, while A_S is obtained from A by removing the pairs containing the arcs in S. We call $D[S]$ an *extension* of D under S. Finally, we associate with $D(N, F, A)$ the weighted directed graph $G(D) = (N, F)$, with length l_{ij} associated with every $(i, j) \in F$.

With every instance $(D(N, F, A), l, c)$ of the job-shop scheduling problem, with c convex and piecewise linear, we associate the convex program $(SCH(D, l, c))$, obtained from Problem (2.3) by dropping all of the disjunctive constraints. Denoting by $z^*(D, l, c)$ the optimum value of $(SCH(D, l, c))$, the original disjunctive problem (2.3) can be restated as the problem of finding a complete selection $\bar{S}$ of A such that $z^*(D[\bar{S}], l, c)$ is minimum. Also, $z^*(D, l, c)$ provides a lower bound for the optimum solution value to $SCH(\bar{D}, l, c)$, where $\bar{D}$ is any extension of D.

2.1 Solution algorithm and lower bound computation

An instance of the (m-RTC) is solved by our algorithm by enumerating all of the feasible routings $R \in \mathcal{R}(M, T)$ and by solving, for each R, the associated instance (D^R, l, c) of the job-shop scheduling problem (2.3). This task is carried out by implicitly enumerating all of the feasible extensions of D^R. However, the enumeration of the (partial) extensions of D can be limited by the following standard arguments. Let UB be any upper bound to the

optimum solution value of Problem (2.3) - e.g., the cost $c(\hat{t})$ of any known feasible solution $\hat{t}$ - and let S be a (partial) selection of A. If the optimum solution value $z^*(D[S], l, c)$ to $SCH(D[S], l, c)$ satisfies $z^*(D[S], l, c) \geq UB$ then no (complete) extension of $D[S]$ can improve on $\hat{t}$ and the problem can be disregarded. Now, Problem $SCH(D[S], l, c)$ is an instance of the so called *optimal potential problem* with convex costs, which can be shown to be the dual of a min-cost flow problem with convex costs and can be solved efficiently even in its integer version ([1]). Since a lower bound computation must be carried out at each branching, we studied a further relaxation to $SCH(D, l, c)$ which proved to be effective in reducing the size of the enumeration tree with very little computational effort.

Let $D(N, F, A)$ be a disjunctive graph, with $|N| \geq 1$, and suppose $G(D)$ does not contain a positive dicycle. Denote by l^*_{ij} the length of a maximum path from $i \in N$ to $j \in N$ in $G(D)$ ($l^*_{ij} = -\infty$ if no ij-path exists). Let $SCH(D, l) \subseteq R^{n+1}$ be the feasible region of $SCH(D, l, c)$. Since we assume $G(D)$ contains no positive dicycle, then $SCH(D, l) \neq \emptyset$; also, $l^*_{ij} < \infty$ for all $i, j \in N$. In what follows, we denote by t_W the sub-vector of $t \in SCH(D, l)$ indexed by W and by $proj_W(D, l)$ the *projection* of $SCH(D, l)$ onto the t_W-space, that is $\tilde{t} \in proj_W(D, l)$ iff there exists $\hat{t} \in SCH(D, l)$ such that $\hat{t}_W = \tilde{t}$.

▶ **Lemma 1.** [10] *Let $W \subseteq N$, with $W \neq \emptyset$. Then*

$$proj_W(D, l) = \{t \in I\!R^{|W|} : t_j - t_i \geq l^*_{ij}, i, j \in W\} \tag{3}$$

So, let $(D(N, F, A), l, c)$ be an instance of Problem (2.3) and let $W = \{d(j) : j = 1, \ldots, |T|\} \cup \{0\}$ be the set of nodes of $G(D)$ corresponding to the exit operations (one for each train in T) and to the start. The projection $proj_W(D, l)$ can be written as:

$$SCH_s(D, l) = \begin{cases} s_j - s_i \geq l^*_{d(i),d(j)} & i, j \in T \\ l^*_{0,d(j)} \leq s_j \leq -l^*_{d(j),0} & j \in T \\ s \in I\!R^{|T|}, \end{cases}$$

where, as before, $s_j = t_{d(j)} - t_0$, $j \in T$. Observe that we have $z^*(D, l, c) = \min\{c(t) : t \in SCH(D, l)\} = \min\{c'(s) : t \in SCH(D, l), s_j = t_{d(j)} - t_0, j \in T\} = \min\{c'(s), s \in SCH_s(D, l)\}$. Also, since node 0 corresponds to the start operation, and we have assumed $t_0 \leq t_i$ for all $i \in N$, then we have $l^*_{0,d(j)} \geq 0$, for $j \in T$. Finally, since $G(D)$ does not contain positive dicycles, we have $-l^*_{d(j),0} \geq l^*_{0,d(j)}$, for $j \in T$.

An optimum solution to $SCH(D, l, c)$ can be obtained by finding an optimum solution s^* to $\min\{c'(s), s \in SCH_s(D, l)\}$, and then "lifting" s^* to a solution $t^* \in SCH(D, l)$: the last task can be carried out by a simple maximum path tree computation. In this way problem $SCH(D, l, c)$ is reduced to an equivalent problem with much fewer variables.

Now, if we let $l_j = l^*_{0,d(j)}$ and $u_j = -l^*_{d(j),0}$, for $j = 1, \ldots |T|$, and we let $q_1 = l_1$ and $q_j = l^*_{d(j-1),d(j)}$, for $j = 2, \ldots, |T|$, then the following convex program $REL(D, l, c)$ provides a relaxation to $SCH(D, l, c)$:

$$\begin{aligned} LB(D, l, c) = \min & \sum_{j \in T} g_j(s_j) + \sum_{j \in T} f_j(s_j - s_{j-1}) \\ s.t. \quad & s_j - s_{j-1} \geq q_j \quad j \in T \\ & l_j \leq s_j \leq u_j \quad j \in T \\ & s \in I\!R^{|T|+1} \end{aligned} \qquad (REL(D, l, c))$$

where again s_0 denotes the departure time of the last departed train. In fact, the feasible region of $REL(D, l, c)$ is obtained from $SCH_s(D, l)$ by dropping some of the defining constraints. Thus, the optimum $LB(D, l, c)$ to $REL(D, l, c)$ provides a lower bound to the optimum solution value associated with every (complete) selection of $D(N, F, A)$. Observe that $REL(D, l, c)$ has only $|T|$ decision variables (the departure times) and few constraints, again corresponding to the constraints of the dual of a min-cost flow problem. In what follows we assume $q_j \geq 0$ for all $j \in T$: this condition is ensured by the *no interchange stipulation* on train departures which imposes $s_j \geq s_{j-1}$ for all $j \in T$, and which in turn is imposed by including the corresponding precedence constraints into Problem 2.3. In [10] we show how to reduce the above problem into a convex min-cost flow on a small network, which in turn can be solved very efficiently (see [7]).

2.2 Computational results

In order to evaluate the overall approach to the (RTC), we performed both static and run-time (real-life) tests (see [10]). Static tests involve a single trains list, and were carried out mainly for assessing the quality of the relaxations and of the branch and bound algorithm. The results clearly show that, when compared to $SCH(D, l, c)$, solving the min-cost flow reformulation of $REL(D, l, c)$ speeds the computing times up to 2.5 times, a very desirable feature for real-time applications. Indeed, an instance of $REL(D, l, c)$, particularly in its min-cost flow reformulation, can be solved (by using the Goldberg and Tarjan code [7]) much more efficiently than the original $SCH(D, l, c)$ instance (solved by CPLEX 10.0); in contrast, the total number of branching nodes increases only slightly. The results become even more impressive when compared with other classical approaches, such as those based on time-indexed reformulations. Run time tests were performed to evaluate the ability of the system to manage real-time traffic and compare its performances to those obtained by human dispatchers and were done during an official test-campaign, which lasted several days. The results show that the dispatchers were in most cases outperformed by the system. This favorable comparison is confirmed by the average result, which shows an increase of more than 8% in a cumulative measure agreed with the ATM engineers (see [10]).

3 Single-track railways traffic control

In this section we briefly describe the basic elements of an optimization based automatic route system for single-track railways. The system here described is already partially in operation on several lines of the Italian railways, namely Parma – S. Zeno and Trento – Bassano in North Italy, Siracusa – Gela, extended to Trapani, Siracusa and Caltanisetta in South Italy and Terontola – Foligno in Central Italy. The full automatic route setting system, based on the optimal recalculation in real-time of the timetables will be put into operation by the end of year 2012.

A single-track line is a sequence of stations joined by a unique track. The track segment between two stations is called *block*. Blocks are sometimes partitioned into sections, and, for safety reasons, trains running in a same direction will be separated by a minimum number of such sections. For brevity we neglect sections in the remainder of the paper but the extension to such case is immediate.

Trains running in opposite directions or trains running in the same direction but at different speeds may need to cross each other somewhere in the line. Of course this can only happen in a station: the exact time and the meeting station is established by the official timetable. However, due to unpredictable delays, it may become impossible or simply

disadvantageous to accord with the official timetable, and new meeting stations should be detected. In most railway systems this is performed manually by human dispatchers, sometimes with the help of a supervising control software which can identify and present possible meeting points, typically according to some local optimal criteria. In contrast, the system we are developing will be able to fully control the traffic along single-track railway lines, by establishing optimal meeting stations and actually controlling train movements in stations and on the tracks between stations. It is important to remark here that, due to very rigid routing schemes, the routes followed by trains in stations can be considered as fixed (as they only depend on the arrival sequence).

Let $S = \{1, \dots, q\}$ be the set of stations and let $B = \{1, \dots, q-1\}$ be the set of blocks, with block i between station i and station $i+1$. Whereas at most one train can occupy the block between two stations, each station $s \in S$ can accommodate up to u_s trains, where u_s is the station *capacity*[1].

Let $R = S \cup B$ the set of railway resources and let T be the set of trains. Any train $i \in T$ runs through a sequence of stations and blocks. So, the route of the train i may be represented by a path $P^i = \{v^i_1, (v^i_1, v^i_2) \dots, (v^i_{l(i)-1}, v^i_{l(i)}), v^i_{l(i)}\}$ where node $v^i_k \in R$ for $1 \le k \le l(i)$ is the k-th railway resource used by i. We denote by V^i (A^i) the set of nodes (arcs) of P^i. The arcs of P^i represent precedence constraints, i.e. the fact that (the resource corresponding to) node v^i_k is visited by the train before node v^i_{k+1} on its route. With each arc $(v^i_k, v^i_{k+1}) \in A^i$ we associate the weight $W^i_{k,k+1} \ge 0$ representing the minimum time necessary to train i to move from the k-th resource to the next. Thus, if v^i_k is a station (node), then $W^i_{k,k+1}$ is the time the train should spend in the station before departing. If v^i_k is a block (node), then $W^i_{k,k+1}$ is the time necessary to reach next station.

Next, we construct the *routes graph* $G^T = (V, A)$ by letting $V = \{r\} \cup \{v \in V^i : i \in T\}$, that is V contains all nodes associated with the train routes plus an additional node r (the *root* of G^T); and $A = \{(r, v^i_1), i \in T\} \cup \{(u, v) \in A^i : i \in T\}$. So the new node r is a source, connected to the first node of each train route P^i. Also, for $i \in T$, we associate with arc (r, v^i_1) the weight W_{ri} which represents the expected number of seconds (from "now") before the train is expected to start its route ($W_{ri} = 0$ if the train is already in the line). In practice, node r represents a common start, which is associated with the origin of the time.

Now, for each node $v = v^i_k$ in $V - \{r\}$ let us denote by $t_v = t^i_k$ the minimum time in which train i can reach the k-th resource on its path. Also, we let $t_r = 0$. Observe that by definition each arc $(u, v) \in A$ with weight W_{uv} represents the constraint $t_v \ge t_u + W_{uv}$. Indeed, if $v = v^i_k$ is a station, then t_v represents the minimum *arrival time* for train i in such station. Similarly, if $v = v^i_k$ is a block b, then t_v represents the minimum time for train i to enter such block or, equivalently, the *departing time* from the station which precedes block b on P^i. Moreover, if such a block b follows station s in P^i and the official departure time from s of train i is D^i_s, then we add the arc (r, v) with weight D^i_s, representing the constraint $t_v \ge t_r + D^i_s = D^i_s$.

For all $v \in V$, the quantity t_v can be computed by a longest-path tree computation on G^T with weights W and root r. The vector $t \in R^V_+$ is called *schedule* or *actual timetable*. The schedule t approximates the behavior of the trains along the line. However, we need to take into account other precedence constraints in order to correctly predict the actual train timetable. In fact, for some pair of trains i and j we need to impose that they meet in a station s of the railway (we include a fictitious station to represent trains meeting outside

[1] The model can be easily extended to the case in which some trains can not be accommodated on sone given platforms

the line). We show now how to model the effect of such decision on the schedule t by adding a suitable set of arcs A_s^{ij} to G^T. The new schedule is then computed by calculating the longest path tree on the resulting graph. We distinguish two cases: i and j travel in opposite directions or they travel in the same direction.

Case 1. Train i and train j, travelling in opposite directions, meet in station s. Clearly, s belongs to both P^i and P^j. So, let v_k^i and v_m^j be the nodes corresponding to station s on P^i and P^j, respectively. Since i and j meet in s, then j leaves s after i enters in s, that is $t_{m+1}^j \geq t_k^i$. Similarly, i leaves s after j enters s, that is $t_{k+1}^i \geq t_m^j$. This is represented by adding the arcs $A_s^{ij} = \{(v_k^i, v_{m+1}^j), (v_m^j, v_{k+1}^i)\}$ with weight 0 to the graph G^T. Observe that these arcs ensure that i and j will not conflict on a block in the resulting schedule, since trains i and j enter the station from opposite directions (and thus they cannot conflict before they enter) and they exit in opposite directions (and they cannot conflict after they meet).

Case 2. Train i and train j, travelling in the same direction, meet in station s. This may be necessary if, for example, a train should catch up and overtake another train. This case is a bit more complicated because, for safety reasons, two trains can never be on the same block, even if running in the same direction. So, again let v_k^i and v_m^j be the nodes corresponding to station s on P^i and P^j, respectively. Let us assume that i precedes j before reaching station s, and follows j afterwards. This means that, for every station s' preceding or coinciding with s on P^i, train i must enter s' before train j has entered the block which immediately precedes s' on both routes (if such block belongs to P^j). This fact can be represented by adding suitable arcs to G^D as shown in the previous case. The roles of i and j are interchanged after station s, and for every station s'' following s on P^j, train j must arrive in s'' before train i has entered the block which immediately precedes s'' on both routes (if such block belongs to P^i).

Evaluating the actual timetable

As for the (m-RTC), also for the *Single-track Railway Traffic Control Problem (s-RTC)* the quality of the actual timetable depends on its conformity to the official timetable. Again, we suppose that such quality is evaluated by a convex piece-wise linear cost function c_v for each $v \in V$, and the cost of the schedule t is compiuted as $c(t) = \sum_{v \in V} c_v(t_v)$.

3.1 A MILP formulation for the real-time traffic control problem in single-track railways

If two trains can possibly meet on the line, they form a *crossing train pair*. In principle, *all* pair of trains can meet on the line, even if, according to the official timetable or a current prediction, they are not supposed to do it. However, by simple heuristic considerations, many such pairs can be excluded in advance. In what follows, the set of possibly crossing train pairs will be denoted by $K = \{\{i, j\} : i \in T, j \in T, i \text{ and } j \text{ crossing}\}$. For every $\{i, j\} \in K$, we let $S(ij)$ be the set of stations where i and j can actually meet - including, when possible, the fictitious station representing the out-line. For every $\{i, j\} \in K$ and every $s \in S(ij)$, we introduce a binary variable y_s^{ij}, with $y_s^{ij} = 1$ if and only if i and j meet in s. Denote by $G(y)$ the graph obtained from G^T by including the arcs of A_s^{ij} when $y_s^{ij} = 1$, for all $\{i, j\} \in K$, $s \in S$. Let $t(y)$ be the schedule obtained by a maximum path-tree computation on $G(y)$. Then the (s-RTC) problem amounts to finding a binary vector y

such that (i) every crossing pair of trains meet in a station, (ii) the stations capacity is not violated and (iii) the cost $c(t(y))$ is minimized.

The following is a mathematical formulation for the (s-RTC):

$$
\begin{array}{lll}
\min & \sum_{v \in V} c_v(t_v) & \\
s.t. & & \\
(i) & t_v - t_u \geq W_{uv} & (u,v) \in A \\
(ii) & t_v - t_u \geq M(1 - y_s^{ij}) & \{i,j\} \in K, s \in S(ij), (u,v) \in A_s^{ij} \\
(iii) & \sum_{s \in S(ij)} y_s^{ij} = 1 & \{i,j\} \in K \\
(iv) & y, t \text{ satisfying capacity } u_s & s \in S \\
(v) & t \in \mathbb{R}^V, y_s^{ij} \in \{0,1\}, \{i,j\} \in K, s \in S(ij) &
\end{array} \tag{4}
$$

where M is a large suitable constant. Let $(\bar{t}, \bar{y})$ satisfying all constraints but the capacity constrains (iv): $\bar{y}$ is called a *meet-point assignment*. Clearly, checking if a meet-point assignment is also satisfying all capacity constraints is an easy task, and we will come back on this later. The above formulation can be strengthen in various ways, but we do not get into detail here. We instead show how to represent constraint (iv) by introducing suitable variables and/or linear inequalities. In [11] and [14] we investigated two alternative approaches. The first is a natural consequence of the definition, but may contain an exponential number of constraints; the second is a compact, flow based representation of station capacities.

A non-compact formulation for station capacity constraints

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

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

$$
\sum_{i,j \in C, i<j} y_s^{ij} <= \frac{1}{2}(u_s + 1)u_s - 1 \tag{5}
$$

for all $C \subseteq T$ with $|C| = u_s$. The number of inequalities of type (5) grows exponentially with u_s. However, in the single-track lines considered, u_s is almost always ≤ 4. And yet, their number becomes quite large even for a small number of trains. For this reason we resort to the classical row generation approach, which amounts in starting with a small subset inequalities and generating new ones only if necessary.

A compact, flow based representation of station capacity constraints

Let us first fix a meet-point assignment $\bar{y}$. For any train $j \in T$, let $Su(j, s, \bar{y})$ be the set of *successors* of j in station s, that is the set of trains $i \in T$ which enters s after j leaves the station. Remark that since the meet-point assignment is given, $Su(j, s, \bar{y})$ is known for all $j \in T$ and $s \in S$ (if s is visited by j). Now, we can think at station platforms as unit resources that can be supplied to trains. Then a train i can receive the platform either

"directly" from the station s, or from a train j such that $i \in Su(j, s, \bar{y})$, which received the platform at an earlier stage. Then the assignment $\bar{y}$ is feasible if every train receives the required platform, as we will show more formally in the sequel. We represent this feasibility problem as a network flow problem, where the nodes are associated with the station s and with the trains.

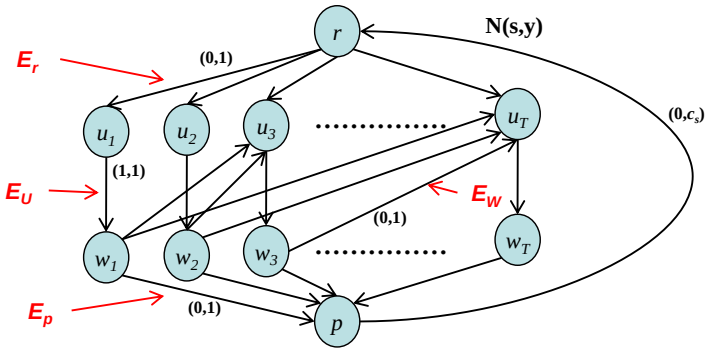

Figure 2 The support network

We focus now on a given station s. To simplify the notation, we assume that every train in T will go through s. Since both s and $\bar{y}$ are fixed, we let $Su(j, s, \bar{y}) = Su(j)$. Also, we assume that trains are ordered by their arrival times in station s. So, $j \in Su(i)$ implies $j > i$.

Let us introduce a support graph $N(s, \bar{y}) = (\{r, p\} \cup U \cup W, E)$, where $U = \{u_1, \ldots, u_{|T|}\}$, $W = \{w_1, \ldots, w_{|T|}\}$. Let the arc set be $E = E_r \cup E_U \cup E_W \cup E_p \cup \{(p, r)\}$, where $E_r = \{(r, u) : u \in U\}$, $E_U = \{(u_j, w_j) : j \in T\}$, $E_W = \{w_i \in W, u_j \in U, j \in Su(i)\}$, $E_p = \{(w, p) : w \in W\}$. With each arc $e \in E$ we associate lower bound l_e and upper bound f_e. Namely, $l_e = 0$ and $f_e = 1$ for $e \in E_r \cup E_W \cup E_p$. Then $l_e = f_e = 1$ for $e \in E_U$ and finally $l_{pr} = 0$, $f_{pr} = c_s$.

We have the following

▶ **Theorem 2.** *[11] The assignment $\bar{y}$ is feasible w.r.t. the station capacity constraints if and only if, for every $s \in S$, the graph $N(s, \bar{y})$ has a circulation satisfying all lower and upper bounds.*

The proof is based on Hoffman's circulation theorem and can be found in [11].

The above result can be used to model the station capacity constraint into our MILP program. To this end, we introduce two binary variables x_s^{ij}, x_s^{ji} for all stations $s \in S$ and all train pairs $\{i, j\} \in K$, with the interpretation that $x_s^{lm} = 1$ if and only if $m \in Su(l, s, y)$. Observe that x can be easily obtained from y by an affine transformation. For example, consider two trains i and j, with $i < j$, and assume that i and j travel in opposite directions, with i running from station 1 to station n and j from station n to station 1. If i and j meet in station $1 \leq s \leq n$, then i precedes j in all stations before s and j precedes i in all stations after s. Thus, $x_s^{ij} = 1 - \sum_{z=1}^{s} y_s^{ij}$ and $x_s^{ji} = 1 - \sum_{z=s}^{n} y_s^{ij}$. Remark that $x_s^{ij} + x_s^{ji} = 1 - y_s^{ij}$. The case of trains running in the same direction is analogous.

Next, we need to represent, for each station $s \in S$, the network flow problem discussed above on the graph $N(s, y)$. This can be done by considering an extended flow network $\bar{N}$ obtained from N by letting $E_W = \{(w_i, u_j) : i \in T, j \in T\}$, leaving all other arc sets

unchanged. So, E_W contains all possible arcs from W to U. Observe that $\bar{N}$ is independent of x. However, to prevent sending flow on "forbidden" arcs, we will fix the upper capacity to 0 whenever $j \notin Su(i, s, y)$ (this in turn depends on x).

Next, we introduce a flow variable z_s^e for every arc of $\bar{N}$. Then we write the flow conservation constraints at the nodes of $\bar{N}$ and lower and upper bounds on the flow variables z_s^e. In particular, lower and upper bounds are defined as for $N(s, y)$ except than for the arcs in E_W. For such arcs we simply let $z_s^{w_i u_j} \leq x_s^{ij}$. In this way, the arc (w_i, u_j) can carry one unit of flow only if $x_s^{ij} = 1$, that is if $j \in Su(i, s, y)$.

3.2 Preliminary implementations and comparisons

The current implementation of the optimization algorithm is rather basic and much can (and will) be done to make it more efficient. In the current version, e.g., the violated constraints of the non-compact formulation are generated only when 0,1 solutions are found by the solver (CPLEX 12.2) (the separation is done by looking at maximal cliques in interval graphs). The CPLEX default parameters setting is used, and so, for example, no particular branching scheme is implemented. Nevertheless, both approaches are able to solve real-life instances corresponding to a 14-hour time window on the Trento-Bassano line, with 30 trains and 23 stations. The non-compact formulation behaved slightly better, namely it took 5.30 sec. of computing time against 7.35 sec for the compact formulation (with an Intel Core i7 870 2.93GHz under Red Hat Enterprise Linux Client release 5.7). Keep in mind that these results are obtained by running a very preliminary implementation. Nevertheless, they prove that, at least for a line of the size of the Trento-Bassano, the (s-RTC) problem can be solved to optimality within the time required by the application.

References

1 Ahuja, R.K., D.S. Hochbaum, J.B. Orlin, *A cut-based algorithm for the nonlinear dual of the minimum cost network flow problem*, Algorithmica 39 (2004) pp. 189–208.

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

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

4 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.

5 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.

6 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.

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

8 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

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

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

11 C. Mannino, T. Nygreen *Compact VS non-compact MILP formulations for Railway Traffic Control in Single-track lines*, working paper, University of Oslo, 2011

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

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

14 T. Nygreen *Real-time Railway Traffic Control in Single-track lines*, master Thesis, University of Oslo, in preparation (October 2011)