Final verison available at SIAM : https://doi.org/10.1137/1.9781611974683.ch6

Integer Programming Techniques for Train Dispatching in Mass Transit and Main Line

Leonardo Lamorgese [*] Carlo Mannino [*] Mauro Piacentini [†]

Abstract

Trains moving in railway systems are often affected by delays or cancellations. This in turn may produce knock-on effects and propagate to other trains and other regions of the network. These undesired effects may be alleviated by suitably re-routing and re-scheduling trains in real-time. Train dispatching is thus a central task in managing railway systems as it allows recovering from undesirable deviations from the timetable, and a better exploitation of railway resources. With few exceptions, dispatching is still almost entirely in the hands of human operators, despite the problem amounts to solving a complex and large optimization problem which does not lend itself to be handled manually. In this chapter we describe how integer programming can be exploited to quickly find optimal solutions to large dispatching problems, and describe real-life implementations of these ideas. The first of these was a fully automated system put in operation in some terminal stations in Milano Underground in 2007. An official test campaign showed how the system was able to improve the performances of dispatchers in terms of train punctuality and regularity. Main Line applications have also followed in the years, from small single-track regional lines to more complex multiple-track lines and large stations.

1 Train dispatching

Railway transportation represents a significant share of the overall transportation sector accounting for 11% of freight and 9% of passenger transport (per kilometer) [17]. Perceived as the "greenest" transportation mode, railway is expected to play an increasing role in future communications. Indeed, railway operators over the world are renewing rolling stock, infrastructure, signalling systems and their organization. This in turn is demanding more research in managing and better exploiting the different components of railway systems. Optimization models have been studied and proposed for, e.g., personnel and crew scheduling, maintenance operations, train composition, timetable creation, dispatching, etc. Integer programming, in particular, has often been exploited to model and solve the associated railway optimization problems. Such models are generally developed to provide middle or long term plans, and are based on predictions on the number of passengers, personnel availability, trains speed, etc. However, in daily operations, disturbances or disruptions may occur, generating deviations from the planned or expected behaviour. As a

[*]SINTEF ICT, Oslo, e-mail: leonardo.lamorgese@sintef.no, carlo.mannino@sintef.no
[†]University of Rome, La Sapienza, e-mail: piacentini@dis.uniroma1.it

consequence, actions must be taken to alleviate the effects of such deviations. A first, possible approach is to take uncertainty into consideration already at a planning stage, by utilizing *robust* or *stochastic* models [4]. Another, complementary way is to carry out real-time replanning decisions when such deviations occur (for a very recent survey on rescheduling in railway operations see [7]). *Train dispatching* may be viewed as a series of replanning actions, carried out in real-time when running trains deviate from the official timetable. In the following we outline today's dispatching process in railway systems (see, e.g. [12]).

Railway systems can be classified as *Mass Transit* (typically urban systems such as subways) and *Main Line*. In either case, a railway is basically an intricate network of interconnected tracks. In the current *fixed block* signaling systems, tracks are segmented into *block sections*. Each block section can accommodate at most one train and is always preceded by a traffic light: incoming trains must stop at red light. In railway networks we may identify regions (sub-networks) with specific features and roles. *Stations* are complex sub-networks where trains can perform various operations (*services*), such as embark or alight passengers, reverse direction, etc. Sections in stations are classified as *stopping points* (where train can actually stop, such as platforms) or *interlocking routes*, which are sequences of track segments connecting stopping points. The schematic representation of a small terminal station is given in Figure 1. In the scheme, stopping points are identified by a green label. In the figure, a possible train route is represented by the directed path going through stopping points $En, P1, S2, P3, Ex$. Each arc of the path corresponds to an interlocking route. So the train enters the station from stopping point En, runs the interlocking route from En to $P1$ (which is a platform) where it disembarks passengers, continues through the next interlocking route to stopping point $S2$ where it reverses direction, and then proceeds along the remaining part of the route to exit from exit point Ex.

Finally, we refer to tracks connecting distinct stations as *links*. Links are often *single-track*, that is trains must use the same physical connection in both directions. Other relevant sub-networks are *stops* (along links) where trains can embark and alight passengers, *junctions* where a track splits into two, equipped with *switches* to allow trains to switch between alternative tracks, and so forth.

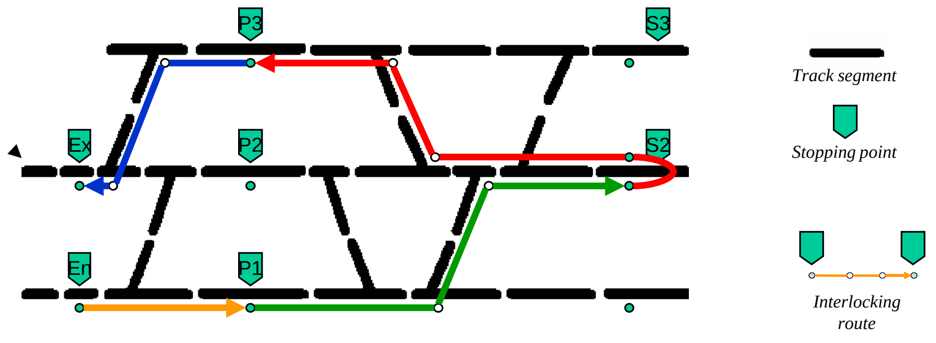

Figure 1: A subway terminal station

A train moving in the railway will run through a sequence of sections, called *route*. The *official timetable* defines, for each train, when it should pass through specific sections of its route, as a platform in a station (i.e. arrival) or the exit point (departure). Timetables may also specify the entire route schedule. Generating good and robust timetables is a hard task, and an entire line of optimization research is devoted to it ([8]).

The task of controlling trains running through a railway network is carried out by experienced operators, *dispatchers*, with support by software tools and remote equipments called *Train Management Systems* (TMSs). The railway network is typically subdivided into small control areas, each assigned to a dispatcher. Dispatchers are constantly updated about the status of the network, the position of trains, their speed, the status of signals and switches, etc. When trains are running according to the official timetable, their role is limited to ensuring that signals and switches are set according to the plan; actually, in many modern route-setting systems, this task is performed by the TMS, while dispatchers only have supervising responsibilities. On the contrary, when one or more trains are running late (or some disruption occurs on the line) dispatchers are required to intervene and apply recovery or control actions. These include: change of dwell times between trains; advising modified train speed; modifying train routes (e.g. by assigning a new platform in a station); changing train order at junctions; changing arrival or departure orders at stations; etc. To give an example, suppose train A is arriving at a station and, according to the timetable, will stop in platform 1. However, the platform is currently occupied by train B, which was supposed to leave the station before the arrival of A but is actually out of order and cannot release the platform. As a consequence, a dispatcher will establish a new platform for train A and possibly change the departure order of trains A and B. In a more complex setting, other trains may also be involved, forcing train A to wait at the signal before the station. Another typical situation occurs when stations 1 and 2 are connected by a single-track link, trains A and B run it in opposite directions with A running from 1 to 2, and A and B are supposed to meet in station 1. Now, assume B is running late. The dispatcher may decide to keep A waiting in station 1, or let B wait in station 2 until A reaches. Or, if B is very late, the dispatcher may even decide to let B wait in a station preceding station 2 on its route. All such control actions are taken in the attempt of reducing the effects of deviations on the future. Based on a general assessment of the situation, on the relative (often simply perceived) importance of the trains involved, on a rough evaluation of the consequences of different recourse actions, on personal experience, rules of thumb and operational rules, the dispatcher is presented with the daunting task of taking, in a few seconds, decisions which may impact the overall network for many hours to come. With some effort, it is possible to define the cost of a real-time train schedule as a function of its deviation from the official timetable. Such cost function may take into account delays of trains in stations (weighted by train relevance), loss of connections between trains, headway between successive trains, etc. It follows that the decisions taken by dispatchers may be interpreted as approximated solutions to a large optimization problem, which consists in finding a new feasible timetable of minimum cost. We refer to this problem as the *Train Dispatching Problem*.

Today's dispatching is supported in various ways by TMSs. Besides interfacing dispatcher decisions with the field, TMSs show the current status of the network in the dispatching area, the position of trains, the status of signals and switches. They are also able to predict train movements and identify potential conflicts in the use of resources. A *conflict* is the simultaneous occupation by two trains of a block section. As mentioned above, this must never occur, and, in general, it is prohibited by protocols and physical devices. However, when trains deviate from the official timetable, potential conflicts may arise. When identified, such conflicts are presented to dispatchers which take and implement recovery decisions. Today's TMSs are in general unable to take such decisions and implement them autonomously. In Section 4 we describe a few exceptions in operation in Europe, based on the modeling and algorithmic ideas presented in Section 2 and Section 3. Actually, according to two very recent studies [7, 25], there are no other examples currently in operation.

2 Basic MILP models for the Train Dispatching problem

In this section we limit ourselves to present the basic MILP models for the Train Dispatching problem. For sake of simplicity we consider the case in which train routes are fixed; extensions to include the routing problem may be found in the literature ([7]). In order to describe train movements we associate, with every section of a train's route, a continuous variable which represents the time the train enters the section. The vector t of entry times, for all trains and all sections in their route, is called *schedule*. Our target is to find a feasible schedule t^* which minimizes some cost function $c(t)$ (which in turn is the measure of the deviations from the official timetable earlier introduced). Next, we introduce the basic constraints which must be satisfied by any feasible schedule t. If l is the minimum running time of a train through a block section, t_u is the time the train enters the section and t_v is the time it enters the next section, then $t_v - t_u \geq l$ holds. Other temporal constraints such as release times and due dates may be represented by similar simple precedence constraints. Now, when the routes of two distinct trains, say trains i and j, share a block section, one of the two trains must exit the section before the other one enters it. So, if t_z and t_u represent the time in which train i enters and exits the section, and t_v and t_w are the times in which train j enters and exits the section, then either $t_z \geq t_w$ *or* $t_v \geq t_u$. This is referred to as a *disjunctive precedence constraint*, and is identified by two ordered pairs $(u, v), (w, z)$.

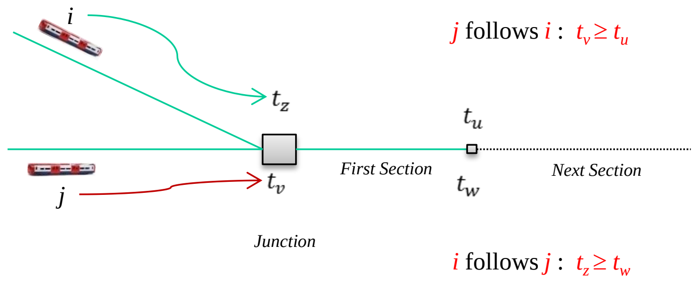

Figure 2: A junction, two incoming trains and associated scheduling variables

The number of such disjunctions may grow very large with the number of trains and the length of their routes in terms of block sections. Every feasible schedule t must satisfy one of the two precedence constraints for every disjunction. So, the dispatching problem may write as:

$$\begin{array}{lll}
\min & c(t) & \\
s.t. & & \\
(i) & t_v - t_u \geq l_{uv} & (u,v) \in A \\
(ii) & t_v - t_u \geq 0 \ \bigvee \ t_z - t_w \geq 0 & \{(u,v),(w,z)\} \in F \\
 & t \in \mathbb{R}^V &
\end{array} \tag{1}$$

where A, F, V are the set of indices of the simple precedence constraints, of the disjunctive precedence constraints and of the variables, respectively. *Disjunctive programs* [3] as the one above were introduced in the context of *job-shop scheduling* problems. Indeed, trains may be viewed as jobs performing certain *operations* (occupying a block section) on a set of machines (block sections). In particular, the dispatching problem belongs to the class of *blocking, no wait job-shop scheduling*, which is known to be hard to solve in theory and in practice [23].

Disjunctive programs may be represented and solved by means of mixed integer programming. Two major approaches compete in the literature: *Time-Indexed Formulations* and *big*-M *Formulations.* In big-M formulations, we associate with each disjunctive constraint $\{(u,v),(w,z)\} \in F$ a binary variable x_{uvwz}, with $x_{uvwz} = 1$ if $t_v - t_u \geq 0$ holds, and $x_{uvwz} = 0$ if $t_z - t_w \geq 0$ holds. In order to ensure this, we introduce a suitably large coefficient M (the notorious big-M) and we replace each disjunctive constraint (1.ii) with two (conjunctive) constraints

$$\begin{array}{lll}
(iia) & t_v - t_u \geq M(x_{uvwz} - 1) & \{(u,v),(w,z)\} \in F \\
(iib) & t_z - t_w \geq -M x_{uvwz} & \{(u,v),(w,z)\} \in F
\end{array} \tag{2}$$

It is easy to see that, when $x_{uvwz} = 1$, $(2.iia)$ reduces to $t_v - t_u \geq 0$ while $(2.iib)$, due to the large coefficient M, becomes redundant, as it is satisfied by any schedule t (satisfying all simple precedence constraints). Similarly, when $x_{uvwz} = 0$, $(2.iia)$ becomes redundant. It is well known that this kind of formulation tends to provide weak lower bounds. Computing strong bounds is important because they provide quality assessments when a solution to problem is at hand and they are crucial in limiting the search space in enumerative solution methods.

Time-indexed formulations for scheduling problems were introduced precisely to cope with this issue. The idea is to discretize the time horizon H into a finite number of periods, i.e. $H = \{1, 2, \ldots\}$, and replace each original continuous variable t_v with a set of $|H|$ binary variables, $y_{v1}, y_{v2}, \ldots$ with the interpretation that $y_{vq} = 1$ implies $t_v = q$. Time-indexed formulations produce better bounds than big-M formulations (see [16]) but, in general, at the cost of increased computational burden. The trade-off typically depends on the ability to limit the number of time-indexed variables, and to work with implicit representations of the overall formulation.

Both representations have been exploited in the literature on train dispatching. Time-Indexed formulations are exploited, e.g., in [9, 20], whereas big-M formulations are developed, e.g., in [2, 6, 13, 15, 25, 26, 27, 28, 29]. However, in our experience, big-M formulations appear to be more suitable for train dispatching, as time-indexed formulations tend to grow too large and become untractable for instances of practical relevance, mostly due to the difficulty in fixing a small time horizon for the scheduling variables.

We remark that a fundamental requirement of a dispatching system is that solutions must be found and returned in real-time or, more precisely, within the time limit set by the application.

Typically, this implies that time available for computation is limited to a few seconds. Now, a simple approach to solving a train dispatching instance could consist in generating the corresponding big-M formulation and then invoking a state-of-the-art MILP solver to solve it. Unfortunately, for real-life instances of practical interest, this naive approach tends to fail in finding within the time limit not only the optimal solution, but often even a feasible solution.

For this reason, almost all works presented in the literature resort to some type of heuristic, i.e. algorithms which search for feasible solutions but are unable to provide optimality certificates.[1] Instead, we decided to follow a different path and developed a decomposition approach which can return and certificate optimal solutions to the dispatching problem.

3 The decomposition principle in dispatching.

Decomposition is a natural way to tackle complex control or optimization problems. In train dispatching, this is how normal operations are carried out in practice. Indeed, even though the railway network is a huge connected graph, each dispatcher only controls a small portion of it, possibly coordinating with other dispatchers who control adjacent areas. The size of each control area is carefully determined so that one dispatcher suffices to manage it. Regional lines, for example, are often controlled by one or two dispatchers. Large stations are typically subdivided into smaller sub-stations, each controlled by a different dispatcher. Even if some communication with neighboring dispatchers is possible, a dispatcher has no authority on dispatching areas assigned to others, and can only inherit decisions without any possibility to affect them.

Another example of decomposition in practical dispatching is the way a dispatcher controls the assigned stretch of regional line. In a broad representation, such railway regions may be viewed as a sequence of stations connected by links. When potential conflicts are identified by the TMS, the dispatcher in charge decides where trains should meet or pass each other on the line (e.g. a station or a double-track link). In taking this decision, a station is considered as an aggregated resource with a given capacity. Detailed decisions about the sequencing of trains in the station are taken in a second phase, as trains approach the station.

Remarkably, this practical decomposition has a counterpart in MILP approaches to train dispatching. The original railway network is again decomposed into sub-networks each with its own controlled stations and links. We associate with each sub-network a corresponding dispatching problem and solve it, possibly to optimality. Then, the solutions to each sub-problem must be recomposed into a unique solution for the overall original problem. Different implementations of the decomposition principle mainly differ in the way sub-problems are generated, in the solution technique, in the way solutions of subproblems are recomposed, and in how the case in which sub-solutions cannot be recomposed is handled.

One version of the decomposition paradigm is the so called *macroscopic/microscopic* approach [5] recently introduced in the optimization literature for train timetabling and dispatching. At a microscopic level, all block sections are considered. At a macroscopic level, every sub-network, such as a station or a dispatching area, is represented in each train route as one section. Indeed, at this macro level, for each train, only (tentative) arrival and departure time at and from each sub-network are actually computed. In the standard approach, a good (possibly optimal) tentative macro-solution is found first. Then the macro-solution is turned into a global solution by extending

[1]The literature on heuristics is very large, and we refer to the comprehensive survey [7] for further details.

it to the sub-regions. If this attempt fails, the initial macro-solution is modified and the scheme iterated. This approach is followed, e.g. in [14] where the macro problem is modeled as a MILP, and where micro problems are solved by means of heuristics. In [15] both macro and micro problems are modeled as MILPs, but again combination and communication between levels is implemented by heuristics. Similar decomposition ideas are also exploited in [11], where the macro problem is obtained by shrinking sub-networks corresponding to dispatching areas.

Next, we will show how the macro-micro approach, and in general the decomposition principle in dispatching, has a counterpart in a classical method in linear and integer optimization, namely *Benders* decomposition. We will actually refer to a generalization of the classical Benders' decomposition to cope with integer sub-problems, as presented in [10, 30]. For simplicity's sake, we will consider the case of a single railway line and let the sub-networks in our decomposition be the stations of such line. However, the scheme may apply also to different hierarchies. If we let t be the scheduling vector, and x be the binary variables corresponding to precedence decisions, then the overall MILP may write as[2]:

$$\begin{array}{llll} \min c(t) & & & \\ s.t. & & & \\ (i) & Ax_L + & Bt & \geq b, \\ (ii) & & Dt + Ex_S & \geq q \\ (iii) & & t \text{ real}, \quad x_L, x_S \text{ binary} & \end{array} \tag{3}$$

The decision vector x has been written as $x = (x_L, x_S)$ to distinguish between decisions associated with stations (x_S) and decisions associated with tracks between stations (x_L). Correspondingly, we may identify the two blocks (i) and (ii) in Program (3). In many relevant practical contexts, as in the real-life implementations of Section 4,the two blocks "communicate" with each other through a small subset of t variables. This happens when decisions taken in the line regions and in the station regions do not"directly" affect each other, but only through their consequences on the scheduling[3]. The idea in Benders' decomposition is to solve a restricted problem (the *master*) where block (3.ii) is dropped. Let (x_L^*, t^*) be the optimal solution to the restricted problem. Then, if t^* can be extended to a a feasible solution (t^*, x_S^*) to (3.ii) (*slave* problem), we are done as (x_L^*, t^*, x_S^*) is an optimal solution to (3). Otherwise (x_L^*, t^*) cannot be extended to a feasible solution for the whole problem, and we identify an inequality $q^T x_L + r^T t \leq k$ which is satisfied by all of the feasible solutions to (3) but violated by (x_L^*, t^*). Such inequality is added to the master problem and the process is iterated. In our experience with real-life dispatching instances, the number of iterations is quite small. A nice feature of the approach is that the slave problem further decomposes into a number of independent sub-problems, one for each station.

We developed this approach in [18, 19, 21], enhancing the algorithm with delayed row and column generation [1]. This allowed us to solve to optimality a number of real-life instances from very different application contexts, as described in the next section.

[2]We assume here that the cost function $c(t)$ is linear.

[3]The situation is different when routing decisions for the line region may affect routing decisions in the stations. Consider for instance the case where a train can choose different routes in the line, involving different entry points in a given station, which in turn may force different routing decisions in the station

4 Real-life Implementations

Systems based on the ideas sketched in this chapter (described in more detail in [18, 19, 21]) are already in operation on several lines in Europe. Also, new implementations are in progress or scheduled in the near future. In particular, we will briefly describe real-life implementations in Italy, Norway and Latvia. Furthermore, we will mention an application for a large station in Italy and a Mass Transit system in operation in 2007-09 in some terminal stations of the Milan underground.

All these TMSs are embedded with our optimization algorithms, either directly (Italy, Latvia) or in a collaborative framework (Norway). Although most of the input is acquired remotely, dispatchers may interact with the system by manually providing further information (e.g. train delays or cancellations, fixing dispatching decisions, network disruptions, etc.). The optimization algorithm will then return new dispatching decisions. Depending on local operative rules, such decisions may be: (1) presented to dispatchers for validation; (2) forwarded directly to the field through remote equipment. The systems deployed in Italy and Norway fall in case (1), while both (1) and (2) apply to Latvia since freight trains are fully controlled by the system whereas decisions regarding passenger trains require human validation. Remarkably, statistics show that, even when the system does not control trains directly, dispatchers follow its suggestions in almost all cases. Another important question regards the"degree of freedom" of the algorithm. For instance, the Italian and Latvian railway operators have specific rules for resolving conflicts, which limit the potential impact of the algorithm as some dispatching solutions are forbidden.

Regional Italian Lines. A first deployment on Main Line of a TMS embedded with our dispatching algorithms was carried out in Italy in 2011, on a minor, single track regional line: Trento - Bassano del Grappa line (23 stops or stations). Since then, the tool has been extended to other lines in Southern, Central and Northern Italy. Table 1 reports some data on these lines.

Regional Line	**Stations/Stops**	**Network Complexity**	**Station Link**
Trento - Bassano	23	Single Line	Single Track
Parma - San Zeno	17	Single Line	Single Track
Foligno	53	Interconnected Lines	Single and Double Tracks
Milano - Mortara	12	Single Line	Single Track
Sicilia	111	Interconnected Lines	Single Track

Table 1: Regional Lines in Italy

The algorithm identifies alternative solutions, which are ranked according to their cost and presented to dispatchers: in practice the first solution in the ranking is the one often chosen. Since validation is left to the dispatchers, the tool is equipped with an exact procedure for detecting whether dispatching decisions lead to deadlock situations.

We have tried to quantify the benefit of relaxing current operative rules and allowing a larger solution space. In particular, we compared restricted and full solution space optimization on a test set of instances from the O-T-F line over a week in January 2013. In this setting, an instance represents the status of network and trains at a given moment in time. In Table 2 we cluster trains in three groups according to their computed delay at final destination.

Solution Space	**delay $\leq$ 5**	**delay$\leq$ 10**	**10<delay$\leq$15**
Restricted	90%	91%	93%
Full	95%	98%	99%

Table 2: Average delay distribution of instances of the $O-T-F$ line over a week in January 2013. The time unit is minutes. Mean number of controlled trains 86, standard deviation 27. Average number of trains running late is 9.

Results show an increase in the number of trains on time or with little delay (+5%), with virtually all trains (99%) arriving at destination with at most 15 minutes delay. On the other hand, restricted solution space solutions averaged 7% of trains arriving at destination with more than 15 minutes delay. In Table 3 we present some computational results and algorithmic information regarding a day of experiments in the above mentioned week in January 2013. On this day, the average number of controlled trains on this line was 107, with the highest number throughout the day being 130 and the lowest 67, while the average number of trains simultaneously on the line was 10. Results are aggregated in five time ranges (four-hours each).

Periods	**Instances**	**Iterations**	**Time (s)**	**Conflicts**	
				Potential	Solved
[04:08]	982	9	3.99	13777	154
[08:12]	941	9	4.58	11639	151
[12:16]	1127	9	3.85	7056	126
[16:20]	1433	5	4.02	3164	49
[20:24]	1232	5	4.57	3617	61

Table 3: Algorithmic information

Column "Periods" represents the time range (hh-hh), column "Instances" is the number of distinct instances solved within that time range, column "Iterations" expresses the *average* number of macro-iterations (i.e. master-slave) of the algorithm, column "Time" shows the *average* computation time (in seconds) and finally the two sub-columns under "Conflicts" express the *average* number of potential conflicts and those solved by the algorithm, respectively. The information regarding how many conflicts are solved in practice with respect to their potential number is expressed to give a measure of the importance of using a delayed column generation approach. Assuming (roughly speaking) that each conflict is expressed in the model by an integer variable, the delayed approach saves from adding around 98% of such integer variables on average, which in turn brings considerable computational benefits.

Although the current implementation represents already an important step towards improved and automatic railway management, these figures show that further improvements may be obtained by applying new regulations and an effective, exact optimization algorithm. This was allowed for the first time in a recent implementation in Norway.

Norwegian Lines. A pioneering dispatching system which exploits the decomposition approach described in the previous section has been tested on some regional lines in Norway (Trondheim-Dombås, Stavanger - Moi) and operated on one of these (Stavanger-Moi). The novelty of such system lies in the possibility of exploring the full solution space, allowing a complete exploitation of the exact algorithm. The system was released in February 2014, displaying the real-time optimized train graph[4] on a screen in the dispatching central. Each time the network status changes (train delayed, deviation etc), a new solution is computed and the graph is updated accordingly. The dispatcher may then decide whether to accept the suggestion or to discard it. At the time of writing, the use of system was on hold until the release of the tender for renovating Norway's entire signalling system, as the system is considered a candidate competitor by the network operator.

We now give some information regarding the Stavager - Moi line. The railway stretching from Moi to Stavanger is 123 km long and visits 16 stations. Every 12 hours, up to and, in some cases, over 100 trains run this line (on average). In particular, around 40% of this traffic is exclusive to the (entirely double-track) portion connecting Stavanger and Sandnes (stations which are more or less 15 km and 5 stops apart), while the remaining traffic also passes this area. As a consequence, this portion of the network is particularly dense and presents a challenge for local dispatchers.

Freight Lines in Latvia. TMSs equipped with our optimization algorithms are currently[5] under commissioning on an extended railway network in the East of Latvia. The overall network is composed by several lines: Daugavpils-Eglaine, Daugavpils-Krustpils, Rezekne-Krustpils, Zilupe-Krustpils, Karzava - Rezekne. In total, there are 52 stations, with 10 communication points and 8 station gates. These lines are characterized by high traffic volumes and are mainly used for freight transportation. The TMS's dispatching decisions regarding freight trains are automatically forwarded to the field, without requiring a dispatcher's validation. Dispatchers accordingly will only focus on solving conflicts involving passenger trains, where, as described above, they will be presented with the best solutions identified by the algorithm.

A large station in Italy. TMSs also prove to be very useful for monitoring and controlling traffic in large stations. Monfalcone (in Northern Italy) is being provided with one such automatic TMS, able to re-route and re-schedule trains in an optimal way with respect to the timetable and the current network situation. The system is scheduled to start commissioning as of March 2016. The system in Monfalcone is required to control three connecting "satellite" stations: Monfalcone Station (14 stopping points and 59 interlocking-routes), Ronchi Nord Station (7 stopping points and 16 interlocking-route), Ronchi Sud Station (7 stopping points and 16 interlocking-routes) and a communication point.

[4]A train graph is a standard graphical representation of a timetable.

[5]October 2015

Mass Transit To our knowledge the first fully automatic dispatching tool operated in some terminal stations of Milano Underground System from 2007 to 2009 [22], managed by ATM (the major Italian municipal transport company). Our optimization algorithm was embedded into the TMS developed by Bombardier Transportation, a large multinational of the transport sector. Prior to its activation, an extensive test campaign was carried out to compare the performances of the system against those of dispatchers in charge at one of the terminal stations. Such direct comparisons are very rare in the literature and in practice, and quite complicated to set up. The difficulty stems from the impossibility for the system and the dispatchers to have control over station and trains at the same time and compare them on exactly the same data input. To get around this issue, during the test campaign 8 pairs of 1-hour time slots with equivalent traffic patterns were identified by ATM engineers. For each of the eight pairs, one of the two time slots was assigned to the automatic system, whereas the other was assigned to dispatchers. Two objective functions were considered: the first measured deviations from the timetable; the second measured deviations in regularity, namely in the difference between actual train headways and desired ones. Final results showed that the system was able to improve both objectives by an average of 8% over the dispatchers, despite the relatively small size of the terminal stations.

5 Concluding remarks

The literature abounds with models and solution algorithms for real-time train traffic management - train dispatching and other related problems such as delay management - but presents very few applications. This shortcoming can be attributed in part to the operators' resistance to innovation, but also, in some measure, to the first attempts in using optimization methods in practice not delivering the bounty that had been "promised" to the industry (e.g. increased efficiency, lower costs). However, as shown in this chapter, the landscape is slowly shifting. Applications of optimization to traffic management are now starting to appear around Europe. Operators are increasingly aware of the potential of optimization-based traffic management, as proven in recent tenders, by explicitly requesting "intelligent" dispatching functionalities for new TMSs. In our tests and real-life applications, we show that much improvement over the current practice can be achieved by using optimization techniques, and mathematical programming in particular. The limitation with our exact approach is that some of our assumptions do not apply to every railway line. Therefore, work has to be done to extend this approach with a more general decomposition scheme.

References

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

[2] M. Aronsson, M. Bohlin, P. Kreuger, *MILP formulations of cumulative constraints for railway scheduling - A comparative study*, Proceedings of 9th Workshop on Algorithmic Approaches for Transportation Modeling (ATMOS), 2009.

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

[4] A. Ben-Tal , L. El Ghaoui and A. Nemirovski , Robust optimization methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

[5] R. Borndrfer, B. Erol, T. Graffagnino, T. Schlechte, E. Swarat *Aggregation Methods for Railway Networks* ZIB-Report 10-23 (November 2010)

[6] M. Boccia, C. Mannino, I. Vasiliev, The dispatching problem on multitrack territories: Heuristic approaches based on mixed integer linear programming, *Networks*, 62 (4), pp. 315–326, 2013

[7] V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf, J. Wagenaar, *An overview of recovery models and algorithms for real-time railway rescheduling*, Transportation Research Part B, 63, pp. 15–37, 2014.

[8] V. Cacchiani, P. Toth, *Nominal and robust train timetabling problems*, European Journal of Operational Research, 219, pp 727–737, 2012.

[9] G. Caimi, M. Fuchsberger, M. Laumanns, M. Lüthi, *A model predictive control approach for discrete-time rescheduling in complex central railway station areas*, Computers & Operations Research

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

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

[12] A. D'Ariano, *Improving Real-Time Train Dispatching: Models, Algorithms and Applications*, TRAIL Thesis Series no. T2008/6, The Netherlands TRAIL Research School, 2008.

[13] A. D'Ariano, D. Pacciarelli, M. Pranzo, *A branch and bound algorithm for scheduling trains in a railway network*, European Journal of Operational research, 183, pp. 643–657, 2007.

[14] T. Dollevoet, F. Corman, A. D'Ariano, D. Huisman, *An iterative optimization framework for delay management and train scheduling*, No EI 2012-10, Econometric Institute Report from Erasmus University Rotterdam, Econometric Institute, 2012.

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

[16] M. Dyer., 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.

[17] European Commission, *EU transport in figures*, Statistical Pocketbook 2013, http://ec.europa.eu/transport/facts-fundings/statistics/doc/2013/pocketbook2013.pdf

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

[19] L. Lamorgese, C. Mannino, *The track formulation for the Train Dispatching problem*, Electronic Notes in Discrete Mathematics, 41, pp. 559–566, 2013

[20] R. Lusby, J. Larsen, M. Ehrgott, D. Ryan, *A set packing inspired method for real-time junction train routing*,Computers & Operations Research 40, pp. 713–724, 2013.

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

[22] C. Mannino, A. Mascis, *Optimal Real-Time traffic control in metro stations*, Operations Research, 57 (4), pp. 10261039, 2009.

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

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

[25] P. Pellegrini, G. Marlire, J. Rodriguez, *Optimal train routing and scheduling for managing traffic perturbations in complex junctions*, Transportation Research Part B, 59, pp. 58–80, 2014.

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

[27] K. Sato, K. Tamura, N. Tomii, *A MIP-based timetable rescheduling formulation and algorithm minimizing further inconvenience to passengers*, Journal of Rail Transport Planning & Management 3, pp 38-53, 2013.

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

[29] J. Törnquist Krasemann, *Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances*, Transportation Research Part C 20, pp. 62-78, 2012.

[30] F. Vanderbeck, L. A. Wolsey, Reformulation and Decomposition of Integer Programs, in *50 Years Of Integer Programming*, Springer, 2010