An optimized, automatic TMS in operations in Roma Tiburtina and Monfalcone stations

S. Foglietta, G. Leo, C. Mannino, P. Perticaroli & M. Piacentini
Optrail, Italy

Abstract

A remarkable number of works on automatic train dispatching have recently appeared in the scientific literature, many of which are also exploiting some exact or heuristic optimization. Despite of this interest, very few automatic dispatching systems are actually in operation in main line or mass transit networks, mostly devoted to simple tasks in small lines. In this work we describe a recent implementation of a real time dispatching system monitoring and controlling trains in two large stations, namely Roma Tiburtina and Monfalcone. The system exploits optimization in order to take and implement crucial dispatching decisions. In particular, the algorithm finds suitable routes and schedules for the movements of trains in the station, so as to minimize costs and delays. The resulting trains movement is conflict-free, and respects safety and specific traffic rules. Also, it computes, exploits and provides accurate traffic forecast information based on the trains and railway real time status. A heuristic and an exact optimization algorithm, the latter based on Mixed Integer Linear Programming, are developed and run independently to introduce redundancy and increase safety. Computational experience on real-life instances shows that both methods are able to provide either good or optimal solutions in a very short time (less than a second). A first release of the system has been in operation since February 2014; the final release is scheduled to be operational by December 2014.
Keywords: automatic railway traffic control, real-time trains dispatching, interlocking, optimization algorithms.

1 Introduction

Nowadays, the increasing demand on passengers and freight transport represents a great opportunity of gain and development for the railway. Actually, railway

WIT Transactions on The Built Environment, Vol 135, © 2014 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/CR140531

can be considered one of the best way to move goods and people all over European countries, because of the low costs, high (nominal) regularity and punctuality as well as the minimal environmental impact, in comparison with other transportation systems. Nevertheless, because of the financial and economical situation in Europe, huge investments on the railway infrastructure are not any longer possible. Therefore, in order to catch the opportunities given by such promising traffic transport trends, it is essential to improve the utilization of the current railways system, both on the network infrastructure and the rolling stocks.

An example of problem with wide room for improvements is given by the management of complex stations (and junctions), which represents highly interconnected areas of the railways network with of big demand of transit and services. In this context, the main challenge is to optimally route and schedule timetabled trains, ensuring quality requirements and operational constraints. This task corresponds to dynamically allocate the tracks and platforms available at the station. Restriction and complexity on the problem are given by the scarcity of the network resources respect to the regular traffic volumes. So far, routing trains is managed by human dispatchers, leading to sub-optimal utilization of the track capacity and an increasing conjection in the rest of the network. As opposed, better result could be reached by developing a simple support system to human operators for an optimal management of the strongly congested parts of the railway network (such as complex stations). A further step would be to completely automatized, as way to make the entire decision process less error-prone and more effective.

The routing problem can arise at different levels of the planning hierarchy, *strategic*, *tactical* and *operational* levels. Differences come out in terms of time-horizon, objectives and degrees of freedom as well as accuracy.

The problem at the strategic level is characterized by a long time horizon and the final goal is to analyze the infrastructural capacity of a railway station as well as to highlight needs for additional railway resources, such as new tracks and platforms. This leads to assess limits of the network respect to increasing traffic volumes as well as the evaluation of new investment alternatives, i.e. the construction and/or modification of some parts of the existing infrastructure.

In the second level, the tactical planning, the time horizon reduces to mid-term and the infrastructure is considered fixed. In a top-down approach, the overall timetable of trains is chosen at macroscopic level, so that it should be assess the feasibility at the microscopic level. For a station, this problem leads to find routes and schedules of the timetabled trains, allowing or suggesting minor deviations on the arrival and departure times.

The last level is the *operational planning*, where the problem is to adjust the yearly timetable to deal with the daily disturbances. Late trains on arrival, accidents, infrastructure maintenance and disruption of the networks easily provide additional delays and make the tactical planning any more feasible. Adjustments on scheduling and also on routing are required in order to avoid conflicts and to minimize the overall deviation from the original plan. Although all these routing problems can be considered similar, because they roughly represent the same stuff, there are some differences that need to be taken into account, in order to

select the best model and algorithm in the three different cases. A key feature for the operational problem is that the environment is dynamic: the situation changes quickly, so that a response should be given as fast as possible. For this reason, respect to this feature, the operational is considered a *real-time* problem. As opposed, the strategic and tactical planning are considered *off-line*. Another difference is given by the corresponding dimensions: off-line models are usually bigger, because more complicated situations need to be evaluated, as opposed to real-time models where dimensions are much smaller because huge deviations from original plan are not allowed.

The routing (with an inner scheduling) problems at the station for the three different levels are usually in a top-down order, with the output of a phase being the input for the successive one. This makes all the three levels quite dependent between each other. From one side, the solution for the off-line planning has a big impact on the real-time problem, affecting system response performances in dealing daily troubles. Robust solution in the off-line step makes the system less dependent by small perturbations in the real-time, making the latter easier to be solved. On the other side, analysis at low levels can be important also at high ones: for example, to really evaluate some investment on the infrastructure, its performances should be analyzed also in terms tactical and operational responses.

This work focuses on models and algorithms for automatic train dispatching in large stations. More emphasis will be left for the real-time task as more demanding in terms of time and more useful in real application.

2 Problem description

This section is devoted to the description of all fundamental objects which play a central role in train dispatching operations in large stations. Finally, a definition of real-time train dispatching problem is introduced.

Stations. A station corresponds to a region of the railway network where trains can stop to perform tasks as embark and alight passengers, meet or pass other trains, do maintenance operations, etc. To our purpose, a station is composed by two sets of tracks: the first is given by the tracks, called *stopping points*, in which trains can stop to execute some operations; the second contains tracks, called *interlocking-routes*, which connect pairs of distinct stopping points. It is possible that two stopping point can be connected by parallel interlocking-routes. Stopping points and interlocking routes are called *station resources*.

In the physical configuration of a station, an interlocking route actually corresponds to a sequence of electrically controlled tracks called *track circuits*. So it may happen that, even if two interlocking routes do not share a stopping point, the two corresponding sequences of track circuits actually cross each other). Special stopping points are the *line points* where the station connects to the railway line, and *platforms* where passengers can be embarked or alighted. Line points typically act as *entry points* or *exit points*, depending on the direction of a train running through them.

Trains. Trains may cross the station, arriving from a distinct station and leaving to another station; they may originate in the station or they may end their run in the station. Trains enter the station from a line point and either (i) exit the station from a different line point or (ii) end their run on a platform. Entry and exit line points are given in the official timetable and can only be changed by an operator, so they are always input to the model. In case (ii) the train will magically disappear from the station after a given time it reaches the platform. Trains originating in a station start their travel from a platform.

Depending on relevance number of passengers, etc., trains are classified in categories which affect the cost function. Since our control is in real-time, a train can be either out of or in the station; in the latter case, it may be in a stopping point or in an interlocking route. If the train is approaching the station, we assume given the expected time of arrival to the entry line point. Otherwise, the train is occupying some station internal resource and we assume given the time when the train started the occupation of such resource.

Train routes. The alternating sequence of stopping points and interlocking-routes encountered by the train while traversing the station is called *train route*. As said, the origin of the route may be either a stopping point, for instance the entry line point or a platform (but also intermediate stopping points sometimes appearing in some stations), or an interlocking route. The destination is either an exit point or a platform. The train is assumed to occupy an interlocking route for a fixed time (the *running time*) known in advance; trains can stop in stopping points, and a minimum stopping time is also assumed and known.

A route P is feasible for a train i if the following conditions are satisfied: (i) P contains a platform, unless the train is already exiting the station; (ii) P is compliant with the station circulation rules and the service type associated to i; (iii) all resources belonging to P are available when they are expected to be occupied by i.

Train Schedules. The movements of a train i along its assigned route P are expressed by a set of time instants in which train i enters each station resource belonging to P. Given a route, a train schedule is the assignment of an entry time instant to each resource of the route. In particular, train i can enter an interlocking-route of P in a time instant greater or equal than the sum of the entry and stopping times associated to the previous stopping point of the resources sequence. Since trains cannot stop on interlocking-routes, the entry time instant of a stopping point has to be equal to sum of entry and traversing times associated to the previous interlocking route of the sequence. A schedule is feasible if it satisfies last two conditions.

Conflicts. Distinct trains traversing the station may want to access simultaneously incompatible station resources, such as the same platform, or two interlocking routes sharing a stopping point or simply crossing each other (recall that interlocking routes may contain several track circuits). Assume r_1 and r_2 are incom-

patible resources, and let i, j be two distinct trains, i traversing r_1 and j traversing r_2. Let t_i and t_j be the time i and j enters r_1 and r_2, respectively. Then, either $t_j \geq t_i + \delta(i, r_1)$ or $t_i \geq t_j + \delta(j, r_2)$. The quantity $\delta(q, r)$ is the *separation time* for train q and resource r, and can be infinite (e.g. when r_1 and r_2 represent the same platform). Solving a station conflict consists in deciding whether one resource of the corresponding conflict pair has to be occupied earlier than the other resource, that is *who goes first*.

Timetable. The *official* timetable is a document containing some wanted features of the movements of trains across the station. In particular, to each train, it assigns the following items: (i) entry point, (ii) arrival time, (iii) stopping platform, (iv) departure time, (v) exit point. Items (iv) and (v) are not assigned for trains terminating in the station; similarly, item (i) and (ii) are not assigned when the train is originating in the station. Arrival and departure times refer to the time in which the train is supposed to arrive and departure from the platform. Finally, even for trains which are simply traversing the station without stopping, a platform (iii) is always assigned. The official timetable actually can be even more specific, and for example specify the sequence of interlocking routes traversed by each train.

While entry and exit point cannot be modified by the decision process (typically, there is only one feasible entry point and one feasible exit point for a given train. Even when multiple entry and exit points are available, only human dispatchers may be allowed to change them), a new platform can be assigned to approaching trains only if the following conditions are satisfied. For each train i, a platform assignment which differs from timetable can be planned within a fixed time τ from the time instant in which train i is expected to enter the station. Moreover, besides satisfying availability and service type requirements, a new assignment can be decided if it has been requested that a conflict occurring outside the station authority area (i.e. on tracks connecting adjacent stations) has to be solved.

Cost function. The cost function measures the deviation from the official timetable. So, late arrivals and departures are penalized, as changes in the assigned platform. This cost also depends on the train category.

The real-time train dispatching problem (RTD). We are given the official timetable, a set of (controlled) trains, their current position in the station or their expected arrival times, the availability status of station resources, a cost function associated to timetable deviations. We want to find a feasible route assignment and a feasible schedule for each train such that no train leaves before its official timetable departure time, no conflict arises in the use of station resources and the overall cost is minimized.

3 The model

The RTD problem is naturally decomposed in two subproblems which are intrinsically related. First subproblem consists of enumerating all feasible

trains routings, according to conditions expressed in previous section. Second subproblem aims to find optimal conflict-free trains scheduling for each given routing, then it can be reformulated as a pure job shop scheduling problem. Analogous decomposition has been introduced by authors of [1] for trains traffic control problem in metro stations. In this section, we discuss this suitable decomposition for RTD problem.

3.1 Routing problem

We model the station by a digraph $D(N, A)$ with node set N and arc set A, such that N corresponds to the set of stopping points, whereas A corresponds to the set of interlocking routes. Let P_i be the train route assigned to train $i \in T$. We denote by $R(P_i)$ the set of railway resources required by P_i, either arcs in A or nodes in N. For each train i, P_i is the union of at most two dipaths belonging to $D(N, A)$, say P_i^{in} and P_i^{out}: P_i^{in} connects the pair of nodes associated to the entry line point and the platform assigned to i; P_i^{out} connects the pair of nodes corresponding to the assigned platform and the exit line point of i. If a train i terminates or starts its running in the station, P_i is given respectively by either P_i^{in} or P_i^{out}. A route is feasible for train i if it is consistent with the timetable movements required for i and each resource is available for railway traffic. Moreover, let observe that, in general, each train can admit more than one feasible route, since each pair of nodes associated to a line point and a platform could be connected by at least one path. Given a set of train T, a routing $\mathcal{P}$ is a set of feasible routes P_i assigned to each train $i \in T$. A routing $\mathcal{P}$ is feasible if it is compliant with real-time statuses information of station resources and it can satisfy railway dispatcher requests.

3.2 Job shop scheduling problem

For each $r \in R(P_i)$, let t_{ir} be the time in which train i enters resource r and let l_{ir} be the minimum time i occupies r. If r corresponds to an interlocking route, then l_{ir} is the running time of i through r. Otherwise, r corresponds to a stopping point and l_{ir} is the minimum stopping time.

For each pair of consecutive resources $r, q \in R(P_i)$, we have:

$$t_{iq} \geq t_{ir} + l_{ir} \tag{1}$$

with equality holding if r is an arc. The condition that train i cannot leave the platform before the official departure time is also modelled by one constraint of type (1).

Let $e, g \in A$ and let $i, j \in T$. Let assume train i traverses e and j traverses g, and i enters e before j enters g. Then, let denote by $h(i, e, g)$ the minimum time j can enter g after i has entered e. Thus, the following constraint has to be satisfied:

$$t_{jg} \geq t_{ie} + h(i, e, g) \tag{2}$$

Interlocking-routes e, g are said to be *incompatible* if $h(i, e, g) > 0$ for some $i \in T$; then, let C be the set of all unordered pairs $\{e, g\}$ of incompatible routes.

The vector t corresponds to the schedule for the trains in T. A schedule of train i corresponds to the subvector $t^i = (t_{i1}, t_{i2}, \dots)$. A train schedule t^i is feasible if t^i satisfies (1). A schedule t is feasible if t^i is feasible for each $i \in T$ and the following disjunctive pair of constraints:

$$(t_{jg} \geq t_{ie} + h(i,e,g)) \bigvee (t_{ie} \geq t_{jg} + h(j,g,e)) \tag{3}$$

is satisfied for each pair $i, j \in T$ and $\{e, g\} \in C$.

Disjunctive graph. The job shop scheduling problem has a useful combinatorial description expressed by *disjunctive graph* (for further details, we refer the reader to [2, 3]). Let $G(V, F, S, p)$ be a disjunctive graph such that: (i) V contains a node for each operation associated to the occupation of station resource by a train; moreover V contains a special node, called *sink*; (ii) F contains arcs representing the precedences between operations which are expressed by constraint (1); moreover F contains a directed arc from the sink to each node representing the occupation by a train of the first resource of its route; (iii) S is the set of disjunctive arcs pairs representing resources occupation conflicts, which are modelled by constraint (3); iv) p is a vector of weights associated to all arcs of disjunctive graph, which correspond to either running times for arcs in F or separation times for arcs in S.

A *selection* Q is a set of arcs obtained by choosing at most one arc from each pair in S. Given a selection Q, let S_Q be a subset of S obtained by removing the disjunctive pairs associated to arcs in Q. Then, the *extension* of G under Q is the disjunctive graph $G(V, F \cup Q, S_Q)$. A selection is called *complete* if exactly one arc is selected from each disjunctive pair. Moreover, a selection Q is *consistent* if digraph $G(V, F \cup Q, p)$ does not contain any strictly positive dicycle, i.e. a dicycle such that the sum of weights associated to its arcs is strictly positive. The well-known relation between scheduling problem and the disjunctive graph representation is based on the fact that each complete consistent selection of a disjunctive graph $G(V, F, S, p)$ is in bijection with a feasible schedule (e.g. see [4]). For a given selection Q, a *forcing* is an arc (i, j) belonging to a disjunctive pair $\{(i, j), (h, k)\} \in S_Q$ such that $G(V, F \cup Q \cup \{h, k\})$ contains a strictly positive dicycle. Finally, a *closure* of G under Q is the minimal forcing-free consistent extension of $G(V, F \cup Q, S_Q, p)$ with respect to $F \cup Q$.

4 Solution approaches

In this section we discuss two solution approaches to RTD problem which exploit the problem decomposition described in previous section. The first approach is an exact method based on Mixed Integer Linear Programming (MILP). The second approach consists of a fast heuristic algorithm based on disjunctive graph representation.

4.1 Exact method

An optimal solution to RTD problem is computed by retrieving the set of all feasible routings for trains in T, say $\mathfrak{P}$, then computing an optimal schedule $t^{\star}_{\mathcal{P}}$ for each $\mathcal{P} \in \mathfrak{P}$. In the following, we outline main phases of the method by omitting further implementation details.

Set $\mathfrak{P}$ is obtained by assigning to each train $i \in T$ a set of feasible platforms U_i which is consistent with timetable, real-time statuses information of station infrastructure and railway dispatcher authority requests. We assume that U_i is a non-empty set for each $i \in T$, since a platform assignment is even forced by human dispatcher for trains affected by no platform availability. Then, for each platform assignment, all feasible trains routes are computed by enumerating suitable paths of station graph $D(N, A)$, so $\mathfrak{P}$ is obtained.

For each $\mathcal{P} \in \mathfrak{P}$, we find an optimal schedule $t^{\star}_{\mathcal{P}}$ by formulating and solving a MILP problem, called $JSS(\mathcal{P})$. Given $\mathcal{P}$, $JSS(\mathcal{P})$ has continuous decision variables t which represent the time instants in which each train $i \in T$ starts the occupation of each station resource belonging to its route P_i. As discussed in Sec. 3.2, variables t have to satisfy constraints (1). Moreover, $JSS(\mathcal{P})$ is characterized by binary variables y which represent resources occupation conflicts. The use of binary variables y is crucial to define disjunctive constrains reported in (3). The objective function of the MILP formulation is given by the sum of delays introduced by scheduled arrival or departure of each train with respected to the arrival or departure times stated in the timetable. Since a complete formulation of $JSS(\mathcal{P})$ has a huge number of binary variables and disjunctive constraints, it cannot be solved in practice. However, we present a suitable separation algorithm which is practically cost-effective. For further details on optimization and separation techniques for integer programming, we refers the reader to [5, 6].

Data: $JSS'(\mathcal{P})$
Result: Optimal solution $(t, y)^{\star}$ to $JSS(\mathcal{P})$
$i \leftarrow 0$;
$JSS_i(\mathcal{P}) \leftarrow JSS'(\mathcal{P})$;
Compute an optimal solution $(t, y)^{\star}_i$ to $JSS_i(\mathcal{P})$;
while $(t, y)^{\star}_i$ *is not conflict-free* **do**
  compute disjunctive pairs of constrains $\{\pi_1, \ldots, \pi_k\}$ violating $(t, y)^{\star}_i$;
  $JSS_{i+1}(\mathcal{P}) \leftarrow JSS_i(\mathcal{P}) \cap \{\pi_1, dots, \pi_k\}$;
  $i \leftarrow i + 1$;
  Compute an optimal solution $(t, y)^{\star}_i$ to $JSS_i(\mathcal{P})$;
end
return $(t, y)^{\star}_i$;

Algorithm 1: Separation algorithm for $JSS(\mathcal{P})$.

The exact method algorthm is reported in Alg. 1. We initialize the separation algorithm by a partial description of $JSS(\mathcal{P})$, denoted by $JSS'(\mathcal{P})$, which

contains only continuous variables t and precedence constraints (1). Since $JSS'(\mathcal{P})$ is a relaxation of $JSS(\mathcal{P})$, if an optimal solution $(t,0)^\star$ to $JSS'(\mathcal{P})$ is feasible for $JSS(\mathcal{P})$, then the solution is also optimal for $JSS(\mathcal{P})$. Otherwise, we identify a set $\{\pi_1,\ldots,\pi_k\}$ of disjunctive pairs of constraints such that each π_j represents a station resource occupation conflict for a pair of trains. Since $\{\pi_1,\ldots,\pi_k\}$ violates solution $(t,0)^\star$, we obtain a strengthening relaxation of $JSS(\mathcal{P})$ by adding $\{\pi_1,\ldots,\pi_k\}$, so we solve the new relaxation and iterate the procedure until an optimal solution to $JSS(\mathcal{P})$ is found.

4.2 Heuristic method

The heuristic approach differs from the exact method (previously described) only for the scheduling phase. In particular, we propose a polynomial combinatorial algorithm which explain the disjunctive graph representation of the job shop scheduling problem, as discussed in Sec. 3.2, in order to obtain feasible trains schedules. Nevertheless the heuristic algorithm does not guarantee any approximation factor with respect to the optimal value, we will see in next section that it allows to retrieve a trains scheduling whose objective function value is very close to the optimum. In the following, we outline the heuristic algorithm procedure, which is reported in Alg. 2.

The heuristic method invokes two subprocedures, namely $closure$ and $schedule$. The $closure$ procedure allows to compute the closure of a disjunctive graph $G(V,F\cup Q,S_Q,p)$ such that the associated digraph $G(V,F\cup Q,p)$ does not contain strictly positive dicycle. This procedure is based on a recursive algorithm which, at each step, adds a forcing (if it exists) to arc set $F\cup Q$, then removes its corresponding disjunctive pair from S_Q. The procedure can identify forcing by a preliminary computation of the maximum distance between all pairs of nodes in V, which is carried out by Floyd-Warshall algorithm on digraph $G(V,F\cup Q,S_Q,-p)$ (e.g. see [7, 8]). For further details on $closure$ procedure, we refer the reader to [1]. Furthermore, $schedule$ procedure allows to compute the scheduling associated to digraph $G_i(V,F\cup Q_i,p)$ at each iteration i. Since $G_i(V,F\cup Q_i,p)$ is retrieved by a closure of a disjunctive graph (i.e. it does not contain any strictly positive dicycle), $schedule$ computes t_i by Dijkstra algorithm on digraph $G_i(V,F\cup Q_i,-p)$. Now, it is easy to check that the introduced heuristic algorithm is correct (a simple proof is given by induction on the number of iterations of Alg. 2).

The main idea of heuristic method is based on providing a feasible schedule by iteratively fixing conflict resolution decisions. In particular, at each iteration i, all resource occupation conflicts corresponding to disjunctive pairs of $G_i(V,F\cup Q_i,S_{Q_i},p)$ are identified. Then, a subset of conflict resolution decisions is fixed by according priority to a train $j^\star$ which introduces the minimal measure $a[j]$ of the expected perturbation behind the timetable. Each value $a[j]$ is computed by considering two cost components: (i) the first component is given by the overall delay increase if train j is assumed to have priority over the set of conflicts in which it is involved; (ii) the second component expresses the expected arrival order

Data: $G_{\mathcal{P}}(V, F, S, p)$, T, timetable
Result: feasible trains scheduling t
$i \leftarrow 0$;
$G_i(V, F \cup Q_i, S_{Q_i}, p) \leftarrow closure(G_{\mathcal{P}}(V, F, S, p))$;
$t_i \leftarrow schedule(G_i(V, F \cup Q_i, p)$;
foreach $h \in T$ **do**
  compute delay $d[h]$ behind timetable of h on $G(V, F \cup Q_i)$;
end
while $S_{Q_i} \neq \emptyset$ **do**
  sort trains in T by expected arrival order on assigned platform;
  foreach $j \in T$ **do**
    $a[j] \leftarrow 0$;
  end
  $L \leftarrow \emptyset$;
  foreach $\{e_h, e_k\} \in S_{Q_i}$ **do**
    $a[h] \leftarrow a[h] + \gamma \cdot h + p(e_h)$;
    $a[k] \leftarrow a[k] + \gamma \cdot k + p(e_k)$;
    $L \leftarrow L \cup \{h, k\}$
  end
  $j^\star \leftarrow \arg\min_{j \in L}\{a[j]\}$;
  foreach $\{e_h, e_k\} \in S_{Q_i}$ **do**
    if $h = j^\star$ *or* $k = j^\star$ **then**
      $F_{i+1} \leftarrow F_i \cup \{e_{j^\star}\}$;
      $S_{Q_{i+1}} \leftarrow S_{Q_i} \setminus \{e_h, e_k\}$;
    end
  end
  $i \leftarrow i + 1$;
  $G_i(V, F \cup Q_i, S_{Q_i}, p) \leftarrow closure(G_i(V, F \cup Q_i, S_{Q_i}, p))$;
  $t_i \leftarrow schedule(G_i(V, F \cup Q_i, p)$;
  foreach $h \in T$ **do**
    compute delay $d[h]$ behind timetable of h on $G(V, F \cup Q_i)$;
  end
end
return t_i;

Algorithm 2: Heuristic algorithm for $JSS(\mathcal{P})$.

of train j to its first station resource with respect to T. The relative importance between cost components can be tuned by suitable parameter $\gamma > 0$.

5 Computational experience

The computational experiences focuses on set of real-word instances which have been provided by Bombardier Transportation, Italy. The RTD problem instances

we have tested are based on real-time data of trains traffic information related to Tiburtina railway station of Rome (Italy), during day September 9th, 2013. The computational experience has been carried out on x86-64 GNU/Linux machine with 4 cores @800MHz and 8GB of RAM. Both solutions approaches have been implemented in C programming language. The exact method, based on integer programming, exploits IBM ILOG Cplex 12.5.1 MIP solver [9], then the separation algorithm has been implemented by Cplex Callable Library callbacks.

In order to test the quality of solutions and the computational performances of introduced methods, we consider 8 instances which represent different railway traffic statuses information of Tiburtina station, associated to different time instants of September 9th, 2013. In particular, each instance refers to a wide timetable composed by 135 trains that are characterized by different categories and service type functionalities. The planning time horizon associated to timetable covers 12 hours.

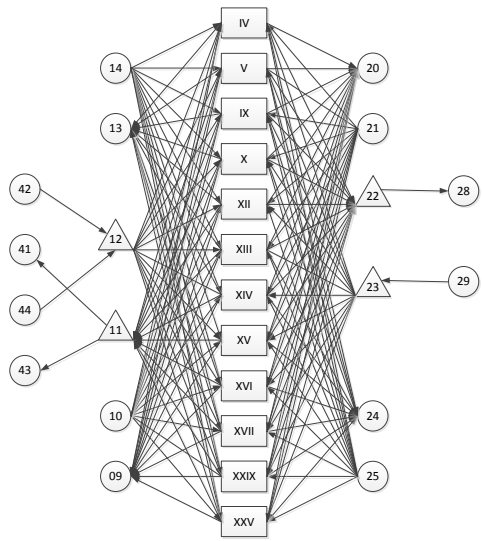

Figure 1: Digraph representing Tiburtina railway station of Rome (Italy).

Tiburtina railway station is composed by 30 stopping points and 62 interlocking routes (a representation of Tiburtina station graph is reported in Fig. 1). The set of stopping points is partitioned into three subsets which respectively contain line points, platforms and intermediate points (a special kind of stopping points which provide interconnections between other stopping points). In particular, Tiburtina station has 12 line points connecting itself to adjacent national railway lines. On the south side, it is connected to lines: *Direttissima* and *Lenta* (both toward Roma Termini station), *Alta Velocità* (toward Roma Prenestina station), *Indipendente* and

Table 1: Computational experience.

	Heuristic Algo		Exact Algo		
Instance	Val(sec)	Time(sec)	Opt(sec)	Time(sec)	Gap(%)
1	44416	0.02	44361	0.93	0.12
2	28260	0.01	28060	0.36	0.71
3	31800	0.02	31450	1.04	1.11
4	29560	0.01	29360	0.11	0.68
5	37060	0.02	36790	1.34	0.73
6	30560	0.02	30210	0.73	1.16
7	32860	0.01	32830	0.21	0.09
8	34226	0.02	34176	1.29	0.15

Locale (both toward Roma Casilina station). Moreover, on the north side, Tiburtina station is connected to lines: *Direttissima* (toward Roma Settebagni station), *Lenta* (toward Roma Salario station), *Merci* (toward Roma Settebagni station and Roma Smistamento cargo terminal station). Moreover, Tiburtina station offers a wide range of railway services, namely passengers transportation, freight transports and high speed rail services. All station objects of Tiburtina are managed by a computer-based interlocking system (the corresponding Italian acronym is ACC: "Apparato centrale computerizzato") provided by Bombardier Transportation Italy.

Table 1 summarizes computational results by reporting for each instance: (i) objective function value (second column) and computation time (third column), both expressed in seconds, for the heuristic method; (ii) optimal solution value (forth column) and computation time (fifth column), both expressed in seconds, for the exact method; (iii) relative gap (sixth column) of the heuristic method's solution with respect to the optimal solution.

Let us observe that our exact method allows to compute optimal solutions to real-life instances of the RTD problem given by a train traffic planning horizon of 12 hours and composed by with 135 trains, within a time lapse of 0.75 seconds on average. Same instances can be solved by our heuristic method within a time lapse of 20 milliseconds on average. Moreover, let observe that the heuristic method allows to compute good quality solutions, which differ from the optimal value for an average 0.6% ratio.

6 Conclusions

The RTD problem play a crucial role in automatic railway traffic control. In this work, we have introduced a suitable decomposition of the problem in two well-studied problems of Operations Research. Then, we have designed and exact

optimization algorithm based on integer programming and heuristic algorithm based on a combinatorial formulation of the problem.

We have implemented and tested our algorithms on real-life instances based on railway traffic information related to Tiburtina station of Rome. The computational experience shows that both methods are able to provide either good or optimal solutions within a time lapse which is less than one second. This experimental work takes part into a joint project of Optrail with Bombardier Transportation Italy, which aims to develop software tools for an efficient and effective real-time railway traffic management.

References

[1] Mannino, C. & Mascis, A., Optimal real-time traffic control in metro stations. *Oper Res*, **57(4)**, pp. 1026–1039, 2009.

[2] Balas, E., *Machine sequencing via disjunctive graphs: An implicit enumeration algorithm*. Defense Technical Information Center, 1969.

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

[4] Neumann, K., Schwindt, C. & Zimmermann, J., *Project Scheduling With Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling With Regular and Nonregular Objective Functions*. Lecture Notes in Economics and Mathematical Systems Series, Springer Verlag, 2002.

[5] Schrijver, A., *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc.: New York, NY, USA, 1986.

[6] Nemhauser, G.L. & Wolsey, L.A., *Integer and Combinatorial Optimization*. Wiley-Interscience: New York, NY, USA, 1988.

[7] Goldberg, A., Tardos, É. & Tarjan, R., *Network Flow Algorithms*. Number 1252 in Computer Science Department: Report STAN-CS, Department of Computer Science, Stanford University, 1989.

[8] Ahuja, R.K., Magnanti, T.L. & Orlin, J.B., *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1993.

[9] IBM, *IBM ILOG CPLEX Optimization Studio 12.5.1 - CPLEX User's Manual*, 2013.