Modeling and Solving the Train Timetabling Problem

Alberto Caprara*, Matteo Fischetti• and Paolo Toth*

* *DEIS, University of Bologna, Italy*

• *DEI, University of Padova, Italy*

Abstract

The *Train Timetabling Problem* (TTP) aims at determining a periodic timetable for a set of trains which does not violate track capacities and satisfies some operational constraints. In particular, we concentrate on the problem of a single, one-way track linking two major stations, with a number of intermediate stations in between. Each train connects two given stations along the track (possibly different from the two major stations) and may have to stop for a minimum time in some of the intermediate stations. Trains can overtake each other only in correspondence of an intermediate station, and a minimum time interval between two consecutive departures and arrivals of trains in each station is specified.

In this paper, we propose a graph theoretic formulation for TTP using a directed multigraph in which nodes correspond to departures/arrivals at a certain station at a given time instant. This formulation is used to derive an integer linear programming model which is relaxed in a Lagrangian way. A novel feature of our model is that the variables in the relaxed constraints are associated only with nodes (as opposed to arcs) of the aforementioned graph. This allows a considerable speed-up in the solution of the relaxation. The relaxation is embedded within a heuristic algorithm which makes extensive use of the dual information associated with the Lagrangian multipliers. We report extensive computational results on real-world instances provided from Ferrovie dello Stato SpA, the Italian railway company, and from Ansaldo Segnalamento Ferroviario SpA.

1 Introduction

The *Train Timetabling Problem* (TTP) aims at determining a periodic timetable for a set of trains which does not violate track capacities and satisfies some operational constraints. The trains have to be run every *period* of a given time horizon (e.g., every day of a time horizon lasting some months). Several variations of the problem can be considered, mainly depending on the objective function to be optimized and on the complexity of the underlying railway network. The problem addressed in this paper arose from a joint research with Ferrovie dello Stato SpA (the Italian railway company, FS for short) within the EU Esprit Project EUROPE-TRIS.

We consider a single, one-way track linking two major stations, with a number of intermediate stations in between. This case is particularly interesting since railway networks typically contain few important lines (called also "corridors") connecting major stations. On these corridors, made of two independent one-way tracks carrying traffic in opposite directions, track resource is limited due to great traffic densities. Once the timetable for the trains on the corridors has been determined, it is relatively easy to find a convenient timetable for the trains on the other lines of the network.

Times are discretized and expressed as integers from 1 to q, where q represents the length of the given period (e.g., minutes from 1 to 1440 for a period of 1 day). In the following, *time instants* indicate a particular instant within the given period, while *time intervals* indicate a continuous interval (not longer than q). Let $S = \{1, \ldots, s\}$ represent the set of stations, numbered according to the order in which they appear along the track. In particular, 1 and s denote the initial and final station, respectively. Analogously, let $T = \{1, \ldots, t\}$ denote the set of trains which are candidate to be run every period of the given time horizon. For each train $j \in T$, a starting station f_j and an ending station l_j ($l_j > f_j$) are given. Let $S^j := \{f_j, \ldots, l_j\} \subseteq S$ be the ordered set of stations visited by train j.

A timetable defines, for each train $j \in T$, the departure time from f_j, the arrival time at l_j, and the arrival and departure times for the intermediate stations $f_j + 1, \ldots, l_j - 1$. The timetable of each train is periodic, i.e, it is kept unchanged every period. The *running time* of train j in the timetable is the time elapsed between the departure of j from f_j and the arrival of j at l_j.

Each train is assigned an *ideal timetable*, which would be the most desirable timetable for the train, that may however be modified in order to satisfy the track capacity constraints. In particular, one is allowed to slow down each train with respect to its ideal timetable, and/or to increase the stopping time interval at the stations. Moreover, one can modify the departure time of each train from its first station, or even cancel the train. The final solution for TTP will be referred to as the *actual timetable*.

The *track capacity constraints* impose that overtaking between trains occurs only within a station. To this end, a train is allowed to stop in any intermediate station (even if the ideal timetable does not include a stop in that station) to give the possibility to some other train to overtake it. Furthermore, for each station $i \in S$, there are lower bounds a_i and d_i on the time interval between two consecutive arrivals and two consecutive departures, respectively. As the speed of a train on a track segment is assumed to be constant, this latter constraint implicitly imposes a time interval of at least $\max\{d_i, a_{i+1}\}$ between two consecutive trains in the track segment connecting stations i and $i+1$. Note that track capacity constraints, along with the fact that the actual timetable has to be repeated every period, may force some trains to be canceled to obtain a feasible solution.

Time windows constraints are also imposed, requiring that train j arrives in a station not later than a given time, or departs from a station not earlier than a given time. These constraints may help in guaranteeing convenient connections with trains running on other tracks.

In the formulation we consider, the objective is to maximize the sum of the profits of the scheduled trains, defined as follows. The *profit* achieved for each train j depends on the train *ideal profit* π_j, on the *shift* ν_j, defined as the absolute difference between the departure times from station f_j in the ideal and actual timetables, and on the *stretch* μ_j, defined as the (nonnegative) difference between the running times in the actual and ideal timetables. Formally, the profit for train j is expressed as

$$\pi_j - \phi_j(\nu_j) - \gamma_j \mu_j, \tag{1}$$

where $\phi_j(\cdot)$ is a user-defined nondecreasing function penalizing the train shift (with $\phi_j(0) = 0$), and γ_j is a given nonnegative parameter (i.e., the function penalizing the train stretch is assumed to be linear). Typically, the profit function is identical for trains of the same type (e.g., intercity trains) running in the same time interval of the period (e.g., between 7:00 AM and 9:00 AM for a period of 1 day). Of course, if the profit of train j turns out to be nonpositive, it is better not to schedule train j, i.e., to cancel it (this observation allows one to tighten the time windows for the train arrival and departure at each station).

The train timetabling problem has received considerable attention in the literature: see, for example, Szpigel [18], Petersen and Merchant [16], Jovanovic and Harker [12], Schrijver and Steenbeek [17], Cai and Goh [2], Harker and Hong [9], Carey and Lockwood [5], Higgins, Kozan and Ferreira [11], Brännlund, Lindberg, Nöu and Nilsson [1], Peeters and Kroon [15], Lindner and Zimmermann [14].

A problem similar to the one we consider in this paper is addressed in Brännlund, Lindberg, Nöu and Nilsson [1]. In this latter paper the authors discretize the time into 1-minute *time slots* and subdivide the track line into *blocks*. The track capacity constraints impose that two trains cannot be in the same block in the same time slot. There is a binary variable x_{sbj} each time the timetable constraints allow train j to be in block b in time slot s. This model is suited for small size instances only. Indeed, for the instances in our test bed, where the line is up to 500 kilometers long, blocks are about 1 kilometer long, and the number of trains to be daily scheduled is about 100, the model would require about $700,000$ capacity constraints and $1,500,000$ variables.

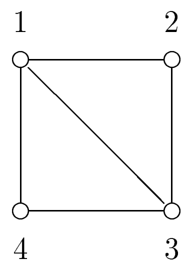
Although the problem is generally referred to as being NP-hard, almost every paper considers a different version of the problem. For this reason, we give an explicit NP-hardness proof for the specific problem we consider. Our proof will also show that any polynomial-time approximation algorithm with worst-case performance guarantee is hopeless for TTP unless P=NP (see, e.g., Kann and Panconesi [13]). The reduction is from the notoriously NP-hard *Max-Independent Set Problem* (MISP) (see, e.g., Garey and Johnson [8]) that, given an undirected graph $H = (N, E)$, where N is the node set and E the edge set, calls for a subset $S \subseteq N$ with maximum cardinality such that $(i, j) \notin E$ for all $i, j \in S$.

Proposition 1 *Any MISP instance can be polynomially transformed into a TTP one such that there is a one-to-one correspondence between MISP and TTP solutions, and corresponding solutions have the same value.*

Proof. Let $H = (N, E)$ be the graph defining the MISP instance, with $E = \{e_1, e_2, \ldots, e_m\}$, where $e_i = (j_i, k_i)$ for $i = 1, \ldots, m$. We define the corresponding TTP instance as follows. Each train corresponds to a node in N, i.e., $t = |N|$. We impose that each train is either scheduled according to its ideal timetable, or is canceled. In particular, for $j \in T$, we let $\pi_j := 1$, $\phi_j(0) := 0$ and $\phi_j(x) := 1$ for $x \neq 0$, and $\gamma_j := 1$. Note that, since times are integer, each timetable which is not identical to the ideal one will have either a shift or a stretch of at least one minute, and the associated profit will be nonpositive. The number of stations is $m + 1$ and we let $f_j := 1, l_j := m + 1$ for each train $j \in T$. Moreover, the minimum time intervals a_i and d_i between consecutive arrivals and departures from each station are both set to 1.

Ideal timetables are defined so that the ideal timetables of two trains j and k are *incompatible* (i.e., the two trains cannot be scheduled at the same time) if and only if $(j, k) \in E$. For each edge $e_i \in E$, we impose that, in the ideal timetable, trains j_i and k_i depart at the same time instant from station i and arrive at the same time instant at station $i + 1$. This is the only incompatibility within the track segment from i to $i + 1$ in the ideal timetables; see Figure 1 for an illustration.

More precisely, consider the first edge $e_1 = (j_1, k_1)$ of H. In order to make the ideal timetables of trains j_1 and k_1 incompatible, we set to j_1 the departure time from station 1 of both j_1 and k_1, whereas for each train $j \neq j_1, k_1$ the departure time from station 1 is set to j; all trains arrive at station 2 one minute after their departure from station 1. Similarly, the incompatibility associated with edge $e_2 = (j_2, k_2)$ is modeled by setting to $t + 1 + j_2$ the departure time from station 2 of both trains j_2 and k_2, whereas for all other trains j the departure time from station 2 is set to $t + 1 + j$; as before, all trains arrive at station 3 one minute after their departure from station 2. We iterate this construction



$H = (N, E)$

station

station		train 1	train 2	train 3	train 4
1	dep.	1	1	3	4
2	arr.	2	2	4	5
	dep.	6	7	6	9
3	arr.	7	8	7	10
	dep.	11	12	13	11
4	arr.	12	13	14	12
	dep.	16	17	17	19
5	arr.	17	18	18	20
	dep.	21	22	23	23
6	arr.	22	23	24	24

Figure 1: The transformation from MISP to TTP applied to instance $H = (N, E)$ with $N = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4)\}$.

for stations $3, 4, \ldots, m$ and for train pairs (j_3, k_3), (j_4, k_4), …, (j_m, k_m). More precisely, for $i = 1, \ldots, m$ we let the departure time from station i be equal to $(i-1)(t+1) + j_i$ for trains j_i and k_i and to $(i-1)(t+1) + j$ for $j \neq j_i, k_i$, and the arrival at station $i+1$ be one minute after the departure from i for all trains. This yields a TTP instance with the desired properties. $\square$

2 A Graph Theoretic Model

We next illustrate a mathematical formulation of the problem, calling for a maximum-profit set of paths in a multigraph.

Let $G = (V, A)$ be the directed acyclic multigraph defined as follows (see Figure 2 for an illustration). The node set V has the form $\{\sigma, \tau\} \cup (U^2 \cup \ldots \cup U^s) \cup (W^1 \cup \ldots \cup W^{s-1})$, where σ and τ are an *artificial source node* and an *artificial terminal node*, respectively, whereas sets U^i, $i \in S \setminus \{1\}$, and W^i, $i \in S \setminus \{s\}$, represent the set of time instants in which some train can arrive at and depart from station i, respectively. We call the nodes in $U^2 \cup \ldots \cup U^s$ and $W^1 \cup \ldots \cup W^{s-1}$ *arrival* and *departure* nodes, respectively. Let $\theta(v)$ be the time instant associated with a given node $v \in V$. Moreover, let $\Delta(u, v) := \theta(v) - \theta(u)$ if $\theta(v) \geq \theta(u)$, $\Delta(u, v) := \theta(v) - \theta(u) + q$ otherwise. Because of the "cyclic" nature of the time horizon, we say that node u *precedes* node v (i.e., $u \preceq v$) if $\Delta(v, u) \geq \Delta(u, v)$ (i.e, if the cyclic time interval between $\theta(v)$ and $\theta(u)$ is not smaller than the cyclic time interval between $\theta(u)$ and $\theta(v)$).

The arc set A is partitioned into sets $A^1, \ldots, A^t$, one for each train $j \in T$. In particular, for every train $j \in T$, A^j contains:

- a set of *starting* arcs of the form (σ, v), for each $v \in W^{f_j}$ corresponding to a feasible departure of train j from station f_j;
- a set of *station* arcs of the form (u, v), for each $u \in U^i$ and $v \in W^i$ such that $i \in S^j \setminus \{f_j, l_j\}$, the time window constraints allow an arrival (resp., departure) of train j at station i (resp., from station i) at $\theta(u)$ (resp., $\theta(v)$), and $\Delta(u, v)$ is at least equal to the minimum stop time of train j in station i (i.e., the stop time in station i in the ideal timetable);
- a set of *segment* arcs of the form (v, u), for each $v \in W^i$ and $u \in U^{i+1}$ such that $i \in S^j \setminus \{l_j\}$, the time window constraints allow a departure (resp., arrival) of train j from station i (resp., at station $i + 1$) at $\theta(v)$ (resp., $\theta(u)$), and $\Delta(v, u)$ is at least equal to the minimum travel time of train j from station i to station $i + 1$ (i.e., the travel time in the ideal timetable);
- a set of *ending* arcs of the form (u, τ), for each $u \in U^{l_j}$ corresponding to a feasible arrival of train j at station l_j.

By construction, G is acyclic and each path P^j from σ to τ in G which uses only arcs in A^j corresponds to a feasible timetable for train j (see Figure 2); in other words all the feasibility constraints pertaining train j are implicitly imposed in the definition of the structure of graph G.

In order to model the objective function, for each train j we associate a (possibly negative) *profit* p_a with each arc $a \in A^j$, defined as follows. For each starting arc of the form (σ, v), $v \in W^{f_j}$, let $\nu(v)$ be the *shift* associated with node v, i.e.,

$$\nu(v) := |\theta(v) - [\text{ideal departure time of train } j \text{ from station } f_j]|.$$

The profit of arc (σ, v) is set equal to $p_{(\sigma,v)} := \pi_j - \phi_j(\nu(v))$. This reflects the fact that the penalty for the shift of train j is uniquely determined by the first arc in the path for train j.

The profit of each station arc of the form (u, v), $u \in U^i, v \in W^i, i \in S^j \setminus \{f_j, l_j\}$, is set to $p_{(u,v)} := -\gamma_j \mu(u, v)$, where $\mu(u, v)$ is the stretch associated with arc (u, v), i.e.,

$$\mu(u, v) := \Delta(u, v) - [\text{ideal stop time for train } j \text{ in station } i].$$

Analogously, the profit of each segment arc of the form (v, u), $v \in W^i, u \in U^{i+1}, i \in S^j \backslash \{l_j\}$, is defined as $p_{(v,u)} := -\gamma_j \mu(u, v)$. where $\mu(v, u)$ is the stretch associated with arc $v, u)$, i.e.,

$$\mu(v, u) := \Delta(v, u) - [\text{ideal travel time for train } j \text{ from station } i \text{ to station } i + 1].$$

The profit of each ending arc of the form (u, τ), $u \in U^{l_j}$, is equal to zero.

The profit associated with an overall train timetable is given by the sum of the profits of the arcs in the paths of G corresponding to the scheduled trains.

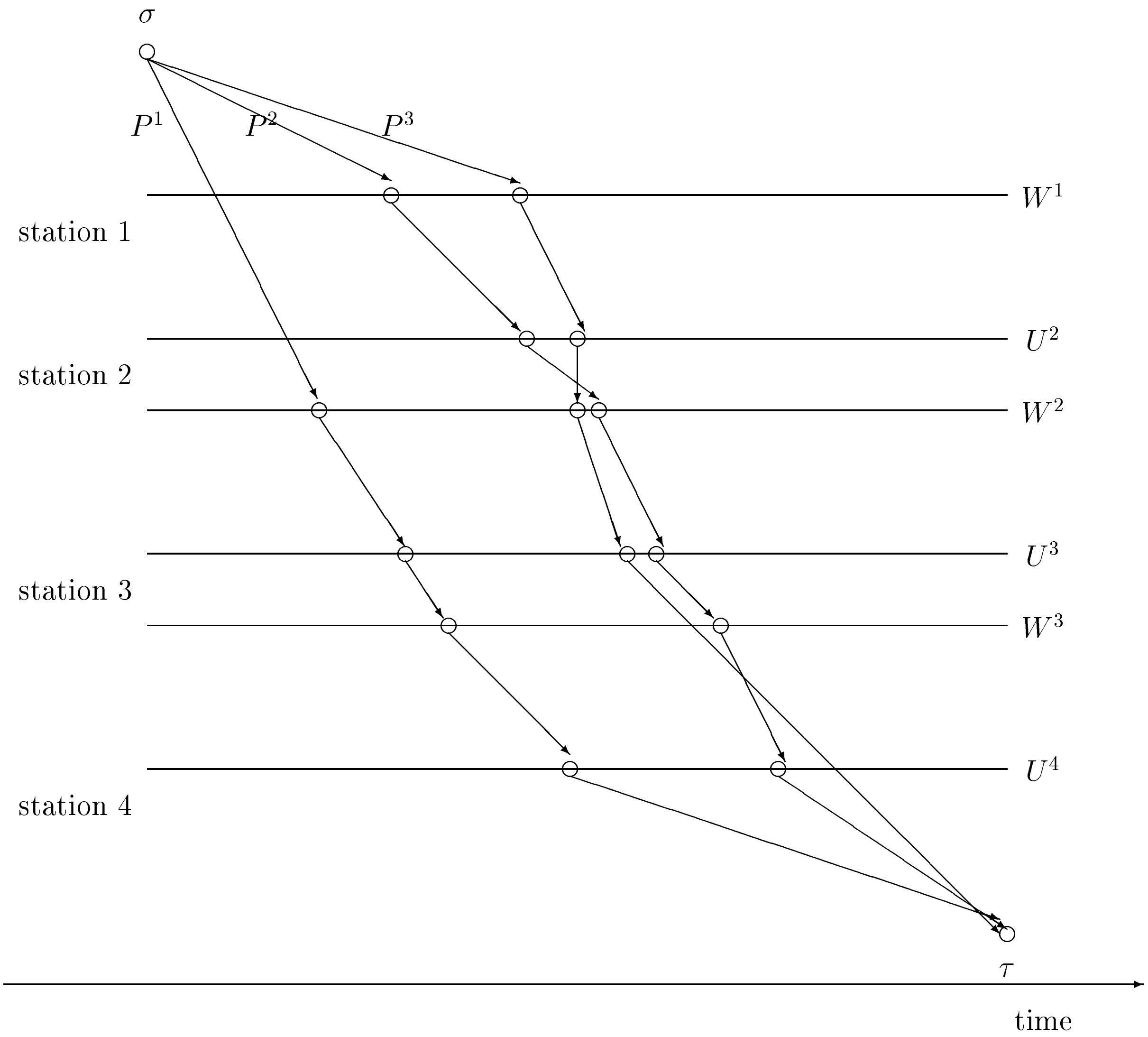


Figure 2: An example of train paths in graph G (with $s = 4$, $t = 3$, $f_1 = 2, l_1 = 4, f_2 = 1, l_2 = 4, f_3 = 1, l_3 = 3$).

To satisfy the track capacity constraints, one should impose that certain pairs of arcs, associated with different trains, cannot be selected in the overall solution. In particular, it

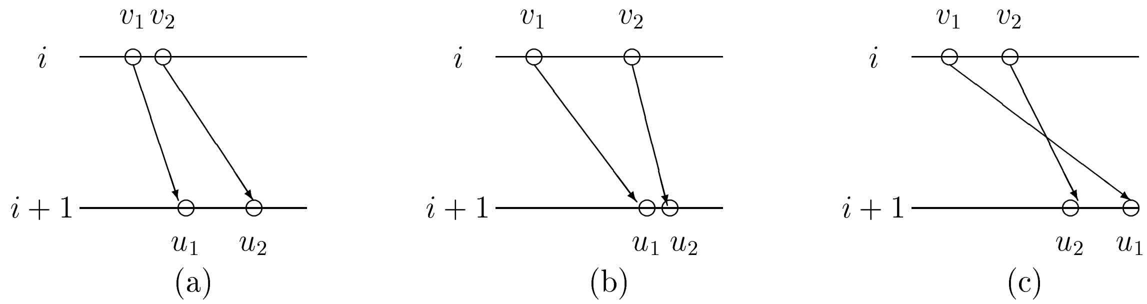


Figure 3: Incompatible pairs of arcs.

is sufficient to state that, for each pair of trains j, k and for each station $i \in (S^j \setminus \{l_j\}) \cap (S^k \setminus \{l_k\})$, the two segment arcs $(v_1, u_1) \in A^j$ and $(v_2, u_2) \in A^k$, $v_1, v_2 \in W^i$, $u_1, u_2 \in U^{i+1}$, cannot be both selected if at least one of the following conditions holds:

- the time interval between the time instants associated with v_1 and v_2 is smaller than d_i (i.e., $v_1 \preceq v_2$ and $\Delta(v_1, v_2) < d_i$), see Figure 3(a);
- the time interval between the time instants associated with u_1 and u_2 is smaller than a_{i+1} (i.e., $u_1 \preceq u_2$ and $\Delta(u_1, u_2) < a_{i+1}$), see Figure 3(b);
- the two arcs *cross* each other (i.e., $v_1 \preceq v_2$ and $u_2 \preceq u_1$), see Figure 3(c).

Such a pair of arcs $(v_1, u_1), (v_2, u_2)$ is called *incompatible*, whereas the other arc pairs are called *compatible*.

An *Integer Linear Programming* (ILP) formulation of TTP is the following. For each $j \in T$ and each arc $a \in A^j$, we introduce a binary variable x_a equal to 1 if and only if arc a is selected in an optimal solution, i.e., the path in the solution associated with train j contains arc a. For notational convenience, for each node $v \in V$ and train $j \in T$, let $\delta_j^+(v)$ and $\delta_j^-(v)$ denote the (possibly empty) sets of arcs in A^j leaving and entering node v, respectively. The ILP model then reads

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a x_a \tag{2}$$

subject to

$$\sum_{a \in \delta_j^+(\sigma)} x_a \le 1, \qquad j \in T, \tag{3}$$

$$\sum_{a \in \delta_j^-(v)} x_a = \sum_{a \in \delta_j^+(v)} x_a, \qquad j \in T, v \in V \setminus \{\sigma, \tau\}, \tag{4}$$

$$x_{a_1} + x_{a_2} \le 1, \qquad (a_1, a_2) \text{ incompatible arc pair}, \tag{5}$$

$$x_a \geq 0, \qquad a \in A, \tag{6}$$

$$x_a \text{ integer}, \qquad a \in A. \tag{7}$$

The objective function (2) is defined as the sum of the profits of the arcs associated with each path in the solution. Constraints (3) impose that at most one arc associated with a train is selected among those leaving the starting node σ, while constraints (4) impose equality on the number of selected arcs associated with a train entering and leaving each arrival or departure node. Consequently, the set of selected arcs associated with a train can either be empty, or define a path from the source to the sink. Finally, constraints (5) forbid the simultaneous selection of an incompatible pair of arcs, imposing the track capacity constraints.

Note that (2)–(7) is indeed a multicommodity flow formulation, in which the commodity index is hidden in the multigraph definition.

Formulation (2)–(7) may be improved by replacing inequalities (5) by the following stronger *clique* constraints. Let $\mathcal{C}$ denote the (exponentially large) family of maximal subsets C of pairwise incompatible arcs. The clique constraints read

$$\sum_{a \in C} x_a \leq 1, \qquad C \in \mathcal{C}. \tag{8}$$

It can be shown that the LP relaxation (2),(3),(4),(8),(6), can be solved efficiently through a cutting plane approach. However, this issue is out of the scope of this paper.

Both the number of variables and the number of constraints of formulation (2),(3),(4),(8), (6),and (7) (even excluding the clique inequalities) are very large for the real-world TTP instances that we consider. Hence, at the present state-of-the-art, it is impossible to design an exact algorithm based on this model which is capable of finding an optimal solution within reasonable computing time. For this reason, we decided to adopt a widely-used tool for finding heuristic solutions to very-large size ILP problems, namely Lagrangian relaxation combined with subgradient optimization. This technique which was proved to be successful at finding good quality solutions within short computing time for several combinatorial optimization problems. For instance, a survey on Lagrangian heuristics for the Set Covering Problem is given in Caprara, Fischetti and Toth [4].

We relax in a Lagrangian way the clique inequalities (8) by means of Lagrangian multipliers $\lambda_C \geq 0$. The Lagrangian relaxed problem then reads

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a x_a + \sum_{C \in \mathcal{C}} \lambda_C (1 - \sum_{a \in C} x_a) = \sum_{j \in T} \sum_{a \in A^j} \overline{p}_a x_a + \sum_{C \in \mathcal{C}} \lambda_C \tag{9}$$

subject to (3),(4),(6),(7), where, for each arc $a \in A$, $\mathcal{C}_a$ denotes the subfamily of cliques containing arc a, and $\overline{p}_a := p_a - \sum_{C \in \mathcal{C}_a} \lambda_C$ is the *Lagrangian profit* associated with a. For each multiplier vector λ, the Lagrangian relaxed problem calls, for each train $j \in T$, for the path with maximum Lagrangian profit ω_j, which can be computed in $O(|A^j|)$ time by a standard longest path algorithm in acyclic graphs. If $\omega_j \leq 0$, train j is canceled in the Lagrangian solution. The corresponding lower bound is given by

$$\sum_{j \in T} \max\{0, \omega_j\} + \sum_{C \in \mathcal{C}} \lambda_C, \tag{10}$$

and one can determine near-optimal multipliers by subgradient optimization.

The main drawback of this approach is not due to the fact that the number of constraints (8) is exponential (these constraints could be handled according to a so-called *relax-and-cut* framework, see the next section), but rather to the fact that the number of variables is very large, and each variable has its own Lagrangian profit associated. Hence, one should keep track of these Lagrangian profits, which is impractical for realistic size instances. Indeed, typical figures are 50 stations and 100 trains, each of which can be shifted by roughly 30 minutes and stretched by roughly 15 minutes, with time windows spanning approximately 75 minutes. With a (typical) time discretization of 1 minute, this implies that the number of arcs leaving each arrival/departure node is about 15, hence the number of arcs associated with each train is about $2 \times 50 \times 75 \times 15$, i.e., about 100,000, hence the overall number of arcs is larger than 10,000,000. Explicitly storing and updating 10,000,000 Lagrangian profits at each subgradient iteration is extremely time (and space) consuming, as well as (alternatively) computing these profits "on line" during the path computation for the solution of the Lagrangian relaxed problem.

Therefore, we developed an alternative model for the problem, illustrated in the next section, in which the track capacity constraints are modeled by variables associated with the *nodes* of G rather than with its arcs. This leads to a more manageable model, since we can associate Lagrangian multipliers with the nodes of G, with a considerable speed-up when the Lagrangian multipliers are updated and a substantial saving in the core memory space required by the algorithm.

3 A New Model

The new model requires the definition of new variables associated with nodes of G as well as with train-node pairs.

For each train $j \in T$, let $V^j = \{v \in V \setminus \{\sigma, \tau\} : \delta_j^+(v) \neq \emptyset \text{ and } \delta_j^-(v) \neq \emptyset\}$ be the set of nodes of G associated with j. For each node $v \in \bigcup_{j \in T} V^j$, let y_v be a binary variable equal to 1 if and only if there exists a train j whose associated path visits node v. Moreover, for each train $j \in T$ and for each node $v \in V^j$ let z_{jv} be a binary variable equal to 1 if and only if the path of train j visits node v. Finally, we let arc variables x_a be defined as in model (2)–(8). Note that the z variables can easily be derived from the x variables as

$$z_{jv} = \sum_{a \in \delta_j^-(v)} x_a, \qquad j \in T, v \in V^j, \tag{11}$$

and the y variables can be derived from the z variables as

$$y_v = \sum_{j \in T} z_{jv}, \qquad v \in V^j, \tag{12}$$

where $\sum_{j \in T} z_{jv} \leq 1$ for $v \in V^j$ since two trains cannot depart/arrive at the same station at the same time.

We next introduce three classes of inequalities to model the track capacity constraints. For each station i, recall that a_i and d_i represent the minimum time intervals between consecutive arrivals at and departures from i, respectively. The following *arrival time* constraints prevent that two consecutive arrivals at the same station are too close in time.

$$\sum_{w:\Delta(u,w)<a_i} y_w \leq 1, \qquad i \in S \setminus \{1\}, u \in U^i. \tag{13}$$

Analogously, the *departure time* constraints read

$$\sum_{w:\Delta(v,w)<d_i} y_w \leq 1, \qquad i \in S \setminus \{s\}, v \in W^i. \tag{14}$$

Both arrival and departure time constraints are *clique* inequalities.

The third class of constraints forbids overtakings between consecutive stations i and $i+1$. Consider two trains j, k and four nodes $v_1, v_2 \in W^i$, $u_1, u_2 \in U^{i+1}$, with $(v_1, u_1) \in A^j$ and $(v_2, u_2) \in A^k$, such that $v_1 \preceq v_2$ and $u_2 \preceq u_1$, i.e., arcs (v_1, u_1) and (v_2, u_2) cross each other as in Figure 3(c). The *overtaking* constraints forbid that the path of train j visits nodes v_1 and u_1 and the path of train k visits nodes v_2 and u_2. Written in its *weak* form, the constraint reads

$$z_{jv_1} + z_{ju_1} + z_{kv_2} + z_{ku_2} \leq 3. \tag{15}$$

These constraints can be strengthened (lifted) as follows. Let $v \in W^i$ and $u \in U^{i+1}$ be such that $v_1 \preceq v \preceq v_2$ and $u_2 \preceq u \preceq u_1$, respectively. The *strong* form of the overtaking constraints, which is the one actually used in our formulation, is

$$\sum_{w \in W^i \cap V^j : w \preceq v} z_{jw} + \sum_{w \in U^{i+1} \cap V^j : w \succeq u} z_{jw} + \sum_{w \in W^i \cap V^k : w \succeq v} z_{kw} + \sum_{w \in U^{i+1} \cap V^k : w \preceq u} z_{kw} \leq 3, \tag{16}$$

which is obviously valid for each $i \in S \setminus \{\sigma\}$, $v \in W^i$, $u \in U^{i+1}$, $j, k \in T, j \neq k$.

Overall, the new formulation has the form (2) subject to (3), (4), (6), (7), (11), (12), (13), (14), (16).

As anticipated, the main advantage of this new formulation is that the Lagrangian relaxation of the track capacity constraints, namely (13), (14) and (16), leads to a relaxed problem in which the profits of the x variables are unchanged, i.e., there is no penalty associated with the arcs of G, whereas there is a Lagrangian penalty associated with the nodes of G (due to the y variables) as well as a Lagrangian penalty associated with train-node pairs (due to the z variables). Formally, the Lagrangian relaxed problem reads

$$\max \sum_{j \in T} \sum_{a \in A^j} p_a x_a - \sum_{v \in V} \overline{q}_v y_v - \sum_{j \in T} \sum_{v \in V^j} \overline{r}_{jv} z_{jv} \tag{17}$$

subject to (3), (4), (6), (7), (11), (12), where $\overline{q}_v$, $v \in V$, and $\overline{r}_{jv}$, $j \in T, v \in V^j$, are *Lagrangian penalties* associated with the new variables. More precisely, whenever a positive Lagrangian multiplier λ_h is associated with constraint (13) (resp. (14)), the visit by any train path of a node w such that y_w is present in the constraint is assigned a penalty λ_h. Analogously, whenever a positive Lagrangian multiplier λ_h is associated with constraint (16), the visit by a path for train j (or k) of a node w such that z_{jw} (or z_{kw}) is present in the constraint is assigned a penalty λ_h. The resulting Lagrangian relaxed problem then calls for a set of paths for the trains, each having maximum Lagrangian profit, given by the sum of the original profits for the arcs in the path, minus the sum of the penalties assigned to the nodes visited by the path. If the maximum Lagrangian profit of a path for a train is nonpositive, the train is not scheduled by the optimal solution of the relaxed problem.

Near-optimal Lagrangian multipliers can be determined through subgradient optimization. As the number of relaxed constraints is far too large to be handled explicitly, we use a dynamic constraint-generation scheme sometimes referred to as *relax-and-cut*, see Fisher [7] and Escudero, Guignard and Malik [6]. We implicitly set all the Lagrangian multipliers to 0 and initialize a *constraint pool structure* to be empty. At each iteration of the subgradient optimization procedure, we identify constraints which are violated by the relaxed solution, possibly adding these constraints to the pool. More precisely, if two trains arrive at station i at time instants which are too close to each other, letting u_1 and u_2, $u_1 \preceq u_2$, be the nodes associated with the train arrivals, we check whether the pool contains any constraint of the form (13) involving both variables y_{u_1} and y_{u_2}. If such a constraint is found, we simply update the associated Lagrangian multiplier, otherwise we add to the pool the constraint (13) corresponding to the node u such that $u_1 \preceq u \preceq u_2$ and $\Delta(u_1, u)$ is as close as possible to $\Delta(u_2, u)$. Train departures from a station in time instants too close to each other are handled in an analogous way, thus adding to the pool constraints of the form (14). Finally, if train j overtakes train k between two consecutive stations i and $i+1$ without violating any departure constraint (14) from station i and arrival constraint (13) in station $i+1$, we have a violated overtaking constraint. In this case, let (v_1, u_1) be the segment arc corresponding to train j and (v_2, u_2) be the segment arc corresponding to train k and assume w.l.o.g. $v_1 \preceq v_2$ and $u_2 \preceq u_1$. We check whether the pool contains any constraint of the form (16) involving variables $z_{jv_1}, z_{ju_1}, z_{kv_2}, z_{ku_2}$. If such a constraint is found, we update its Lagrangian multiplier, otherwise we add to the pool the constraint (16) corresponding to trains j and k, stations i and $i+1$ and nodes $u \in U^{i+1}$ and $v \in W^i$ such that $\Delta(v, v_1)$ is as close as possible to $\Delta(v, v_2)$ and $\Delta(u, u_1)$ is as close as possible to $\Delta(u, u_2)$.

In the remainder of the section, we describe the additional details of the implementation of the subgradient optimization procedure.

Consider the h-th constraint in the pool, and let τ_h be the *subgradient component* of this constraint corresponding to the optimal solution of the current relaxation, given by the difference between the right-hand-side and the left-hand-side values. The associated Lagrangian multiplier λ_h is updated in a standard way by using the well-known formula introduced by Held and Karp [10]:

$$\lambda_h = \max \left\{ \lambda_h + \rho \frac{UB - LB}{||\tau||^2} \tau_h, 0 \right\} \tag{18}$$

where LB is the value of the best TTP solution found, UB is the best upper bound, $\rho > 0$ is a given *step size parameter*, and τ is the overall *subgradient vector* associated with the solution of the current relaxation. We remove the constraint from the pool if the associated multiplier reaches the value zero.

The initial value of the step size ρ is 1. We use the standard technique of halving ρ when no upper bound improvement occurs for a certain number of iterations (10 in our implementation). As to the stopping condition, it arises when ρ becomes smaller than a given threshold, and will be illustrated in detail in the next section.

4 The Heuristic Method

At each iteration of the subgradient optimization procedure, besides solving the Lagrangian relaxed problem, we also compute a heuristic solution by ranking the trains by decreasing values of the Lagrangian profit of the associated path in the relaxed solution, and then by scheduling the trains one by one, as shown in the next subsection. Being the schedule of the previous trains fixed, finding the (locally optimal) schedule of the next train j in the sequence simply amounts to the solution of a maximum profit path problem in the subgraph of G resulting from the removal of the arcs of A^j which are incompatible with those corresponding to the trains already scheduled.

Even for slightly different Lagrangian multipliers the resulting heuristic solutions may have considerably different quality, hence these solutions are worthwhile computing at each iteration of the subgradient optimization procedure, as mentioned, e.g., in Caprara, Fischetti and Toth [3] for the Set Covering Problem.

4.1 Basic constructive heuristic

For each train, according to the ranking, the heuristic chooses the path with maximum Lagrangian profit that is compatible with the paths already computed (those corresponding to the trains with higher ranking). If the actual (i.e., not Lagrangian) profit of this path is nonpositive, or if no path can be found for the train, the train is not scheduled in the solution. Choosing the path with maximum Lagrangian profit, instead of the path with maximum actual profit, is an attempt to take care of the trains with lower ranking which still have to be scheduled. Besides finding better solutions than its counterpart based on actual profits, this policy yields substantially different heuristic solutions for different Lagrangian multipliers, increasing the likelihood of improving the incumbent solution.

4.2 Refining heuristic

After the computation of each heuristic solution by the procedure of the previous subsection, we execute the following refining procedure.

For each train that is scheduled in the solution (in arbitrary order), we check if the associated timetable is shifted and/or stretched with respect to the ideal one. If this is the case, we compute the path with maximum actual profit for the train which is compatible with the paths for the other trains (which are kept fixed). This path may turn out to have a higher profit than the initial one, which was computed so as to maximize the Lagrangian profit, in which case the old path is replaced by the new one.

Afterwards, for each train that is not scheduled in the solution (in any order), we compute the path with maximum actual profit compatible with the scheduled trains, and test whether this profit is indeed positive. This may happen either because we are now considering actual profits instead of Lagrangian ones, or because the paths of the other trains have been changed by the previous phase. Whenever a path with positive profit for an unscheduled train is found, it is added to the solution.

The two phases above, namely the recomputation of the paths for the trains already scheduled and the attempt to find positive profit paths for the unscheduled trains, are iterated until no improvement in the solution value is achieved in both phases.

4.3 Train fixing and removing

As shown, e.g., in Caprara, Fischetti and Toth [3], the iterative fixing of the value of some variables and the reapplication of a Lagrangian heuristic can be very successful in obtaining good solutions for many classical combinatorial optimization problems. We have therefore implemented the following path fixing/unfixing scheme. We will say that a train is *fixed* if a path associated with the train is fixed in the solution, *removed* if the train is not considered when a solution is constructed, and *free* otherwise.

As already mentioned, our subgradient procedure halves the step size ρ when no upper bound improvement occurs for 10 iterations. When the value of ρ becomes smaller than a given threshold (0.1 in our implementation), we restore the Lagrangian multipliers corresponding to the incumbent solution (i.e., the multipliers at the subgradient iteration in which our heuristic found the best solution) and call a fixing/removing procedure working as follows.

If there are $r > 0$ free trains that are not scheduled in the incumbent solution, the $\min\{r, \lceil t/20 \rceil\}$ of them with lowest Lagrangian profit are removed. Otherwise (i.e., if all the free trains are scheduled), we iteratively perform the following steps. For each free train j we find the path P^j with maximum Lagrangian profit ω_j that is compatible with the paths already fixed. We find the train j^* such that $\omega_{j^*} = \max\{\omega_j : j \text{ free}\}$ and, if the actual profit of path P^{j^*} is positive, we fix j^* according to P^{j^*}. If the actual profit of P^{j^*} is nonpositve, we do not consider j^* in the subsequent iterations of the procedure. The procedure is iterated until $\lceil t/20 \rceil$ previously free trains have been fixed.

After each removal or fixing, we update the Lagrangian multipliers accordingly, as illustrated below. At the end of the fixing/removing procedure, the subgradient procedure is restarted from the multipliers obtained, and ρ is set to 0.25.

When the current upper bound for the problem with trains removed and fixed becomes not larger than the profit of the incumbent solution, we call an unfixing procedure that frees the $\lceil t/20 \rceil$ fixed trains with lowest Lagrangian profit as well as the $\min\{r, \lceil t/20 \rceil\}$ removed trains with highest Lagrangian profit (with respect to the current multipliers), where r now denotes the number of trains removed. Then, we restart the subgradient optimization procedure, with $\rho = 0.25$ and with the multipliers corresponding to the incumbent solution.

If the incumbent solution was not improved between a call to the unfixing procedure and the iteration in which step size ρ became smaller than 0.1, *all* trains are freed and the procedure is restarted with the best multipliers found with all trains free (i.e., those yielding the best upper bound value) and $\rho = 0.25$.

The algorithm terminates when a given time limit (or iteration limit) is reached.

When a train j is removed or fixed, the Lagrangian multipliers are updated trying to take into account this choice, as follows. First of all, the multipliers for all constraints (16) involving train j are set to 0. Second, for each constraint h of the form (13) or (14) and for each free train k (including j), we count how many times, say ψ_k, (over all subgradient iterations) the path in the Lagrangian solution for train k visited nodes in the constraint. Let F be the set of trains k such that $\psi_k > 0$. If the fixed/removed train $j \in F$ and $|F| = 2$, the multiplier λ_h of constraint h is set to 0. Otherwise, if $j \in F$, we set

$$\lambda_h := \lambda_h \left(1 - \frac{\psi_j}{\sum_{k \in F} \psi_k}\right),$$

heuristically taking into account the contribution of train j to λ_h.

4.4 Data structures for computing the train paths

In order to compute the paths for the trains and to handle the Lagrangian multipliers, we use a global structure for G that stores information for each node of the graph. In particular, we keep track of the "level of freedom" of each node u, via a variable called FORBIDDEN_FLAG(u), that tells whether a train path may go through the node, in which case the level is 0, or not, in which case it is set to a positive value. We also store for each node the sum of the Lagrangian multipliers for constraints (13) and (14) involving the node. As mentioned above, we do not store explicitly the arcs of the graph, as this would require a huge amount of memory for instances of realistic size.

Moreover, in our implementation we store, for each train j, the graph in which the paths for the train are computed. This data structure is called TRAIN_GRAPH(j), and contains all the nodes that the path of train j may visit. In a previous version of the code, the train graphs were not represented explicitly, but only implicitly redefined every time a path for a train was computed. This first version had the advantage that the core memory requirement was very small, but the obvious drawback that the train path computation (that has to be carried out $O(t)$ times at every iteration of the subgradient optimization procedure) was much slower. In order to limit the core memory requirement, anyway, the TRAIN_GRAPH(j) is stored efficiently as follows.

For each node u in TRAIN_GRAPH(j), we store the index of the corresponding node in the global graph, the sum of the Lagrangian multipliers for constraints (16) involving train

j and node u, and the list of the *successors* of u, i.e., the nodes v such that arc (u, v) is feasible for a path for train j. This list is stored in the following compact way. Assuming that u is a departure node from station i (the case of an arrival node is analogous), we store only the first and the last (according to increasing time instants) arrival nodes for the train in station $i + 1$ that are compatible with departure node u.

After some path has been added to the solution (either because it is fixed or because we are constructing a feasible heuristic solution), some of the successors v of node u may become infeasible as the selection of arc (u, v) would violate constraints (16). We therefore associate with node u a (possibly empty) list of *forbidden successors*, that stores the first and last forbidden successors of u.

5 Computational Results

Our algorithm was implemented in C and tested on a set of instances from FS SpA and from Ansaldo Segnalamento Ferroviario SpA (Ansaldo for short), a large company working for FS and taking care of part of the timetable design. The time period considered is one day, and the time discretization is one minute (i.e., $q = 1440$).

The instances given by FS were designed within the EUROPE-TRIS Project with the aim of simulating the requests of different train operators along a main track line crossing the border of two European countries. The instances provided by Ansaldo correspond to the actual timetables requested by the FS operators.

For almost all the instances above, all of the trains can be scheduled in a solution. We were also asked by FS for testing our algorithm in a different situation, in which the number of trains requested is considerably larger than the number of trains that can be scheduled. To this end, we were also provided with some highly congested instances.

The lower bounds a_i and d_i between consecutive arrivals and departures of trains in a station i are set to 4 and 2 minutes, respectively, according to the current operational rules. The function penalizing the train shift is defined as $\phi_j(\nu_j) := \alpha_j \nu_j$, i.e., the penalty is linear in the shift. The coefficients used in the objective function are illustrated in Table 1. In particular, the profit coefficients are identical for the trains of the same type. For instance, the profit achieved if a Eurostar train is scheduled according to its ideal timetable is 200, and there is a penalty of 7 for each minute of shift as well as a penalty of 10 for each minute of stretch. Note that the shift is penalized less than the stretch for all train types. No time window constraint is imposed on the instances that we consider.

The main characteristics of the instances by FS are outlined in Table 2, whereas the characteristics of the instances by Ansaldo are given in Table 3 and the characteristics of

Train Type	π_j	α_j	γ_j
Eurostar	200	7	10
Euronight	150	7	10
Intercity	120	6	9
Express	110	5	8
Direct	100	5	8
Local	100	5	6
Freight	100	2	3

Table 1: Train profit coefficients depending on the train type.

the congested instances are given in Table 4. Entry **# trains** gives the total number of trains on input followed by an array with the number of Eurostar, Euronight, Intercity, Express, Direct, Local and Freight trains, respectively. Moreover, entry **timing** gives the total number of trains whose departure time is, respectively, between 0:00am and 3:59am; 4:00am and 7:59am; 8:00am and 11:59am; 12:00pm and 3:59pm; 4:00pm and 7:59pm; and 8:00pm and 11:59pm.

Instance	first stat.	last stat.	# stat.	# trains	timing
BN-BO	Brennero	Bologna	40	68 (1,0,5,13,11,38,0)	(2,16,11,18,14,7)
MU-VR	Munich (D)	Verona	48	54 (0,0,0,7,0,47,0)	(0,12,28,14,0,0)
CH-RM	Chiasso	Rome	73	36 (15,0,9,0,0,7,5)	(0,4,13,15,4,0)
MO-MI-1	Modane (F)	Milan	39	16 (1,0,3,0,4,5,3)	(0,0,0,13,3,0)
MO-MI-2	Modane (F)	Milan	39	23 (0,3,1,11,0,2,6)	(9,10,4,0,0,0)

Table 2: Characteristics of the instances from FS SpA.

Instance	first stat.	last stat.	# stat.	# trains	timing
PC-BO-1	Piacenza	Bologna	16	221 (28,3,52,35,28,14,61)	(30,44,34,30,46,37)
PC-BO-2	Piacenza	Bologna	16	93 (12,3,18,10,20,6,24)	(7,13,17,17,22,17)
PC-BO-3	Piacenza	Bologna	16	60 (12,1,14,0,12,7,14)	(0,7,25,24,4,0)
PC-BO-4	Piacenza	Bologna	16	40 (6,0,10,0,12,2,10)	(0,0,40,0,0,0)
BZ-VR	Bolzano	Verona	20	128 (34,0,1,10,11,38,34)	(16,21,17,26,29,19)
CH-MI	Chiasso	Milan	15	194 (20,1,29,8,19,66,51)	(21,38,27,42,32,34)

Table 3: Characteristics of the instances from Ansaldo.

Instance	first stat.	last stat.	# stat.	# trains	timing
MO-MI-50	Modane (F)	Milan	39	50 (9,0,14,0,9,11,7)	(0,0,3,28,16,3)
MI-MI-100	Modane (F)	Milan	39	100 (19,0,26,0,13,22,20)	(0,0,5,49,38,8)
MI-MI-200	Modane (F)	Milan	39	200 (39,0,30,0,42,54,35)	(0,0,5,94,83,18)
MI-MI-300	Modane (F)	Milan	39	300 (65,0,50,0,58,62,65)	(0,0,17,146,119,18)
MI-MI-400	Modane (F)	Milan	39	400 (72,0,90,0,74,75,89)	(0,0,24,177,162,37)
MI-MI-500	Modane (F)	Milan	39	500 (102,0,96,0,95,101,106)	(0,0,36,225,192,47)

Table 4: Characteristics of the congested instances.

Tables 5, 6 and 7 report the results obtained with our algorithm on the instances illustrated above. We run our algorithm on a Digital Ultimate Workstation 533MHz, with a limit of 1,000 subgradient iterations, and recorded the best solution value and the best upper bound found before this limit. The entries in the table have the following meaning:

ideal prof. is the sum of the profits achievable by scheduling each train according to its ideal timetable, i.e., $\sum_{j \in T} \pi_j$ (this is a trivial upper bound on the final profit).

best UB is the best upper bound found by the subgradient optimization procedure with an ad hoc tuning of the parameters (which are not the same used for the determination of the best heuristic solution). Within parentheses, we also report the percentage decrease over the ideal profit.

greedy sol. is the solution value found by scheduling the trains by decreasing values of π_j (breaking ties arbitrarily) and assigning to each train the timetable corresponding to the maximum profit path compatible with the previous trains. The manual methods proceed in a similar way, therefore the quality of the solution provided by the FS practitioners is typically close to the value given in this entry.

best sol. is the value of the best solution found by our heuristic method, with an indication of the percentage improvement over the greedy solution.

GAP% is the percentage gap between the value of the best heuristic solution and the best upper bound.

 # sched is the number of trains scheduled in the best solution.

avg. ν is the average shift (in minutes) for the trains scheduled in the best solution.

avg. μ is the average stretch (in minutes) for the trains scheduled in the best solution.

time is the overall running time needed to find the best solution (expressed in CPU seconds).

As anticipated, our method succeeded in scheduling almost all trains for most instances from FS and Ansaldo. The exceptions are instances MU-VR, PC-BO-1 and PC-BO-4. For most instances, the final percentage gap is quite small (less than 2%). For the remaining cases, our best heuristic solution is considerably better than the greedy one.

Table 7 analyzes highly congested instances obtained by increasing the number of required trains to be daily scheduled. As expected, our method allows for a substantial improvement with respect to the greedy solutions and of the upper bound with respect to the ideal profit.

Name	ideal prof.	best UB	greedy sol.	best sol.	GAP%	# sched.	avg. ν	avg. μ	time
BN-BO	7130	6891 (3.4%)	6746	6779 (0.5%)	(1.7%)	68	0.6	0.3	34
MU-VR	5470	4991 (8.8%)	3332	4208 (20.8%)	(18.6%)	48	1.4	1.1	92
CH-RM	5280	5129 (2.9%)	4844	4871 (0.6%)	(5.3%)	35	1.9	1.3	48
MO-MI-1	1760	1713 (2.7%)	1648	1684 (2.1%)	(1.7%)	16	0.6	0.4	0
MO-MI-2	2580	2533 (1.8%)	2486	2520 (1.3%)	(0.6%)	23	0.9	0.3	78

Table 5: Results on instances from FS SpA.

Name	ideal prof.	best UB	greedy sol.	best sol.	GAP%	# sched.	avg. ν	avg. μ	time
PC-BO-1	25740	24142 (6.2%)	19535	21397 (8.7%)	(12.8%)	192	0.8	0.8	672
PC-BO-2	11010	10947 (0.6%)	10848	10882 (0.3%)	(0.6%)	93	0.0	0.2	118
PC-BO-3	7450	7222 (3.1%)	6709	7119 (5.8%)	(1.4%)	60	0.6	0.8	271
PC-BO-4	4800	4130 (14.0%)	3350	3656 (8.4%)	(13.0%)	35	3.2	0.9	266
BZ-VR	16300	16101 (1.2%)	15902	16003 (0.6%)	(0.6%)	127	0.3	0.0	266
CH-MI	21930	21467 (2.1%)	20923	21215 (1.4%)	(1.2%)	193	0.7	0.1	285

Table 6: Results on instances from Ansaldo.

The large running time spent for obtaining the best solution may not be acceptable, e.g., when the problem is solved in a real-time environment when some trains have been delayed and one would like to adjust the schedules, or when a teleconference is held to decide the train timetables (this one of the scenarios considered by the EUROPE-TRIS Project). In these cases, solutions of comparable quality may be obtained in a much smaller running time. Table 8 illustrates the values of the current best solution as a function of the CPU time for instance MO-MI-500. The figure shows that good solutions are found in short computing times. The other instances exhibit a similar behavior.

Name	ideal prof.	best UB	greedy sol.	best sol.	GAP%	# sched.	avg. ν	avg. μ	time
MO-MI-50	6180	5757 (6.8%)	4907	5176 (5.2%)	(11.2%)	47	3.0	0.6	68
MO-MI-100	12420	11194 (9.9%)	8657	9322 (7.1%)	(20.1%)	78	1.4	0.7	701
MO-MI-200	24500	20313 (17.1%)	14932	18539 (19.5%)	(9.6%)	162	2.3	0.1	1408
MO-MI-300	37500	27280 (27.3%)	20150	24566 (18.0%)	(11.0%)	207	3.2	0.1	3943
MO-MI-400	49000	31039 (36.7%)	23133	27170 (14.9%)	(14.2%)	233	3.0	0.2	4657
MO-MI-500	62120	34930 (43.8%)	24771	30658 (19.2%)	(13.9%)	260	3.1	0.1	5501

Table 7: Results on highly congested instances.

sol.	24771	26819	27548	28523	28719	28873	28882	29291	29325	29770	29792	30062	30203	30400	30427	30564	30658
time	14	23	34	45	82	323	383	481	789	829	1314	1331	1429	1587	2487	4048	5504

Table 8: Incumbent solution value updating for instance MO-MI-500.

Acknowledgments

This work has been supported by EU Esprit Project EUROPE-TRIS through Consorzio Padova Ricerche. We are grateful to Pier Luigi Guida from FS for his efforts in coordinating the project and for useful discussions. Moreover, thanks are due to Giuseppe Sacco from FS and to Carlo Milani from Ansaldo for having provided us with the instances presented in the paper.

References

[1] U. Brännlund, P.O. Lindberg, A. Nöu and J.E. Nilsson, "Allocation of Scarce Track Capacity Using Lagrangian Relaxation", *Transportation Science* 32, 358–369 (1998).

[2] X. Cai and C.J. Goh, "A Fast Heuristic for the Train Scheduling Problem", *Computers and Operations Research* 21, 499-510 (1994).

[3] A. Caprara, M. Fischetti and P. Toth, "A Heuristic Method for the Set Covering Problem", *Operations Research* 47, 730–743 (1999).

[4] A. Caprara, M. Fischetti and P. Toth, "Algorithms for the Set Covering Problem", to appear in *Annals of Operations Research.*

[5] M. Carey and D. Lockwood, "A Model, Algorithms and Strategy for Train Pathing", *Journal of the Operational Research Society* 46, 988–1005 (1995).

[6] L.F. Escudero, M. Guignard and K. Malik, "A Lagrangian Relax and Cut Approach for the Sequential Ordering with Precedence Constraints", *Annals of Operations Research* 50, 219–237 (1994).

[7] M. Fisher, "Optimal Solution of Vehicle Routing Problems Using Minimum k-Trees", *Operations Research* 42, 626–642 (1994).

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[9] P.T. Harker and S. Hong, "Pricing of Track Time in Railroad Operations: an Internal Market Approach", *Transportations Research* 28-B, 197-212 (1994).

[10] M. Held and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II", *Mathematical Programming* 1, 6–25 (1971).

[11] A. Higgings, E. Kozan, L. Ferreira, "Heuristic Techniques for Single Line Train Scheduling", *Journal of Heuristics* 3, 43–62 (1997).

[12] D. Jovanovic and P.T. Harker, "Tactical Scheduling of Rail Operations: the SCAN I System", *Transportation Science* 25, 46–64 (1991).

[13] V. Kann and A. Panconesi, "Hardness of Approximation", in M. Dell'Amico, F. Maffioli, S. Martello (eds.), *Annotated Bibliographies in Combinatorial Optimization*, J. Wiley and Sons, Chichester (1997), 13–30.

[14] T. Lindner and U.T. Zimmermann, "Cost-Oriented Train Scheduling", *8th International Conference on Computer Aided Scheduling of Public Transport (CASPT 2000)*, Berlin, June 2000.

[15] L.W.P. Peeters and L.G. Kroon, "A Cycle Based Optimization Model for the Cyclic Railway Timetabling Problem", *8th International Conference on Computer Aided Scheduling of Public Transport (CASPT 2000)*, Berlin, June 2000.

[16] E.R. Petersen and R. Merchant, "Scheduling of Trains in a Linear Railway System", *INFOR* 19, 246–258 (1981).

[17] A. Schrijver and A. Steenbeek, "Timetable Construction for Railned", Technical Report, CWI, Amsterdam, 1994 (in Dutch).

[18] B. Szpigel, "Optimal Train Scheduling on a Single Track Railway", in M. Ross (editor) *OR'72*, North-Holland, Amsterdam, 343–351, 1973.