Non-cyclic Train Timetabling and Comparability Graphs
Valentina Cacchiani, Alberto Caprara, Paolo Toth

DEIS, Università di Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy
{valentina.cacchiani,alberto.caprara,paolo.toth}@unibo.it

Abstract

We consider the customary formulation of non-cyclic train timetabling, calling for a maximum-profit collection of compatible paths in a suitable graph. The associated ILP models look for a maximum-weight clique in a(n exponentially-large) compatibility graph. By taking a close look at the structure of this graph, we analyze the existing ILP models and propose some new stronger ones, all having the essential property that both separation and column generation can be carried out efficiently. Computational results show that the LP relaxations of the new formulations can lead to notably stronger upper bounds on highly-congested instances.

1 Introduction

The train timetabling problem has been widely studied in the literature: we refer to [1] (Deliverable D3.1) and [6] for surveys on the problem, in the cyclic and non-cyclic versions. In cyclic timetabling the timetable is repeated with a cycle time (typically one hour), and all time instants (expressing train departures and arrivals) are expressed modulo this cycle time, whereas in non-cyclic timetabling time instants can be ordered linearly. In other words, only in the non-cyclic case the notion “time instant \(i_1\) is before time instant \(i_2\)” is defined. Non-cyclicity does not prevent timetables from being repeated with a cycle time (typically one day), but in this case one needs to have a sufficiently “wide” time interval within the cycle time (typically in the night) during which no train is running, so that time instants can be ordered from the end of this interval to its beginning. For instance, in the case study of [5], non-cyclicity is guaranteed if, within a day, for each track there is an interval of 4 minutes with no train arriving and an interval of 2 minutes with no train departing (moreover, these two intervals may be distinct from track to track).

From an application viewpoint cyclicity is an advantage for the passengers, as the timetable can easily be remembered. However, cyclicity is also more expensive since the same timetable is run in the off-peak hours, keeping the railway network congested and the number of trains running high although this would not be necessary to match the demand. In addition, in a competitive market, where more train operators utilize the same infrastructure and the infrastructure manager modifies (and possibly cancels) their requests, the non-cyclic version of timetabling seems to be more appropriate and efficient.

1.1 The problem considered

We consider a general version of the Non-cyclic Train Timetabling Problem (NTTP), which calls for a maximum-profit set of timetables for a set of trains traveling on a railway network, composed by stations connected together by one or more tracks. The timetables must satisfy the following track capacity constraints:

- a minimum time interval must elapse between two consecutive departures on the same track on the same direction;
- a minimum time interval must elapse between two consecutive arrivals on the same track on the same direction;
- overtaking along a track is not allowed.

Moreover, for possible two-way tracks along which trains may travel in opposite directions, a minimum time interval must elapse between an arrival of a train on the track in one direction and a departure of a train on the track in the opposite direction, and crossing along a track is not allowed. In this work, for simplicity, we focus our attention on the case in which all trains travel in the same direction along the tracks, which is almost always the case at the planning stage and simplifies the presentation a lot. On the other hand, all the results in this paper apply (or can easily be extended) to the case in which there are two-way tracks.

There are a few real-world special cases of this general problem, with different objective functions, and the results of this paper apply to all of these. In order to give a concrete example here, we briefly illustrate the case in [6]. In this case, for each train, we are given on input an ideal timetable, specifying the desired departure and arrival time at each station that must be visited by the train. Altogether, the ideal timetables do not satisfy the track capacity constraints, and they must be changed by either shifting the departure of a train from its first station (and consequently shifting the entire timetable) or stretching the running time of a train, i.e., increasing the time elapsing from the departure from its first station to the arrival at its last station. These changes produce the so-called actual timetables. (In general, the path followed by the train may not be unique, even if it has to visit all the stations specified in the ideal timetable, see, e.g., [4].)

If a train is scheduled according to its ideal timetable, it gains an ideal profit. Otherwise, the profit is decreased depending on its shift and stretch. If a train gets a null or negative profit, due to these changes, it is cancelled. The goal is to change as little as possible the ideal timetables, in order to produce on output maximum-profit timetables satisfying the track capacity constraints.

1.2 Graph representation

Let $T := \{1, \ldots, |T|\}$ denote the set of trains. A customary formulation of the problem (see, e.g., [6]) considers a discretization of the time horizon with interval discretization $\delta$. For simplicity, we assume $\delta = 1$, i.e., all times are integers expressed in units of the discretization interval. We let $H := \{1, \ldots, |H|\}$ denote the set of time instants in the (non-cyclic) time horizon, numbered according to their linear order. For instance, we might have $\delta$ equal to one minute and $|H| = 1440$, the number of minutes in a day. Moreover, let $L$ be the set of tracks in the railway network, each joining two stations without intermediate stations in between.

Discretization allows one to define a directed acyclic graph $G = (V, A)$ in which nodes correspond to events, namely to arrivals or departures of trains in stations along specified tracks at given time instants. Formally, each node $v \in V$ can either be a departure node or an arrival node, and is associated with a time instant $h(v) \in H$, a track $\ell(v) \in L$, and a station $s(v)$ that is one of the endpoints of $\ell(v)$. The arcs represent the travel of a train between two stations along specified tracks, or the stop of a train at a station. Formally, each arc $(u, v) \in A$ is such that $h(u) \leq h(v)$ and can either be a travel arc or a stop arc. For a travel arc, $u$ is a departure node, $v$ an arrival node, $\ell(u) = \ell(v)$ (i.e., the two nodes are associated with the same track), $s(u), s(v)$ are the two endpoints of $\ell(u)$, and $h(v) - h(u)$ is the associated travel time. For a stop arc, $u$ is an arrival node, $v$ a departure node, $s(u) = s(v)$ (i.e., the two nodes are associated with the same station), and $h(v) - h(u)$ is the associated stop time.
If the path for a train \( t \in T \) must visit stations \( s_1, s_2, \ldots, s_m \) in this order, a timetable for \( t \) is obtained by defining, for \( i = 1, \ldots, m - 1 \), a time instant for the departure of \( t \) from \( s_i \), a time instant for the arrival of \( t \) at \( s_{i+1} \), and a track \( \ell \in L \) having \( s_i, s_{i+1} \) as endpoints along which \( t \) travels. Adding to \( G \) an artificial source \( \sigma \) with an outgoing arc to all departure nodes and an artificial sink \( \tau \) with an ingoing arc from all arrival nodes, there is a correspondence between the feasible timetables for a train \( t \in T \) and the collection \( P^t \) of the paths from \( \sigma \) to \( \tau \) in a suitable arc-induced subgraph \( G^t \) of \( G \) (see [5, 6] for further details).

We consider the (typical) case in which the profit/cost associated with a timetable can be expressed as a linear function of the arcs in the corresponding path. In this case, the best timetable for a single train \( t \in T \) is given by a maximum-profit path on the acyclic graph \( G^t \), and can be computed in linear time (in the size of \( G^t \)) by dynamic programming. The difficulty of the problem comes from the fact that paths for distinct trains can conflict, due to the track capacity constraints illustrated next.

Each track \( \ell \in L \) is associated with a subset \( A_\ell \subseteq A \) of the arcs in \( G \), representing some train traveling along \( \ell \) with given departure and arrival times. Moreover, for each train \( t \in T \) and track \( \ell \in L \), every path in \( P^t \) can contain at most one arc in \( A_\ell \) (meaning that the train path can traverse each track at most once). For a given track \( \ell \in L \), consider two trains \( t_1, t_2 \) along with two paths \( P_1 \in P^{t_1}, P_2 \in P^{t_2} \) containing two arcs \( a_1 \in P_1 \cap A_\ell, a_2 \in P_2 \cap A_\ell \). Arc \( a_1 \) represents the departure of \( t_1 \) from the initial station of \( \ell \) at time, say, \( d_1 \) and its arrival at the final station of \( \ell \) at time, say, \( r_1 \). Similarly, arc \( a_2 \) represents the departure of \( t_2 \) from the initial station of \( \ell \) at time, say, \( d_2 \) and its arrival at the final station of \( \ell \) at time, say, \( r_2 \). Assuming without loss of generality that \( d_1 \leq d_2 \), we have that paths \( P_1, P_2 \) respect the track capacity constraints on track \( \ell \) if

\[
d_2 \geq d_1 + \alpha_\ell \quad \text{and} \quad r_2 \geq r_1 + \beta_\ell,
\]

where \( \alpha_\ell \) and \( \beta_\ell \) are given parameters, possibly depending on the track \( \ell \). In words, there is a minimum time distance between departures (\( \alpha_\ell \)) and arrivals (\( \beta_\ell \)) along each track \( \ell \), and trains cannot overtake each other along this track.

1.3 Contents and notation

In this paper, we will take a close look at existing ILP formulations for NTTP, involving (exponentially-many) binary variables associated with paths in \( G \) corresponding to timetables. We observe that all these ILPs call for a maximum-weight clique in the same (exponentially-large) compatibility graph, contain only stable-set constraints, and differ only in the type of stable sets actually considered. This allows us to unify all these ILPs under a common framework, which is our first main contribution. Moreover, this unification process naturally suggests new, stronger, ILP formulations of the same type, sharing with the existing ones the property that both separation and column generation can be carried out efficiently. This is our second main contribution. A third main contribution would have been achieved if these new ILPs had allowed us to find much better upper bounds for the real-world instances that we considered in previous papers. Unfortunately, this is not the case, for reasons that will be discussed in the experimental results section. However, for suitable “highly-congested” variants of these instances, we indeed report notably better upper bounds.

In order to have a general view of the ILP formulations and to be able to compare them, it will be fundamental to point out the underlying graphs; for this reason we conclude the introduction with some basic graph-theoretic notions and notations that we will use extensively.
Given an undirected graph \( F \), a (maximal) stable set is a (maximal) node subset such that no node pair in the subset is an edge of \( F \). Let \( \mathcal{S}(F) \) denote the collection of all maximal stable sets of \( F \). A (maximal) clique is a (maximal) node subset of \( F \) such that all node pairs in the subset are edges of \( F \). The complement of \( F \) is the graph on the same node set whose edges are exactly the node pairs that are not edges of \( F \). Given two undirected graphs \( F_1, F_2 \) on the same node set, their edge intersection \( F_1 \cap F_2 \) is the graph on the same node set with the edges that are present in both \( F_1 \) and \( F_2 \).

A comparability graph on node set \( N \) is an undirected graph whose edges can be oriented so as to get an acyclic directed graph \( D = (N, A) \) which is transitive, i.e., such that \((i, j), (j, k) \in A\) implies \((i, k) \in A\). Note that, given a comparability graph and the associated orientation, the relation \( \prec \) on node set \( N \) given by \( i \prec j \) if and only if \((i, j) \in A\) is a partial order on \( N \). Viceversa, given a partial order \( \prec \) on a set \( N \), we get a comparability graph by first considering the directed graph \( D = (N, A) \) where \((i, j) \in A\) if and only if \( i \prec j \) and then neglecting the orientation of the arcs in \( A \). An interval graph is a graph whose nodes correspond to intervals on the real line and whose edges correspond to pairs of intervals that have nonempty intersection.

## 2 ILP Formulations, Graphs, and Separation

A natural class of ILP formulations for NTTP considered, e.g., in [2, 4, 5] contains binary variables associated with the arcs of \( G \). These ILPs are well suited for dual-heuristic approaches such as Lagrangian relaxation with subgradient optimization. On the other hand, as far as canonical LP-based approaches are considered, the LP relaxations of these ILPs are extremely expensive to solve exactly, as noted in [3], where the solution times are not even reported as they are orders of magnitude larger than those of the equivalent (in terms of optimal value) LP relaxations of ILP formulations obtained by associating a binary variable \( x_P \) with each path \( P \in \mathcal{P} \). In this section, we present several such ILP formulations and discuss the underlying graphs and the associated separation problems.

For each \( t \in T \), recalling that \( \mathcal{P}^t \) denotes the collection of possible paths for train \( t \), let \( \pi_P \) be the profit of path \( P \in \mathcal{P}^t \). Moreover, let \( \mathcal{P} := \mathcal{P}^1 \cup \cdots \cup \mathcal{P}^t \) be the overall (multi-)collection of paths. Two paths \( P_1, P_2 \in \mathcal{P} \) are compatible, i.e., they can be both selected in the solution, if the following conditions hold:

- the two paths are associated with distinct trains;
- for each track \( \ell \in L \) traversed by both \( P_1, P_2 \), the two paths respect the track capacity constraints on \( \ell \).

The objective is the maximization of the profits of the paths selected with the constraint that all paths selected are compatible. The compatibility relation is naturally represented by an auxiliary graph \( F = (\mathcal{P}, E) \) with one node for each path and an edge joining each pair of compatible paths. Then, NTTP calls for a maximum-weight clique in \( F \). Note that \( F \) is the edge intersection of the following \(|L| + 1\) graphs on node set \( \mathcal{P} \):

- \( F_t \), in which two nodes are joined by an edge if and only if the corresponding paths are associated with distinct trains;
• $F_\ell, \ell \in L$, in which two nodes are joined by an edge if and only if the corresponding paths either do not both traverse track $\ell$, or they traverse it by respecting the track capacity constraints.

2.1 The structure of $F_T$ and $F_\ell$

The structure of $F_T$ is elementary, namely it is a collection of the $|T|$ stable sets $P_t, t \in T$, with edges joining each pair of nodes belonging to distinct stable sets. In other words:

Observation 1 $F_T$ is a complete $|T|$-partite graph;

The structure of $F_\ell$ is more interesting:

Observation 2 For $\ell \in L$, $F_\ell$ is a comparability graph.

Proof We define the partial order associated with $F_\ell$. First consider the paths in $P$ with one arc associated with track $\ell$. Two such paths $P_1, P_2$ are joined by an edge in $F_\ell$ if (1) holds for the associated departure and arrival times $d_1, d_2, r_1, r_2$, in which case we say that $P_1 \prec P_2$. This is clearly a transitive relation. This partial order can easily be extended to all paths with no arc associated with track $\ell$, since these are nodes incident to all other nodes in $F_\ell$.

2.2 A general, impractical ILP formulation

The most natural ILP formulation of NTTP would be to associate a constraint with each stable set of $F$. The formulation reads:

$$\max \sum_{P \in P} \pi_P x_P, \quad (2)$$

$$\sum_{P \in S} x_P \leq 1, \quad S \in S(F), \quad (3)$$

$$x_P \in \{0, 1\}, \quad P \in P. \quad (4)$$

Although the corresponding LP relaxation is fairly weak for maximum-weight clique in general, in the cases in which the objects represented by the nodes have a special structure (e.g., knapsack solutions, paths or cycles in a graph) the resulting upper bound may turn out to be strong, see, e.g., [7, 9]. On the other hand, even putting aside the fact that $|P|$ may be exponential, the solution of this LP relaxation turns out to be hard, recalling the well-known equivalence between separation and optimization [8].

Proposition 1 The separation of constraints (3) is strongly NP-complete even when $|P| = |T|$ (one feasible path per train), namely the problem of finding a maximum-weight stable set on a generic graph with $n$ nodes can be reduced to it, setting $|T| := |P| := n$.

Proof Proposition 1 in [5] shows how to transform a generic undirected graph $H$ with $n$ nodes into an NTTP instance with $n$ trains, one path per train, and paths that are compatible if and only if the corresponding nodes in $F$ are joined by an edge. (In other words, the auxiliary graph $F$ associated with this NTTP instance coincides with $H$.) At this point, the problem of testing if $H$ contains a stable set of weight larger than a threshold $B$ is equivalent to testing if there exists a constraint (3) that is violated by a solution $x^*$, by setting the value $x_P$ of each path equal to the weight of the corresponding node in $H$ divided by $B$. \hfill $\square$
Jointly with the fact that $|\mathcal{P}|$ is in general exponentially large, there appears to be no chance in practice to solve effectively the LP relaxation of the above ILP formulation for NTTP instances of interest, since the separation of (3) leads to a much harder column generation problem with respect to the other (weaker) constraints discussed next, as the dual variables associated with (3) do not correspond to arcs of $G$.

2.3 Practical ILP formulations

Forgetting about the whole set of constraints (3), a natural approach is to concentrate on alternative constraints with the following structure.

**Definition 1** A set of constraints of the form

$$\sum_{P \in S} x_P \leq 1, \quad S \in S',$$

(5)

is said to be practical for NTTP if:

(i) together with the binary condition (4), it defines a valid ILP formulation for NTTP;

(ii) can be separated in polynomial time in the size of $G$ and in the number of nonzero components of the LP solution to be separated;

(iii) the column generation problem for the variables associated with each train $t \in T$ can be carried out by computing an optimal path on the graph $G^t$ with appropriate arc costs.

In other words, (iii) is the natural requirement that the column generation problem has the same structure as the problem of finding the best path for a given train for the original profits.

Requirement (i) is easy to deal with, namely:

**Observation 3** Consider a collection of graphs $F_1, \ldots, F_m$ whose edge intersection yields $F$. The set of constraints

$$\sum_{P \in S} x_P \leq 1, \quad S \in S(F_1) \cup \cdots \cup S(F_m),$$

satisfies (i).

Requirement (ii) needs to be addressed separately case by case (as it is generally the case with complexity issues). As to (iii), it is satisfied if the following technical condition holds.

**Observation 4** Consider a set of constraints of the form (5) such that, for each $t \in T$ and $S \in S'$, either $S \cap \mathcal{P}^t = \mathcal{P}^t$, or there exist $\ell \in L$ and $\bar{A} \subseteq A_\ell$ for which $S \cap \mathcal{P}^t$ is the subset of paths in $\mathcal{P}^t$ containing one arc in $\bar{A}$. This set of constraints satisfies (iii).

**Proof** Assume the constraints are as in the statement. The column generation problem associated with a train $t \in T$ calls for a path $P \in \mathcal{P}^t$ with positive reduced profit, the latter being given by the difference between the profit $\pi_P$ and the sum of the dual values for the constraints in which $x_P$ has coefficient 1. As already mentioned, profit $\pi_P$ is a linear function of the arcs in $P$, say $\pi_a$ is the profit of each arc $a$ in $G^t$. Let $\sigma$ be the sum of the dual values of the constraints $S$ such that $S \cap \mathcal{P}^t = \mathcal{P}^t$. Moreover, for $a \in G^t$, let $\rho_a$ be the sum of the dual values of the constraints $S$ associated with arc sets $\bar{A}$ with $a \in \bar{A}$. The reduced profit of $P$ is given by $\sum_{a \in P} (\pi_a - \rho_a) - \sigma$, i.e., it is a linear function of the arcs in $P$, as required. \qed
2.4 The ILP formulation in [3]

In [3], building on the previous work in [5], we somehow (implicitly) applied Observations 3 and 4 as follows. For a train pair \(\{t_1, t_2\} \subseteq T\), let \(F_T(P_{t_1} \cup P_{t_2})\) and \(F_\ell(P_{t_1} \cup P_{t_2})\) be the subgraphs of \(F_T\) and \(F_\ell\) induced by node set \(P_{t_1} \cup P_{t_2} \subseteq P\). We considered the constraints associated with the stable sets of the edge intersection \(F_T(P_{t_1} \cup P_{t_2}) \cap F_\ell(P_{t_1} \cup P_{t_2})\), called overtaking constraints in [3]:

\[
\sum_{P \in S} x_P \leq 1, \quad \ell \in L, \quad \{t_1, t_2\} \subseteq T, \quad S \in \mathcal{S}(F_T(P_{t_1} \cup P_{t_2}) \cap F_\ell(P_{t_1} \cup P_{t_2})).
\]  

**Proposition 2** Constraints (6) satisfy requirements (i)-(iii) in Definition 1.

**Proof** Extend graph \(F_T(P_{t_1} \cup P_{t_2}) \cap F_\ell(P_{t_1} \cup P_{t_2})\) by including also nodes in \(P \setminus (P_{t_1} \cup P_{t_2})\), connected to all the nodes in \(P_{t_1} \cup P_{t_2}\). The edge intersection of these extended graphs for \(\{t_1, t_2\} \subseteq T\) and \(\ell \in L\) yields \(F\), showing that these constraints satisfy (i). As to (ii), note that \(F_T(P_{t_1} \cup P_{t_2})\) is a complete bipartite graph, and therefore \(F_T(P_{t_1} \cup P_{t_2}) \cap F_\ell(P_{t_1} \cup P_{t_2})\) is a bipartite graph. Therefore, the separation of constraints (6), in optimization form, calls for a maximum-weight stable set of this bipartite graph, considering only the paths corresponding to nonzero components of the LP solution, and can be carried out in polynomial time by flow techniques (see, e.g., [8]). Finally, these constraints satisfy (iii) since, by the maximality of the stable sets, for each path \(P \in P_{t_1} \cap S\) that contains an arc in \(A_\ell\), all other paths in \(P_{t_1}\) containing that arc are also in \(S\) (and the same holds for the paths in \(P_{t_2}\)). \(\square\)

Although constraints (6) are sufficient to define an ILP formulation, they tend to be fairly weak in practice and, in any case, rather slow to separate. For this reason, the following stronger constraints are used in [3]. First of all, we use the obvious constraints associated with the maximal stable sets of \(F_T\):

\[
\sum_{P \in S} x_P \leq 1, \quad S \in \mathcal{S}(F_T),
\]

which, due to Observation 1, read:

\[
\sum_{P \in P_{t_1}} x_P \leq 1, \quad t \in T,
\]  

(i.e., for each train we select at most one path in the solution) and do not need to be separated as they are only \(|T|\). Moreover, we considered the edge-induced subgraph \(F_\ell^d\) of \(F_\ell\) associated with the relaxation of (1):

\[
d_2 \geq d_1 + \alpha_\ell,
\]

and the edge-induced subgraph \(F_\ell^r\) of \(F_\ell\) associated with the relaxation of (1):

\[
r_2 \geq r_1 + \beta_\ell.
\]

It is easy to check that both \(F_\ell^d\) and \(F_\ell^r\) are not only comparability graphs, but also the complement of an interval graph. The corresponding constraints, called departure and arrival constraints, respectively, in [3], read:

\[
\sum_{P \in S} x_P \leq 1, \quad \ell \in L, \quad S \in \mathcal{S}(F_\ell^d),
\]
\[
\sum_{P \in S} x_P \leq 1, \quad \ell \in L, \ S \in S(F_{\l}^T),
\]
and can be separated in linear time (in the size of \(G\) and the number of nonzero variables of the current LP solution). Moreover, it is easy to see that they satisfy the requirement in Observation 4. Only if all these constraints are satisfied, we proceed with the separation of (6).

2.5 A second, natural ILP formulation

A natural formulation which is in fact simpler than the one in [3] (once the structure of \(F\) is clear) is obtained by combining Observation 3 and the fact that \(F\) is the edge intersection of \(F_T\) and \(F_{\ell}\) for \(\ell \in L\), leading to the following constraints. For \(F_T\), we have constraints (7) already mentioned above. As to \(F_{\ell}\), we have the constraints:

\[
\sum_{P \in S} x_P \leq 1, \quad \ell \in L, \ S \in S(F_{\ell}),
\]

which are clearly stronger than (8) and (9).

**Proposition 3** Constraints (7) and (10) satisfy requirements (i)-(iii) in Definition 1.

**Proof** Requirement (i) follows from Observation 3. As to (ii), there are \(|T|\) constraints (7), while from Observation 2 the separation of (10) calls for the determination of a maximum-weight stable set in a comparability graph, considering only the paths corresponding to nonzero components of the LP solution, which can be found efficiently by flow techniques (see, e.g., [8]). Finally, (iii) follows from Observation 4, noting that if a path \(P \in \mathcal{P}\) is in \(S\) for a constraint (10), all other paths in \(\mathcal{P}\) containing arc \(P \cap A_{\ell}\) are also in \(S\). \(\Box\)

2.6 A third natural, and stronger, ILP formulation

A third alternative to constraints (3) can be obtained by merging the main ideas in the previous two ILP formulations: the resulting formulation is stronger than both. Specifically, we consider, for \(\ell \in L\) the edge intersection of \(F_T \cap F_{\ell}\), noting that \(F\) itself is the edge intersection of these \(|L|\) graphs. The constraints that replace (3) in this third formulation are:

\[
\sum_{P \in S} x_P \leq 1, \quad \ell \in L, \ S \in S(F_T \cap F_{\ell}).
\]

These constraints are clearly stronger than (7) and (10) (and also stronger than (6)), and are easily checked to satisfy requirements (i) and (iii) in Definition 1 (see below). On the other hand the complexity of their separation is unclear in general; for instance we do not know an answer to:

**Question 1** What is the complexity of finding a maximum-weight stable set in the edge intersection of a complete multipartite and a comparability graph?

Nevertheless, for the instances in our case study, we are able to devise a polynomial time algorithm, as discussed below.

For each track \(\ell \in L\), let \(T_{\ell} \subseteq T\) denote the set of trains whose path may contain an arc associated with track \(\ell\). We say that travel times are fixed if, for each track \(\ell \in L\) and train \(t \in T_{\ell}\),
there exists a value $\theta_{t,\ell}$ such that all arcs in $A_\ell$ in all paths in $\mathcal{P}_t$ have an arrival time equal to the departure time plus $\theta_{t,\ell}$. Moreover, let $\rho_\ell := \max_{t \in T_\ell} \theta_{t,\ell} - \min_{t \in T_\ell} \theta_{t,\ell}$ be the difference between the maximum and the minimum travel time on track $\ell$. Finally, recall $\alpha_\ell$ and $\beta_\ell$ from (1), the minimum distances between consecutive departures and arrivals, and $|H|$, the number of time instants in the time horizon. The proof of the following result is deferred to the Appendix.

**Proposition 4** If travel times are fixed, a maximum-weight stable set in $F_T \cap F_\ell$ can be found by dynamic programming with time complexity

\[
O \left( \sum_{t \in T_\ell} |\mathcal{P}_t| + |T_\ell| \cdot |H| \cdot (\rho_\ell + \alpha_\ell + \beta_\ell)^3 \cdot (\rho_\ell + 1)^{3|\beta_\ell - \alpha_\ell|} \right)
\]

and space complexity

\[
O \left( |T_\ell| \cdot |H| \cdot (\rho_\ell + \alpha_\ell + \beta_\ell) \cdot (\rho_\ell + 1)^{3|\beta_\ell - \alpha_\ell|} \right).
\]

**Proposition 5** If travel times are fixed and $|\beta_\ell - \alpha_\ell|$ is bounded by a constant, constraints (11) satisfy requirements (i)-(iii) in Definition 1.

**Proof** Requirement (i) follows again from Observation 3, (ii) from Proposition 4, considering only the paths corresponding to nonzero components of the LP solution, and (iii) from Observation 4, noting that if a path $P \in \mathcal{P}_t$ is in $S$ for a constraint (11), all other paths in $\mathcal{P}_t$ containing arc $P \cap A_\ell$ are also in $S$. $\square$

The dynamic programming procedure in Proposition 4 has a fairly high time and space complexity, which make it slow in practice. This can be compared with the time complexity of the separation of the previous constraints, for each $\ell \in L$, still for the case in which travel times are fixed, as discussed in the Appendix:

- $O \left( \sum_{t \in T_\ell} |\mathcal{P}_t| + |T_\ell|^2 \cdot |H| \cdot (\rho_\ell + \alpha_\ell + \beta_\ell) \right)$ for constraints (6), by enumeration;
- $O \left( \sum_{t \in T_\ell} |\mathcal{P}_t| + |H| \right)$ for constraints (8) and (9), by enumeration;
- $O \left( \sum_{t \in T_\ell} |\mathcal{P}_t| + |T_\ell|^3 \cdot |H|^3 \right)$ for constraints (10), by a minimum flow computation.

Note that the asymptotic worst-case time complexity of the minimum flow computation is also fairly high. On the other hand, in this case, the practical average-case complexity is much smaller than the worst-case complexity, while the two essentially coincide for the dynamic programming procedure. Accordingly, we developed two different methods to separate constraints (11) heuristically.

The first heuristic separation procedure is a simple (randomized) greedy heuristic for maximum-weight stable set that, starting from the empty solution, at each iteration selects a node with a probability proportional to the ratio between the weight of the node and the sum of the weights of its neighbors. The node selected is added to the stable set and it is removed from the graph together with all its neighbors.

The second heuristic procedure uses the fact that, as already mentioned in Section 2.5, a maximum-weight stable set in a comparability graph can be found efficiently. Specifically, we consider graph $F_T \cap F_\ell$, along with the comparability graph $F_\ell$ and the associated transitive directed graph $D$. We orient all edges in $F_T \cap F_\ell$ as the corresponding arcs in $D$: the resulting graph $D'$ is
not necessarily transitive, as it has only a subset of the arcs of $D$. Then, we compute the transitive closure $D''$ of $D'$, and finally find a maximum-weight stable set in the comparability graph obtained by ignoring the edge orientations in $D''$. What we obtain is a stable set in $F_T \cap F_\ell$, though not necessarily the one with maximum weight as we have added some edges. On the other hand, given that $D''$ tends to contain notably fewer arcs than $D$, the constraints that we separate in this way tend to be stronger than (10). Moreover, before adding one of these constraints to the current LP, we verify if the associated stable set is maximal for $F_T \cap F_\ell$ and, if not, we add nodes so as to make it maximal. In the sequel we will refer to this heuristic separation method as the transitivization procedure.

3 Computational Results

Our code was implemented in C and run on a PC Intel Core Duo, 2.3 GHz, 2GB RAM, using CPLEX 10.0 as LP solver. We used (our own implementation of) the column generation procedure in [3] and implemented all separation procedures discussed in the previous section, including the (minimum) flow computation required to find a maximum-weight stable set in a comparability graph. Indeed, by taking into account the structure of our instances, our simple implementation widely outperforms the state-of-the-art general-purpose flow codes available.

All the instances considered have a cycle time of one day and, as discussed in the introduction, a sufficiently wide (2-4 minutes) time interval in which nothing is happening to treat them as non-cyclic. For the real-world instances in [5], the number of stations in which overtaking is possible is very large and travel times along tracks in $L$ are very small. This makes it unlikely to have serious interferences between three or more trains along a track, making the formulation in [3] essentially as strong as the new ones proposed in this paper. On the other hand, considering the variation in which overtaking is possible only within the main stations, as it would be highly desirable in practice, the likelihood that interferences involve more than two trains increases significantly, and we can show that the new models provide better bounds. Accordingly, for the computational results reported here, we considered 5 main corridors of the Italian railway network, limiting the set of stations to those in which a crew change is allowed.

We first considered a set of highly-congested small instances. In Table 1, besides the corridor name and the number of trains $|\mathcal{T}|$, we report the upper bounds associated with the LP relaxations of the ILP formulations in the previous section and the associated solution times in seconds. Versions Greedy and Trans. of constraints (11) refer to the heuristic separation of these constraints by the greedy heuristic and the transitivization procedure of Section 2.6, respectively. Not counting these two versions, the left part of the table shows that the quality of the upper bound improves slightly from the LP in [3] to the LP with constraints (7),(10), and significantly from the latter to the LP with constraints (11). On the other hand, the solution time increases by more than one order of magnitude going from one formulation to the other. Still, if we resort to heuristic separation of constraints (11), with the greedy heuristic we get a lower bound which is still much better than those of the first two formulations within a relatively small running time, whereas with the transitivization procedure we get a lower bound which is basically the same as the one obtained by exact separation by dynamic programming, within a running time that is one order of magnitude smaller.

Larger highly-congested instances are considered in Table 2. For these instances, the use of the dynamic programming procedure for the separation of (11) is out of reach. We report the value
Table 1: Upper bound values and solution times for small highly-congested instances.

| Instance     | | | | | |
|--------------|---|----------------|---|----------------|---|----------------|---|----------------|---|
| Bologna-Milano | 12 | 1210.7 | 8 | 1187.9 | 226 | 1110.5 | 11750 | 1135.7 | 202 |
| Bologna-Roma | 12 | 1184.0 | 28 | 1120.1 | 948 | 996.9 | 41050 | 1044.9 | 71 |
| Brennero-Bologna | 12 | 1169.7 | 15 | 1147.9 | 1933 | 1056.5 | 47691 | 1067.7 | 430 |
| Milano-Roma | 12 | 1105.1 | 177 | 1029.6 | 694 | 947.1 | 31010 | 972.5 | 350 |
| Modane-Milano | 12 | 1136.5 | 49 | 1079.1 | 569 | 993.9 | 29479 | 1015.5 | 150 |

Table 2: Upper bound values, solution times, and optimality gaps for larger highly-congested instances.

| Instance     | | | | | |
|--------------|---|----------------|---|----------------|---|----------------|---|----------------|---|
| Bolzano-Verona | 101 | 12455 | 5 | 12685.8 | 8 | 1.8% | 12685.8 | 8 | 1.8% |
| Modane-Milano | 59 | 4876 | 6 | 5382.4 | 9 | 9.4% | 5329.1 | 41 | 8.5% |
| Munich-Verona | 54 | 4044 | 6 | 4191.5 | 14 | 3.5% | 4191.5 | 14 | 3.5% |
| Piacenza-Bologna-a | 39 | 3666 | 6 | 3882.3 | 57 | 5.6% | 3871.6 | 83 | 5.3% |
| Piacenza-Bologna-b | 91 | 9507 | 7 | 9820.6 | 80 | 3.2% | 9819.9 | 82 | 3.2% |
| Piacenza-Bologna-c | 57 | 5550 | 10 | 5905.6 | 183 | 6.0% | 5886.9 | 427 | 5.7% |
| Piacenza-Bologna-d | 210 | 16216 | 17 | 19270.8 | 27225 | 15.8% | 19243.3 | TL | 15.7% |

Table 3: Upper bound values, solution times, and optimality gaps for real-world instances related to those in [5].
References


Appendix: Separation when Travel Times are Fixed

As discussed in [3], a constraint (8) is uniquely identified by the first time instant $d_{\text{first}}$ of departure of paths in $\mathcal{P}$ in the constraint, i.e., there are $|H|$ constraints (8) for each $\ell \in L$. The left-hand-sides of all these constraints can be computed in constant time per constraint after having defined, in $O \left( \sum_{t \in T_\ell} |\mathcal{P}_t| + |H| \right)$ time, for each time instant $h \in H$, the sum of the weights of the paths in $\mathcal{P}$ departing at $h$ on track $\ell$. The same applies to constraints (9).

As also discussed in [3], once $\{t_1, t_2\} \subseteq T_\ell$ is specified, a constraint (6) is uniquely identified by the first time instants $d_{\text{first}}^1$ and $d_{\text{first}}^2$ of departure of paths in $\mathcal{P}^{t_1}$ and $\mathcal{P}^{t_2}$, respectively, in the constraint. There are $|H|$ possible values for $d_{\text{first}}^1$ and, having fixed this, $O(\rho_\ell + \alpha_\ell + \beta_\ell)$ for $d_{\text{first}}^2$ if one wants at least one path in $\mathcal{P}^{t_1}$ and one path in $\mathcal{P}^{t_2}$ in the constraint. In other words, for each $\ell \in L$ there are $O(|T_\ell|^2 \cdot |H| \cdot (\rho_\ell + \alpha_\ell + \beta_\ell))$ constraints (6). Again, the left-hand-sides of all these constraints can be computed in constant time per constraint after having defined, for each time instant $h \in H$ and train $t \in \{t_1, t_2\}$, the sum $w_{h,t}$ of the weights of the paths in $\mathcal{P}^t$ departing at $h$ on track $\ell$. The overall time complexity is then $O \left( \sum_{t \in T_\ell} |\mathcal{P}_t| + |T_\ell|^2 \cdot |H| \cdot (\rho_\ell + \alpha_\ell + \beta_\ell) \right)$.

The separation of constraints (10) calls for a maximum-weight stable set, say $S$, in the comparability graph $F_\ell$. It is easy to check that, in case travel times are fixed, if a path $P \in \mathcal{P}^t$ is in $S$, then all paths in $\mathcal{P}^t$ departing on $\ell$ at the same time instant as $P$ are in $S$. Accordingly, one
can find a maximum-weight stable set in the graph with one node \(n_{h,t}\) of weight \(w_{h,t}\) (as defined above) for each \(t \in T_\ell\) and \(h \in H\), and with edges connecting nodes \(n_{h_1,t_1}\) and \(n_{h_2,t_2}\) if a departure of \(t_1\) at \(h_1\) and a departure of \(t_2\) at \(h_2\) respect the track capacity constraints on \(\ell\). The number of nodes is \(O(|T_\ell| \cdot |H|)\), and the graph may be dense. The computation of a stable set on a comparability graph can be carried out by finding a minimum flow from a dummy source to a dummy sink in the associated transitive acyclic directed graph (with lower bounds on the node flows equal to the node weights and infinite arc capacities, see, e.g., [8]). For a dense graph, the best asymptotic time complexity for a minimum (or maximum) flow computation is cubic in the number of vertices. Including the definition of the node weights, this leads to an overall time complexity of 

\[
O\left(\sum_{t \in T_\ell} |P^t| + |T_\ell|^3 \cdot |H|^3\right).
\]

**Proof of Proposition 4** For simplicity, we first illustrate the dynamic programming procedure for the case \(\alpha_\ell = \beta_\ell\). In this case, we consider the trains by increasing value of travel time \(\theta_{t,\ell}\), and, to simplify the notation, we assume \(T_\ell = \{1, \ldots, |T_\ell|\}\), where a larger train index corresponds to a larger travel time. A state in the dynamic programming procedure, associated with a stable set \(S\) of \(F_T \cap F_\ell\), is represented by the triple \((t, d_{\text{min}}, r_{\text{max}})\), where:

- \(t\) is the last train considered, i.e., \(S\) contains only paths in \(P^1 \cup \cdots \cup P^t\);
- \(d_{\text{min}}\) is the smallest departure time on \(\ell\) of a path in \(S\);
- \(r_{\text{max}}\) is the largest arrival time on \(\ell\) of a path in \(S\).

We denote by \(w(t, d_{\text{min}}, r_{\text{max}})\) the weight of the best such \(S\).

Once all states associated with trains \(1, \ldots, t - 1\) have been computed, we consider the stable sets that can be obtained by adding paths in \(P^t\) to these states. It is easy to check that, in order to find maximal stable sets, we are only interested in the addition of subsets of \(P^t\) defined by a pair \((d_{\text{first}}, d_{\text{last}})\), corresponding to all paths in \(P^t\) whose departure time on \(\ell\) is between \(d_{\text{first}}\) and \(d_{\text{last}}\) (and therefore whose arrival time is between \(d_{\text{first}} + \theta_{t,\ell}\) and \(d_{\text{last}} + \theta_{t,\ell}\)). Let \(g_t(d_{\text{first}}, d_{\text{last}})\) denote the weight of these paths. Also easy is to check that a subset \((d_{\text{first}}, d_{\text{last}})\) can be combined with a state \((t - 1, d_{\text{min}}, r_{\text{max}})\) (i.e., the union of the two is a stable set) if and only if (a) \(d_{\text{last}} < d_{\text{min}} + \alpha_\ell\) (a path for \(t\) departing at a time instant \(\geq d_{\text{min}} + \alpha_\ell\) and a path for any train \(< t\) departing at \(d_{\text{min}}\) respect the track capacity constraints on track \(\ell\)), and (b) \(d_{\text{first}} > r_{\text{max}} - \theta_{t,\ell} - \beta_\ell\) (a path for \(t\) departing at a time instant \(\leq r_{\text{max}} - \theta_{t,\ell} - \beta_\ell\) and a path for any train \(< t\) arriving at \(r_{\text{max}}\) respect the track capacity constraints on track \(\ell\)). The resulting state is \((t, \min\{d_{\text{min}}, d_{\text{first}}\}, \max\{r_{\text{max}}, d_{\text{last}} + \theta_{t,\ell}\})\).

Accordingly, in the dynamic programming recursion for train \(t\), we first set \(w(t, d_{\text{min}}, r_{\text{max}}) := 0\) for all values \(d_{\text{min}}, r_{\text{max}}\). Then, we consider all states \((t - 1, d_{\text{min}}, r_{\text{max}})\) and subsets \((d_{\text{first}}, d_{\text{last}})\) such that (a) and (b) above hold, and, if

\[
w(t - 1, d_{\text{min}}, r_{\text{max}}) + g_t(d_{\text{first}}, d_{\text{last}}) > w(t, \min\{d_{\text{min}}, d_{\text{first}}\}, \max\{r_{\text{max}}, d_{\text{last}} + \theta_{t,\ell}\}),
\]

we set

\[
w(t, \min\{d_{\text{min}}, d_{\text{first}}\}, \max\{r_{\text{max}}, d_{\text{last}} + \theta_{t,\ell}\}) := w(t - 1, d_{\text{min}}, r_{\text{max}}) + g_t(d_{\text{first}}, d_{\text{last}}).
\]

After having considered all trains, the maximum of \(w(t, d_{\text{min}}, r_{\text{max}})\) over all values \(t, d_{\text{min}}, r_{\text{max}}\) is the maximum weight of a stable set in \(F_T \cap F_\ell\). (The obvious details about how to store and reconstruct this stable set are omitted.) There are up to \(|H|\) possible values for \(d_{\text{min}}\), and, for a given \(d_{\text{min}}\), the
number of possible values of \( r_{\text{max}} \) such that there may be paths associated with at least two trains in a state \((t, d_{\text{min}}, r_{\text{max}})\) is \(O(\rho_\ell + \alpha_\ell + \beta_\ell)\). This proves the space complexity (for \( |\beta_\ell - \alpha_\ell| = 0 \)). The time complexity follows from the fact that the number of subsets \((d_{\text{first}}, d_{\text{last}})\) such that (a) and (b) above may hold for a given state is \(O((\rho_\ell + \alpha_\ell + \beta_\ell)^2)\), and that all weights \(g_t(d_{\text{first}}, d_{\text{last}})\) can be computed in time linear in the number of weights and in \(\sum_{t \in T} |P^t|\) in a preprocessing phase.

The generalization of the method for the case \(\alpha_\ell \neq \beta_\ell\) is straightforward, but much more tedious to state in detail and (mainly) with a much higher space and time complexity. For instance, consider the case \(\beta_\ell > \alpha_\ell\) (the case \(\beta_\ell < \alpha_\ell\) is analogous). A state is now represented by \((t, d_{\text{min}}, r_{\text{max}}, r_{\text{min}}^0, \ldots, r_{\text{min}}^{\beta_\ell - \alpha_\ell - 1})\), where \(r_{\text{min}}^i\) is the smallest arrival time of the paths in \(S\) departing at time \(d_{\text{min}} + i\) for \(i = 0, \ldots, \beta_\ell - \alpha_\ell - 1\). The time and space complexity in this general case follow from the fact that there are \(\rho_\ell + 1\) possible values for \(r_{\text{min}}^i\). \(\square\)