Effective lower bound for job-shop scheduling in railway traffic control

Carlo Mannino* Alessandro Mascis*

*Department of Computer and System Sciences, Sapienza University of Rome
Via Ariosto 25, 00185 Rome, Italy

*Bombardier Transportation
Via Cerchiara 125, 00131 Roma, Italy

Abstract

Train movements across railway stations are still operated by human dispatchers. Motivated by an application provided by ATM, the major Italian municipal transport company, we developed a real-time automated traffic control system to operate trains in metro stations. The system optimally controls the trains in a metro station by identifying a suitable routing and by establishing an optimum schedule of the performed operations. For each candidate routing an instance of the blocking, no-wait job-shop scheduling problem with convex costs is solved to optimality by branch and bound. We developed an effective lower bound, based on minimum cost flow computations, in order to speed up the enumeration process. Computational testing in a real environment proved that the algorithm is able to solve relevant practical instances within the very tight time limit imposed by the application.

Keywords: *job-shop scheduling, disjunctive programming, traffic control system*

1 Introduction

Train movements across railway and metro stations are managed by integrated systems, namely Operation Control Centers, which include decision support software, remote control equipment and human operators. Worldwide this sector is worth over one billion euros. In Italy alone, control centers supervise more than 2000 stations. Trains move across stations in order to perform a number of prescribed operations (*services*), such as arrival and departure, unload and pick-up of passengers, etc. Train movements are in most cases still operated by human dispatchers. They establish suitable routes for each of the trains in the station. Such routes go through track segments devoted to the specific services required by the train. Dispatchers are able to guide trains by re-orienting track switches and by switching signals status: While doing this, they also enforce a timing (schedule) of the different operations performed by each train. Final outcome is a real-time plan, that is a family of routes, one for each of the trains, plus a schedule of the operations performed by the trains. The (Station) Traffic Control Problem (TCP) is the problem of generating a real-time plan maximizing (a weighted sum of) punctuality and regularity.

The application which motivated our research was provided by the Azienda Trasporti Milanesi (ATM), the major Italian municipal transport company, which manages the public transport service of the city of Milan and other 85 towns in the district. ATM intended to develop a fully automated traffic control system both to increase traffic performances and to save on operating costs. Within this project, we developed an algorithm (OptPlan) to solve the TCP to optimality.

A natural decomposition approach to the solution of the TCP consists in first selecting a routing for the trains, and then calculating a schedule for the operations. Since trains compete for using scarce

(single capacity) resources (such as tracks and platforms), this problem falls into the class of *job-shop scheduling* problems, where trains can be viewed as *jobs* and tracks and platforms are *machines*. Namely, we deal with an instance of *blocking, no wait job-shop scheduling*.

Job-shop scheduling problems can be represented and solved by Mixed Integer Linear Programming (MILP). A classical representation is based on *disjunctive programming* (see [4]) and makes use of one continuous variable for each of the operations to represent its starting time; precedence relations between operations asking for common resources are imposed by disjunctive constraints. In our approach, all feasible routings are enumerated, and, for each routing, an instance of the blocking, no-wait job-shop scheduling problem is solved to optimality by exploiting the corresponding disjunctive formulation. In particular, we make use of an extension of the classical representation of job-shop scheduling instances by the so called *disjunctive graph*, introduced by Balas in [3]. As in an early work by Balas, properties of the (extended) disjunctive graph are exploited to drive the enumeration process, and also to derive a new lower bounding procedure which allows us to find optimum plans within the one second time limit imposed by the application.

After a rigorous and extensive test-campaign, the automatic route setting system was put into operation on July 2007 in Milano metro system. A full version of this paper can be found in [6].

2 The Problem

Metro Stations. A terminal metro station is a facility where passengers may board and alight from trains, and in which trains can reverse direction or perform a number of additional operations. Such operations are called *train services*. A metro station can be viewed as a set of *track segments*, the minimal controllable rail units, which in turn may be distinguished into *stopping points* and *interlocking-routes*. A stopping point is a track segment in which a train can stop to execute a service while an interlocking-route is the rail track between two stopping points, and is actually formed by a sequence of track segments. For our purposes, a metro station is represented by means of a directed graph $M = (P, I)$ where P is the set of *stopping nodes* (points) and $I \subseteq P \times P$ is the set of *interlocking arcs* (routes). A performable service is associated with every stopping node $p \in P$.

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

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

Scheduling Nodes and arcs of a route r correspond to rail tracks. In order to provide a complete description of the movements of a train along its route r, we need to establish the exact time when the train enters each track, or, equivalently, a *starting time* for all of the nodes and arcs of r. Now, let $a = (u, v) \in r$, and let t_u, t_v, t_a denote the starting times of nodes u, v and arc a, respectively: then, since the train enters stopping point u *before* running interlocking-route a, it must be $t_a - t_u \geq d_u$ (*precedence constraint*). Also, since a train cannot be stopped while running through an interlocking-route (*no-wait constraint*), we have $t_v - t_a = d_a$. If $R \in \mathcal{R}(M, T)$ is a feasible routing, an assignment of starting times to all nodes and arcs of all routes in R is called a *schedule* for R.

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

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

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

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

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

$$c'(s) = \sum_{j=1}^{|T|} g(s_j) + \sum_{j=1}^{|T|} f(s_j - s_{j-1}). \tag{2}$$

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

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

In short, the TCP is tackled by enumerating all feasible routings in $\mathcal{R}(M,T)$ and then by solving, for each $R \in \mathcal{R}(M,T)$, the associated job-shop scheduling problem. Therefore, for any $R \in \mathcal{R}(M,T)$, we have a set of operations $N = N(R) = \{0, \ldots, n\}$, where 0 is a dummy operation (called *start*), while the operations $\{1, \ldots, n\}$ correspond to the stopping nodes and the interlocking arcs of all of the routes in R. With every $i \in N$ we associate a *starting time* $t_i \in \Re$. The vector $t \in \Re^{n+1}$ is called a *schedule* of N, and we assume $t_i - t_0 \geq 0$, for all $i \in N$. The departure time s_j of Train $j \in T$ is related to the starting time of the exit node $d(r_j)$ of r_j through the equation $s_j = t_{d(r_j)} - t_0$, for $j \in T$.

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

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

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

The job-shop scheduling problem is NP-hard and can be formulated as the following *disjunctive program* (see Balas [4]):

Problem 2.3

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

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

A *selection* $S \subseteq Z(A)$ is a set of arcs obtained from A by choosing at most one arc from each pair. The selection is *complete* if $|S \cap \{(i,j),(h,k)\}| = 1$ for all $((i,j),(h,k)) \in A$, i.e. exactly one arc from each pair is chosen. Every selection S of $D(N,F,A)$ naturally defines a new disjunctive graph $D[S] = (N, F_S, A_S)$, where $F_S = F \cup S$, while A_S is obtained from A by removing the pairs corresponding to the arcs in S. We call $D[S]$ an *extension* of D under S. Finally, we associate with $D(N,F,A)$ the weighted directed graph $G(D) = (N,F)$, with length l_{ij} associated with every $(i,j) \in F$.

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

3 Solution algorithm and lower bound computation

An instance of the TCP is solved by our algorithm by enumerating all of the feasible routings $R \in \mathcal{R}(M,T)$ and by solving, for each R, the associated instance (D^R,l,c) of the job-shop scheduling problem (2.3). This task is carried out by implicitly enumerating all of the feasible extensions of D^R. However, the enumeration of the (partial) extensions of D can be limited by the following standard arguments. Let UB be any upper bound to the optimum solution value of Problem (2.3) - e.g., the cost $c(\hat{t})$ of any known feasible solution $\hat{t}$ - and let S be a (partial) selection of A. If the optimum solution value $z^*(D[S],l,c)$ to $SCH(D[S],l,c)$ satisfies $z^*(D[S],l,c) \geq UB$ then no (complete) extension of $D[S]$ can improve on $\hat{t}$ and the problem can be disregarded. Now, Problem $SCH(D[S],l,c)$ is an instance of the so called *optimal potential problem* with convex costs ([8]), which can be shown to be the dual of a min-cost flow problem with convex costs and can be solved efficiently even in its integer version ([2]). Since a lower bound computation must be carried out at each branching, we studied a further relaxation to $SCH(D,l,c)$ which proved to be effective in reducing the size of the enumeration tree with very little computational effort.

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

Lemma 3.1 [6] *Let $W \subseteq N$, with $W \neq \emptyset$. Then*

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

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

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

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

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

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

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

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

We show now how to reduce $REL(D,l,c)$ to a min-cost flow problem on a graph with $|T|+2$ vertices and only a few arcs. To do this we introduce a variable vector $y \in \Re^{|T|}_+$ and we let $y_j = s_j - s_{j-1}$, for $j \in T$. Observe that, for $j \in T$, we have $s_j = y_1 + \ldots + y_j$. Then $REL(D,l,c)$ can be rewritten as

$$\begin{aligned} \min \quad & \textstyle\sum_{j=1}^{|T|} g_j(\sum_{i=1,\ldots,j} y_i) + \sum_{j=1}^{|T|} f_j(y_j) & & \\ & y_j \geq q_j & j \in T & \quad (4) \\ & y_1 + \ldots + y_j \geq l_j & j \in T & \quad (5) \\ & y_1 + \ldots + y_j \leq u_j & j \in T & \quad (6) \\ & y \in \Re^{|T|} & & \end{aligned}$$

The above program can be easily transformed into an equivalent min-cost flow on a suitable directed network $H(V,E)$ with lower and upper capacities on the arcs. Namely, let $V = \{0,1,\ldots,|T|+1\}$ and let $E = E_1 \cup E_2 \cup \{(|T|+1,0)\}$, where $E_1 = \{(0,j) : j \in T\}$ and $E_2 = \{(j,j+1) : j \in T\}$. With every arc $(0,j) \in E_1$, we associate lower capacity $q_j \geq 0$. With every arc $(j,j+1) \in E_2$, we associate lower

capacity $l_j \geq 0$ and upper capacity $u_j \geq 0$. Now, let $x \in R_+^{|E|}$ be a circulation of $H(V, E)$ satisfying lower and upper capacities. Then $\bar{y}_j = \bar{x}_{0j}$, for $j \in T$ satisfies (4), (5) and (6). In fact, the lower capacity on arc $(0, j)$ implies that $\bar{x}_{0j} = \bar{y}_j \geq q_j$, for all $j \in T$, and (4) are satisfied. Also, observe that $\bar{y}_1 + \ldots + \bar{y}_j = \bar{x}_{j,j+1}$, for $j \in T$, and the lower and upper capacities on arcs $(j, j+1)$, for $j \in T$, ensure that (5) and (6) are satisfied. Thus, if we associate the cost $f_j(x_{0j})$ with the flow variables x_{0j}, for $j \in T$, the cost $g_j(x_{j,j+1})$ with the flow variables $x_{j,j+1}$, for $j \in T$, and zero cost with the remaining variable $x_{|t|+1,0}$, the original problem is finally reduced to finding a circulation of H of minimum cost.

An instance of the min-cost flow problem with piecewise linear convex costs can be easily transformed into an equivalent one with linear costs by substituting each arc with a number of parallel arcs equal to the number of break points of the associated cost function (see [1]). This corresponds to adding at most $5 \cdot |T|$ arcs to the original network H.

4 Computational results

In order to evaluate the overall approach to the TCP, we performed both static and run-time (real-life) tests (see [6]). Static tests involve a single trains list, and were carried out mainly for assessing the quality of the relaxations and of the branch and bound algorithm. The results clearly show that, when compared to $SCH(D, l, c)$, solving the min-cost flow reformulation of $REL(D, l, c)$ speeds the computing times up to 2.5 times, a very desirable feature for real-time applications. Indeed, an instance of $REL(D, l, c)$, particularly in its min-cost flow reformulation, can be solved (by using the Goldberg and Tarjan code [5]) much more efficiently than the original $SCH(D, l, c)$ instance (solved by CPLEX 10.0); in contrast, the total number of branching nodes increases only slightly. The results become even more impressive when compared with other classical approaches, such as those based on time-indexed reformulations ([7]). Run time tests were performed to evaluate the ability of the system to manage real-time traffic and compare its performances to those obtained by human dispatchers and were done during an official test-campaign, which lasted several days. The results shows that dispatchers were in most cases outperformed by the system. This favorable comparison is confirmed by the average result, which shows an increase of more than 8% in a cumulative measure agreed with the ATM engineers (see [6]).

References

[1] Ahuja, R.K., T.L. Magnanti, J.B. Orlin, *Network Flows*, Prentice-Hall, 1993.

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

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

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

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

[6] G C. Mannino, A. Mascis, Real-Time Traffic Control in Metro Stations, *Operations Research* (2009) to appear.

[7] Queyranne, M., A.S. Schulz, *Polyhedral Approaches to Machine Scheduling*, Tech Rep. 408/1994, Technische Universitat Berlin, 1994.

[8] Rockafeller, R.T., *Network flows and monotropic optimization*, John Wiley & Sons, 1984.