Effects of flexible timetables in real-time scheduling of rail operations

Andrea D'Ariano* Dario Pacciarelli† Marco Pranzo‡

*Department of Transport and Planning - Delft University of Technology, The Netherlands
Stevinweg 1 - P.O. Box 5048
2600 GA Delft, The Netherlands
a.dariano@tudelft.nl

†Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre
Via della Vasca Navale 79
00146, Roma, Italy
pacciarelli@dia.uniroma3.it

‡Dipartimento di Ingegneria dell'Informazione, Università di Siena
via Roma, 56
53100 Siena, Italy
pranzo@dii.unisi.it

1 Introduction

A widespread strategy to regulate railway traffic in congested areas consists operating in real-time with strict adherence to an off-line developed timetable. However, unforeseen events may require to partially modifying the planned timetable, and this real-time process is called *Conflict Detection and Resolution* (CDR). Heuristic and exact algorithms for CDR have been proposed, e.g. by [10, 4, 3].

Time reserves are usually inserted in the timetable to reduce the effects of minor perturbations. Larger time reserves allow to increase train punctuality, but reduce the capacity of the lines, i.e. the maximum number of trains per hour. Hence, in congested areas, the amount of time reserves that can be inserted in the timetable is limited. In order to improve the reliability of a timetable, without decreasing the capacity of the lines, a recent trend in the Dutch railway companies [6, 12] is for managing congested areas by planning less detailed in the off-line phase, and solving all possible conflicts among trains in real-time. The new concepts of dynamic traffic management [11] and *flexible timetable* belong to this context. Specifically, a traditional rigid timetable specifies, for each train and each station, the platform track, the arrival/departure times and the total order of trains at each merging and crossing point in the network. A flexible timetable defines (i) a set of feasible platform tracks for each train at each station, (ii) time windows of [minimum, maximum] arrival/departure times for the trains at

a set of relevant points (a larger time window corresponding to having more flexibility) and (*iii*) a partial order of trains at each merging and crossing point. The complete definition of these three elements is postponed to real-time management. The idea is that the larger degree of freedom, let by flexible timetables to the real-time control system, offers the possibility to better manage rail traffic.

In this paper, we focus on points (*ii*) and (*iii*), and study the effectiveness of flexible arrival/departure times as a strategy for improving the timetable robustness. We investigate the relations between flexibility, timetable robustness and delay minimization.

In order to carry out the analysis, we evaluate the effects of different perturbations occurring in a network under rigid and flexible timetables. To make the comparison independent from the particular CDR system, we compute proven optimal solutions and compare train delays when varying the timetable robustness and the flexibility. To this aim, we model the CDR problem as a job shop scheduling problem with additional constraints, and formulate it by means of an *alternative graph* [7]. A branch and bound procedure allows finding optimal solutions to practical size instances within short time limits. We carry out an extensive computational study based on a bottleneck area of the Dutch rail network.

2 Models and algorithms

In this section, we model the CDR problem by means of the alternative graph formulation [7]. Railway networks, under the fixed block signaling system, are organized into track segments called *block section*. Each block section can host at most one train at a time. A *conflict* occurs whenever two or more trains require the same block section at the same time. The *CDR problem* is the real-time problem of finding a conflict free schedule compatible with the real-time status of the network and such that trains arrive and depart with the smallest possible delay. Note that, our definition of the CDR problem is slightly different from the one most used in practice. In fact, while the common practice is to detect and solve conflicts *one at a time*, in our definition the aim of CDR is to develop a *new conflict free plan*, i.e. we consider the overall CDR problem as a unique problem to be solved in the best way.

The combinatorial structure of the CDR problem is similar to that of the no-store job shop scheduling problem [8], where jobs correspond to trains and machines to block sections. We formulate the CDR problem with an alternative graph. This consists of a triple $\mathcal{G} = (N, F, A)$, where N is a set of nodes, F is a set of fixed arcs and A is a set of pairs of alternative arcs. A feasible solution is obtained by choosing a set S of arcs, exactly one for each alternative pair, in such a way that graph $G(S) = (N, F \cup S)$ is acyclic. An optimal solution S^* minimizes the length of the longest path in $G(S^*)$ from a dummy node 0, called *start node*, which precedes all other nodes, to a dummy node n, called *finish node*, which follows all other nodes.

An *operation* is the passing of train T_i through a particular block section S_j, and is associated to a *node* $ij \in N$ of the alternative graph. Therefore, the route of T_i is a chain of nodes. An arc (ij, hk) represents a *precedence relation* between two nodes ij and hk. The *starting time* t_{hk} of operation hk is constrained to be $t_{hk} \geq t_{ij} + p_{ijhk}$, where p_{ijhk} is the length of arc (ij, hk). Fixed arcs correspond to precedence constraints between consecutive operations of the same train, the arc weights indicating the *traversing time* of the train through the block

sections. Since a block section cannot host two trains at the same time, whenever two jobs require the same block section, there is a potential conflict. In this case, a processing order must be defined between the incompatible operations, and we model it by introducing in the graph a suitable pair of *alternative arcs* (see Figure 1 (a) and (b)). Each alternative arc models a possible precedence between two operations. Its length, that we call *setup time*, is the time between the entrance of the head of a train in a block section and the exit of its tail from the previous one, plus additional time margins [9].

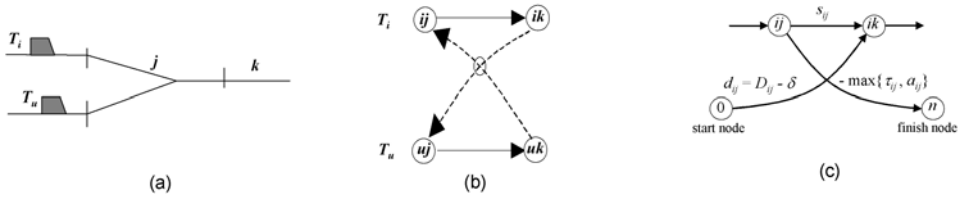

Figure 1: Alternative graphs for two trains (b) at a junction (a) and a train at a station (c)

The CDR problem consists of assigning a *starting time* t_{ij} to each node $ij \in N$, i.e. of planning the exact time each train will enter each block section, such that all fixed precedence relations, and exactly one for each pair of the alternative precedence relations, are satisfied. By denoting with $l^S(ij, hk)$ the length of a longest path from node ij to node hk in $\mathcal{G}(S)$, a feasible assignment of starting times to nodes is $t_{ij} = l^S(0, ij)$, for $ij \in N$.

The objective of the CDR problem is to minimize the starting time of the finish node $t_n = l^S(0, n)$. In what follows, we formally define this objective function. With a flexible timetable, for each train T_i stopping at platform S_j, a *maximum arrival time* a_{ij}, a *minimum dwell time* s_{ij} and a *minimum departure time* d_{ij} are associated. T_i is not allowed to depart from S_j before d_{ij}, and is considered late if arriving after a_{ij}. The *total delay* of T_i at S_j is the difference between its arrival time t_{ij} and a_{ij}. We partition the total delay in two parts as follows. Let τ_{ij} be the earliest possible arrival time of T_i at S_j, computed when the train travels at maximum speed and disregarding the presence of other trains. If $\tau_{ij} > a_{ij}$, then the quantity $\tau_{ij} - a_{ij}$ is an unavoidable delay that cannot be recovered by real-time rescheduling train operations. We call $\max\{0, \tau_{ij} - a_{ij}\}$ the *initial delay* of the train. The quantity $\max\{0, t_{ij} - \max\{\tau_{ij}, a_{ij}\}\}$ is called *consecutive delay*, which is due to the solution of conflicts with the other trains in the network. We do not take into account initial delays in the scheduling phase, and define a *modified due date* for T_i at S_j equal to $\max\{\tau_{ij}, a_{ij}\}$. The alternative graph formulation of the arrival/departure time of T_i at S_j is shown in Figure 1 (c), where s_{ij} is the minimum dwell time and d_{ij} is the minimum departure time of T_i at S_j.

We define the quantity $\delta = \min_{i,j}\{\delta_{ij} = a_{ij} + s_{ij} - d_{ij}\}$ as the timetable *flexibility*. In a rigid timetable, $\delta = 0$ holds. In our computational experiments we study the effects of varying δ, starting from a rigid timetable. To ensure fairness in the comparison, we set the maximum arrival times a_{ij} equal to the arrival times of the rigid timetable, since increasing these values would immediately result in reduced train delays without changing their actual arrival/departure times. Thus, increasing the flexibility δ corresponds to setting the minimum departure times d_{ij} equal to the rigid departure times D_{ij} minus δ, i.e. $d_{ij} = D_{ij} - \delta$ (see Figure 1 (c)). In other words, the effect of a larger flexibility δ simply corresponds to the possibility to depart earlier than planned in the rigid timetable.

We implemented four algorithms for the CDR problem: two simple dispatching rules that we use as a surrogate for the human dispatcher behavior, a simple greedy heuristic based on global information and a branch and bound algorithm.

The first dispatching rule is the First Come First Served (FCFS) dispatching rule, commonly adopted in railway practice, which simply consists of giving precedence to the train arriving first at a block section. The second dispatching rule, called First Leave First Served (FLFS), is as follows. When two trains approach a block section, the time required for each train to enter and traverse it is first computed. Then, precedence is given to the train which is able to leave the block section first. The greedy heuristic is the algorithm AMCC described in [7], which selects one alternative arc at the time, avoiding the one which would cause the largest delay.

The branch and bound algorithm is described in details in [3]. The lower bound is an adaptation to the railway scheduling problem of the single machine Jackson's Preemptive Schedule [1]. Implication rules are used to speed up the branch and bound algorithm. We implemented the implication rules of Carlier and Pinson [2], which are dynamically computed during the execution of the solution procedure. We also develop static implication rules, which are computed off-line on the basis of the topology of the rail network, which allow to obtain a significant speed up of the branch and bound algorithm [3].

The branching scheme used in our algorithm is binary and consists of selecting an alternative pair of arcs according to the AMCC rule. The search strategy is a mixed strategy in which we alternate some repetitions of the depth-first visit to the selection of the node with the smallest lower bound.

3 Computational experiences

In this section, we report our results on practical size instances based on the dispatching area of Schiphol (Figure 2), a bottleneck area of the Dutch rail network. This includes 86 block sections, 16 platforms, two largely independent traffic directions and mixed passenger/freight traffic.

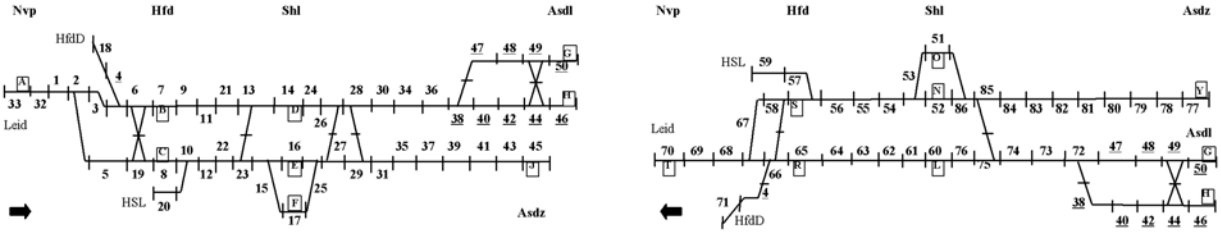

Figure 2: The Schiphol dispatching area network

We considered typical rolling stock characteristics and the provisional Dutch timetable for 2007 [5], a rigid cyclical timetable with 54 trains circulating every hour. Timetable time reserves are divided into *recovery time* (obtained by planning trains traveling at less than maximum speed and designed to recover initial delays) and *buffer time* (obtained by inserting an extra

separation time between consecutive trains and designed to reduce consecutive delays).

We construct a flexible timetable by replacing the rigid arrival/departure times with flexible windows of [minimum, maximum] arrival/departure times. To ensure fairness when comparing the results for different values of the flexibility, we set the maximum arrival times constant in all experiments and equal to the rigid arrival times. We set the minimum departure times equal to the rigid departure times minus $\delta = \{0, 30, 60, 90, 120\}$, the value $\delta = 0$ corresponding to the rigid timetable. Hence, four flexible timetables are generated with different flexibility.

We study the network by simulating different real-time traffic conditions, and computing every time a rescheduling of train operations. We model initial perturbations by letting some trains enter late the network. Different instances are generated by varying the number of delayed trains from 7 to 27 and setting the input delay in accordance with a Gaussian or uniform distribution in a range $[0, max]$, for max varying from 200 to 1800 seconds. In total, we generate 60 perturbation schemes. For each perturbation scheme, we generate an instance of the CDR problem for $\delta = \{0, 30, 60, 90, 120\}$. The algorithms are executed on a laptop equipped with a 1.6 GHz processor. We evaluate the solutions in terms of maximum and average consecutive delays, since initial delays cannot be avoided.

Figure 3 shows the maximum and average consecutive delays obtained by the four algorithms for different values of the flexibility δ. Each value reported in the figure is the average over the 60 perturbation schemes. For some instances one or more of the three initial algorithms may fail in the computation of a feasible solution. In order to compare the different results, for each failure we consider a penalty of 10 minutes for the maximum consecutive delay and 1 minute for the average consecutive delay.

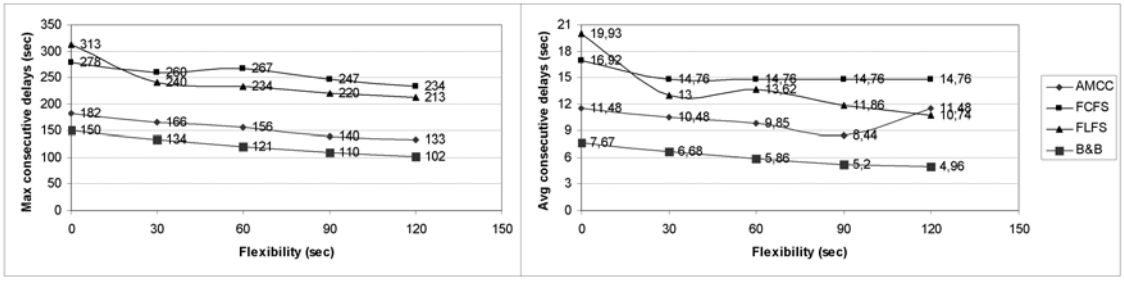

Figure 3: Maximum and average consecutive delays for the four algorithms

As expected, all the four algorithms take advantage of an increasing flexibility, even if the three initial heuristics exhibit less consistent behaviors. The truncated branch and bound evidently outperforms the other algorithms. It finds proven optimal solutions in 297 cases out of 300 within the time limit of 120 seconds, and the average computation time is 1.94 seconds. The First Come First Served (FCFS) and the First Leave First Served (FLFS) rules, being the most "local" rules, give the worst results. Both maximum and average consecutive delays are more than double with respect to the values found by the truncated branch and bound. The solution found by AMCC is similar to the optimal solution, and almost always better then the solutions found by FCFS and FLFS.

Figure 4 shows the maximum and average consecutive delays obtained by the truncated branch and bound algorithm when varying the flexibility δ and the timetable robustness. Starting

from the cyclical timetable of the first set of experiments, we generate a second, less robust, timetable by removing all buffer times. The three curves in Figure 4 show the propagation of the 60 perturbation schemes occurring in the first hour over the first, second and third hours of traffic.

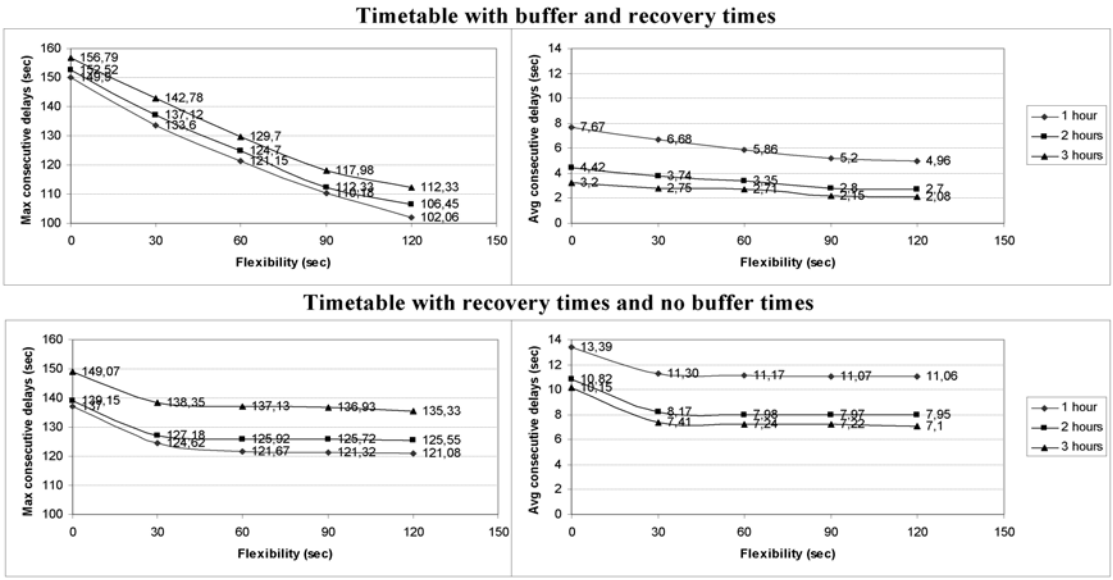

Figure 4: Maximum and average consecutive delays with or without buffer times

For the timetable with buffer times the perturbations of the first hour propagate modestly to the second and the third hours. On the contrary, for the timetable without buffer times the perturbations of the first hour propagate significantly up to the third hour. The advantage of flexibility is especially relevant for the more robust timetable, since a flexibility of 120 seconds produces approximately 30% of reduction in the maximum consecutive delay. Flexibility becomes less effective when the timetable does not include sufficient time reserves. Specifically, the timetable with no buffer time does not allow to take advantage from a flexibility $\delta > 30$ seconds.

4 Conclusions

Computational experience shows that the flexible timetable is preferable to the rigid one, since it offers more freedom to solve conflicts and increases punctuality without decreasing the throughput. The use of advanced optimization algorithms for the CDR problem improves the benefits of flexible timetables in terms of delay minimization with respect to local dispatching rules. Finally, the advantage of flexibility is more significant when the timetable contains buffer time, since it allows CDR algorithms to fully make use of time reserves.

References

[1] Carlier J. (1982) The one-machine sequencing problem. *European Journal of Operational Research* **11**, 42–47.

[2] Carlier J. and Pinson E., (1994) Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research* **78**, 146–161.

[3] D'Ariano A., Pacciarelli A. and Pranzo M. (2005) A branch and bound algorithm for scheduling trains on a railway network. Technical Report DIA-97-2005. Dipartimento di Informatica e Automazione, Università Roma Tre.

[4] Dorfman M.J. and Medanic J. (2004) Scheduling trains on a railway network using a discrete event model of railway traffic. *Transportation Research, Part B* **38**, 81–98.

[5] Hemelrijk R., Kruijer J. and de Vries D.K. (2003) Schiphol tunnel 2007. Description of the situation. Internal report, Holland Railconsult, Utrecht.

[6] Middelkoop, A.D. and Hemelrijk, R. (2005) Exploring the effects of Dynamic Traffic Management. *Proceedings of Dagstuhl Seminar no. 04261 on "Algorithmic Methods for Railway Optimization"*, Schloss Dagstuhl Wadern Germany, June 2004.

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

[8] Mascis, A., Pacciarelli D. and Pranzo M. (2001) Train Scheduling in a Regional Railway Network. *Preprints of the 4th Triennial Symposium on Transportation Analysis, Sao Miguel, Portugal*, 13-19 June 2001 **3**, 487–492.

[9] J. Pachl, Railway Operation and Control. VTD Rail Publishing, Mountlake Terrace, USA, 2002.

[10] Şahin İ. (1999) Railway traffic control and train scheduling based on inter-train conflict management. *Transportation Research, Part B* **33**, 511–534.

[11] Schaafsma, A.A.M. (2001) *Dynamisch Railverkeersmanagement; besturingsconcept voor railverkeer op basis van het Lagenmodel Verkeer en Vervoer*, Delft 2001, ISBN 90-407-2219-6. In Dutch, English summary.

[12] Schaafsma, A.A.M. (2005) Dynamic traffic management - innovative solution for the Schiphol bottleneck 2007. *Proceedings of the First International Seminar on Railway Operations Modelling and Analysis* 2005, 8-10 June Delft, The Netherlands. ISBN 90-9019596-3. www.jror.nl/seminars.