Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 183 (2007) 643–657
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A branch and bound algorithm for scheduling trains in a railway network

Andrea D'Ariano [a], Dario Pacciarelli [b,*], Marco Pranzo [c]

[a] Department of Transport and Planning, Delft University of Technology, P.O. Box 5048, 2600 GA Delft, The Netherlands
[b] Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, via della vasca navale, 79 – 00146 Roma, Italy
[c] Dipartimento di Ingegneria dell'Informazione, Università di Siena, via Roma, 56 – 53100 Siena, Italy

Received 27 October 2005; accepted 5 October 2006
Available online 15 December 2006

Abstract

The paper studies a train scheduling problem faced by railway infrastructure managers during real-time traffic control. When train operations are perturbed, a new conflict-free timetable of feasible arrival and departure times needs to be re-computed, such that the deviation from the original one is minimized. The problem can be viewed as a huge job shop scheduling problem with no-store constraints. We make use of a careful estimation of time separation among trains, and model the scheduling problem with an alternative graph formulation. We develop a branch and bound algorithm which includes implication rules enabling to speed up the computation. An experimental study, based on a bottleneck area of the Dutch rail network, shows that a truncated version of the algorithm provides proven optimal or near optimal solutions within short time limits.

Keywords: Train scheduling; Real-time conflict resolution; Alternative graph; Branch and bound algorithm

1. Introduction

Railway operators plan train services in detail, defining several months in advance the train order and timing at crossings, junctions and platform tracks. A robust timetable is able to deal with minor delays occurring in real-time. However, technical failures and disturbances may influence the running times, dwelling and departing events, thus causing primary delays. Due to the interaction between trains, these delays may be propagated as secondary delays to other trains in the network. A partial modification of the timetable may then be required during operations. Hence, managing railway traffic in real-time requires re-scheduling train movements through the network, minimizing secondary delays and ensuring the feasibility of the resulting plan of operations. This process requires effective solutions within minutes and is called train dispatching, or *conflict resolution*.

To a large extent, conflict resolution issues are solved by human operators, called dispatchers, each controlling a single dispatching area. Elementary

* Corresponding author. Tel.: +39 06 55173238.
E-mail addresses: a.dariano@tudelft.nl (A. D'Ariano), pacciarelli@dia.uniroma3.it (D. Pacciarelli), pranzo@dii.unisi.it (M. Pranzo).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.10.034

Decision Support Systems have been developed to help them to quickly and effectively re-schedule train movements, but the usual policy still consists of scheduling trains following the order in the timetable or according to commonly adopted dispatching rules. For example, practical applications of local simple measures for the Dutch railway network, based on train re-ordering at crossing points on a first come first served basis, are described in [11,24]. However, we believe there is a need for more effective conflict resolution systems which are able to exploit global information about the status of the network. In this paper, we therefore address the real-time conflict resolution problem in the development of a detailed optimization model and a branch and bound procedure. The model and algorithms discussed here may also have other applications, e.g., as planning tools to validate the impact of possible timetable perturbations.

There has been increasing interest on train scheduling and timetabling problems since the pioneering paper of Szpigel [25], in which the problem of scheduling trains on a single track line was modeled as a job shop scheduling problem and solved with a branch and bound algorithm. Cordeau et al. [7] present a comprehensive survey on train scheduling and routing problems. In view of the extensiveness of their review, we provide only a brief summary here.

At least two main research lines can be identified in this context, the timetable design problem and the real-time conflict resolution problem. The former case includes a number of off-line problems at various levels of detail, and typically focuses on large rail networks and simplified models. For example, the track line between two stations is typically represented as a single resource, and it is possible to cancel potential trains from the schedule when the track capacity is insufficient. Most authors address this problem on a line with single and/or multiple track segments [25,17,15,2,20,3].

The research on real-time conflict resolution systems aims to support the dispatchers in restoring the schedules' feasibility, given the real-time positions of the trains. In order to provide physically feasible solutions with short computation times, of particular interest is a limited time horizon and a detailed representation of the area managed by a single dispatcher. Hence, the basic resource is the track line between two signals, and it is possible to cancel some train journeys by solely assigning an alternative trip to the vehicle. In the few recent papers addressing the conflict resolution problem, Higgins et al. [12] develop a branch and bound method, while Şahin [23] and Higgins et al. [13] develop heuristic algorithms for re-scheduling trains on a railway line by modifying crossing and overtaking in case of conflicts. Adenso Diaz et al. [1] described a real-time dispatching system on a regional network composed of a single corridor with three major lines. Fay [10] suggested an expert system using Petri Nets and a fuzzy rule-base for train traffic control during disturbances and performed experiments on small fictional instances. Ping et al. [22] proposed a method based on genetic algorithms for adjusting the train orders and times on a double-track line. Dorfman and Medanic [9] propose a discrete-event model for scheduling trains on a single line and a greedy strategy to obtain sub-optimal schedules. Dessouky et al. [8] develop a detailed MILP formulation and a branch and bound procedure for determining the optimal dispatching times for the trains traveling in a complex rail network.

Our work is concerned with the real time conflict resolution problem on a regional rail network. Our optimization model incorporates a detailed description of the network topology, including railway signal aspects and safety rules, on the basis of each track segment between two block signals being able to host at most one train at a time. We specifically model train trips as jobs to be scheduled on track segments, which are viewed as machines, as in Szpigel [25], Şahin [23], and Oliveira and Smith [20]. However, differing from other authors, we explicitly include no-store constraints in the optimization model, which requires that a train, having reached the end of a track segment, cannot enter the subsequent segment if the latter is occupied by another train, thus preventing other trains from entering the former segment. To achieve this, we use the alternative graph formulation of Mascis and Pacciarelli [18], which allows modeling job shop scheduling problems with no-wait and no-store constraints. We also show that the alternative graph model can easily incorporate several other railway relevant traffic regulation rules and constraints, which are rarely taken into account in the literature, as observed by Oliveira and Smith [20]. Examples of such aspects include speed restrictions, precedence and meeting constraints.

We describe a specific branch and bound procedure to solve the conflict resolution problem. It is worthwhile noting that the alternative graph formulation of a practical size instance may include hun-

dreds of machines (track segments) and tens of jobs (trains), resulting in a huge job shop problem with no-store and other constraints to be solved within the strict time limits imposed by the real-time nature of the problem. To this end, we exploit new properties of a feasible solution, which allow the design of efficient *static implication rules*. Such rules are used to speed up our branch and bound algorithm. Computational experiments, based on a heavily congested rail network, show that our branch and bound algorithm is able to provide a proven optimal or a near optimal solution within a short time.

The paper is organized as follows. In Section 2 the alternative graph formulation of the conflict resolution problem is described in detail and an illustrative example is presented. In Section 3 the static implication rules and the branch and bound algorithm for solving the problem are presented. In Section 4 our computational experience is reported.

2. Train scheduling formulation

In its basic form a railway network is composed of *track segments* and *signals*. Signals allow control of traffic on the network, and the avoidance of any potential collision between trains. There are block signals before every junction as well as along the lines and inside the stations. A *block section* is a track segment between two block signals.

Signalling systems vary quite a lot from country to country. Here, we consider the three main aspects of the Dutch signalling system NS54 that are common to most railway standards. A signal aspect may be either red, yellow or green. The red aspect means that the subsequent block section is either out of service or occupied by another train, the yellow aspect means that the subsequent block section is empty, but the one after is occupied by another train, and a green aspect means that the two subsequent block sections are empty. A train is allowed to enter the next block section at high speed only if the signal is green, it has usually to slow down to a maximum speed of 40 km/h if the signal aspect is yellow, and must stop before the end of the current section if the signal aspect is red. Thus, each block section can host at most one train at a time. A more detailed description of different aspects of railway signalling systems and traffic control regulations can be found in [21].

The passing of a train through a particular block section is called an *operation*. A block section takes a given time to be traversed, called the *running time* of the associated operation, which is known in advance for each train since all trains travel at their programmed speed whenever possible. Besides the running time of an operation, a delay may occur at the end of a block section if the signal aspect is red. The running time of the subsequent operation starts when the head (the first axle) of the train enters the new block section. However, the previous block section remains blocked by the train for a certain amount of time, referred to in this paper as *setup time*. This takes into account the time between the entrance of the head of the train in a block section and the exit of its tail (the last axle) from the previous one, plus an additional safety margin (see, e.g., [19]). After the setup time the signal before the previous section is switched to yellow and one before that to green.

A *conflict* occurs whenever two or more trains require the same block section at the same time. The real-time *conflict resolution problem* (CRP) is the on-line 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. In other words, solving CRP requires assigning *starting times* $t_1, \ldots, t_n$ to operations, i.e., defining the exact time each train can enter each block section.

Note that our definition of the conflict resolution problem is slightly different from the one most used by train dispatchers. In fact, in the practice of conflict resolution, conflicts arising in the entire network are detected and solved *one at a time*. According to Kauppi et al. [16] this lack of overview in the time/distance domain causes sub-optimization. In our definition, the aim of conflict resolution is to develop a *new conflict free plan,* such that the overall deviation from the planned schedule is minimized.

From the above discussion it follows that the combinatorial structure of CRP is similar to that of a no-store job shop scheduling problem, a block section corresponding to a no-store machine, and a train corresponding to a job. No-store (or blocking) constraints model the absence of buffers between machines. Therefore, after processing, a job cannot leave a machine until the subsequent machine becomes available. This no-store job shop scheduling problem has been successfully modeled with the alternative graph formulation of Mascis and Pacciarelli [18].

In the rest of this section, CRP is formulated by means of the alternative graph model. Each job

(train) must pass through a prescribed sequence of machines (block sections). The passing of a train through a particular block section (an operation) is a *node* of the alternative graph. The route of a train is therefore a chain of nodes to be processed in sequence. Given two consecutive nodes k and i in the sequence, with f_k denoting the running time of operation k, i can start only after at least f_k time units from the completion of k, i.e.,

$$t_i \geqslant t_k + f_k.$$

We model this constraint with a *fixed arc* (k,i) of weight f_k. Given a fixed arc (k,i), we say that i is the *successor* of k and denote it by $i = \sigma(k)$ (see Fig. 1).

Since a block section cannot host two trains at the same time, whenever two jobs require the same resource, there is a potential conflict. In this case, a processing order must be defined between the incompatible operations, and we model this by introducing a suitable pair of *alternative arcs*. So, let j and k be two conflicting operations, and let $h = \sigma(j)$ and $i = \sigma(k)$ be their respective successors. If j is scheduled before k and f_{hk} is the setup time of the block section, then k can start only after f_{hk} time units from the starting time of h, i.e.,

$$t_k \geqslant t_h + f_{hk}.$$

Similarly, if k is scheduled before j, then we have $t_j \geqslant t_i + f_{ij}$. The two constraints are alternative, and we model them with a pair of alternative arcs $((i,j),(h,k))$ (see Fig. 1). CRP therefore reduces to the problem of assigning starting times to the operations such that all fixed precedence relations, and exactly one for each pair of the alternative precedence relations, are satisfied.

Let N be the set of all the operations (nodes), F be the set of all the fixed arcs, and A be the set of all the alternative arcs in an instance. Then we call the triple $\mathcal{G} = (N,F,A)$ an alternative graph. Set N includes two dummy nodes 0 and n, "start" and "finish", respectively. Set F includes also outgoing arcs from node 0 and ingoing arcs to node n.

A *selection* S is a set of arcs obtained from A by choosing only one arc from each pair. The selection is *complete* if exactly one arc from each pair is chosen. Given a pair of alternative arcs $((i,j),(h,k)) \in A$, we say that arc (i,j) is *selected* in S if $(i,j) \in S$, whereas we say that arc (i,j) is *forbidden* in S if arc $(h,k) \in S$. Finally, the pair is *unselected* if neither (i,j) nor (h,k) is selected in S.

Given a selection S, we let $\mathcal{G}(S)$ indicate the graph $(N, F \cup S)$. A selection S is *consistent* if the graph $\mathcal{G}(S)$ has no positive length cycles. If S is consistent, the value of a longest path from i to j in $\mathcal{G}(S)$ is $l^S(i,j)$. Specifically, for each node $i \in N$, we say that the quantity $r_i = l^S(0,i)$ is the *head of node i*, and the quantity $q_i = l^S(i,n)$ is the *tail of node i*.

Given a consistent selection S, we call *extension* of S a complete consistent selection S' such that $S \subset S'$, if it exists. This notation indicates that a conflict free schedule is associated with a complete consistent selection on the corresponding alternative graph. In fact, a cycle represents an operation preceding itself, which is infeasible. Given an initial selection S^{in}, potentially empty, the objective of CRP is to find an extension S of S^{in} such that the length of a longest path from node 0 to node n in $\mathcal{G}(S)$, i.e., $l^S(0,n)$, is minimized. The selection S^{in} represents the precedence constraints implied by the initial positions of the trains and/or by order of their entrance into the network.

We next show how the longest path $l^S(0,n)$ can represent the maximum secondary delay of the associated schedule. Let k be the operation associated to the entrance of train T_α at station B_β, and i be the operation associated with its departure. Let $\omega_{\alpha\beta}$ also be the scheduled dwell time of T_α at B_β (see Fig. 2). The constraint on the minimum departure time $\delta_{\alpha\beta}$ of T_α from B_β, imposed by the timetable, $t_i \geqslant \delta_{\alpha\beta}$, is obtained by adding arc $(0,i)$ with weight $\delta_{\alpha\beta}$. In order to compute the maximum secondary delay, we first have to evaluate the unavoidable primary delay of T_α at B_β, as follows. Let $S = \emptyset$ represent the empty selection, i.e, the selection associated to

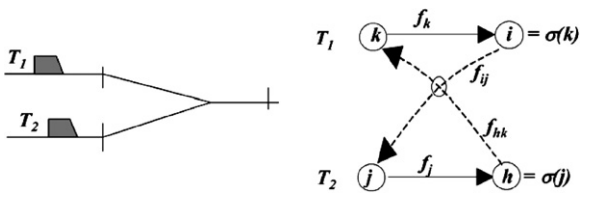

Fig. 1. The alternative graph formulation for two trains at a junction.

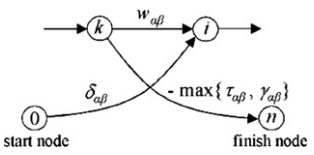

Fig. 2. The alternative graph formulation of a scheduled stop.

the graph $\mathcal{G}(\emptyset)$ composed only of fixed arcs, and let $\tau_{\alpha\beta} = l^{\emptyset}(0,k)$ be the earliest possible arrival time of T_α at B_β while respecting all the constraints on train input delay, maximum speed, acceleration, and minimum departure time. In this phase, possible conflicts with other trains are not considered, i.e., the value $\tau_{\alpha\beta}$ is a lower bound on the arrival time of T_α at B_β in a feasible solution. Let $\gamma_{\alpha\beta}$ be the scheduled arrival time of T_α at B_β in the timetable. If $\tau_{\alpha\beta} > \gamma_{\alpha\beta}$, then the quantity $\tau_{\alpha\beta} - \gamma_{\alpha\beta}$ is the (unavoidable) *primary delay* that cannot be recovered by rescheduling train operations. Hence, the *secondary delay* of T_α at B_β in a feasible schedule can be computed as $t_k - \max\{\tau_{\alpha\beta},\gamma_{\alpha\beta}\}$. We call the quantity $\max\{\tau_{\alpha\beta},\gamma_{\alpha\beta}\}$ the *modified due date* of T_α at B_β. By adding a fixed arc (k,n) from k to the finish arc n, having weight $-\max\{\tau_{\alpha\beta},\gamma_{\alpha\beta}\}$, the length of the path from 0 to n passing through (k,n) is equal to the secondary delay of T_α at B_β (see Fig. 2). Adding one such arc to the alternative graph for each train at each scheduled stop, as well as at its exit point from the network, $l^S(0,n)$ is proved to be equal to the maximum secondary delay.

Besides the cited examples, the alternative graph allows formulation of situations more general that the ones we have mentioned. For example, Oliveira and Smith [20] point out the relevance of precedence and meeting constraints in the management of railway operations. A *precedence constraint* between two trains at a station may be necessary if one train is carrying the crew or a vehicle necessary for another train. In such cases the latter train must wait for the arrival of the former one before departing. This constraint can be modeled by simply adding one fixed arc of suitable weight from the node associated to the arrival of the former train to the node associated to the departure of the latter one. A *meeting constraint* allows specification of a minimum dwell time during which two given trains must be together at a given station for exchanging passengers or goods. This can be modeled by adding two fixed arcs of suitable weight: an arc from the node associated with the arrival of the former train to the node associated with the departure of the latter one, another arc from the node associated with the arrival of the latter train to the node associated to the departure of the former one.

As fast trains require green signal aspect in order to travel at their planned speed, it is important to schedule their movements by ensuring that two empty block sections are available to them. We next describe how to model this situation for a small conflict resolution problem with two trains traveling at different speed. Fig. 3 shows a railway network with four block sections (denoted as 1, 2, 3 and 9), a simple station with two platforms (6 and 7), and three junctions (4, 5 and 8). All these resources have a capacity of one. At time $t = 0$ there are two trains in the network. Train T_A is a slow train running from block section 3 to block section 9, and stopping at platform 6. Train T_A can therefore enter a block section only if the signal aspect is yellow or green. Train T_B is a fast train running from block section 1 to block section 9 through platform 7 without stopping. Train T_B can therefore enter a block section at high speed only if the signal aspect is green.

In Fig. 4, the alternative graph for this example is reported. For the sake of clarity, each node of the alternative graph is indicated with a pair train-block section, e.g., A4 indicates the train T_A traversing block section 4. Trains T_A and T_B share resources 4, 5, 8 and 9. There are therefore four pairs of alternative arcs, each associated with a pair of possibly conflicting operations. A pair of alternative arcs is represented in Fig. 4 by connecting the two paired arcs with a small circle. The weight on fixed and alternative arcs is not depicted. The weights δ_{Ain} and δ_{Bin} on arcs $(0,\mathrm{A4})$ and $(0,\mathrm{B2})$, respectively,

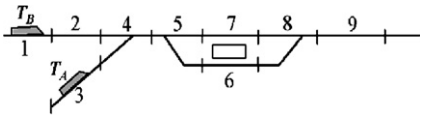

Fig. 3. A small rail network.

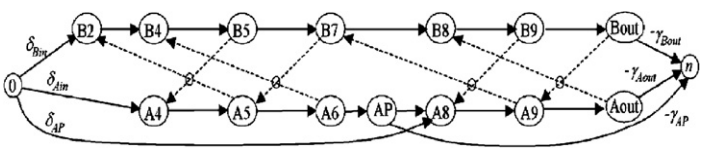

Fig. 4. The alternative graph for the example with two trains.

represent the time at which the two trains are expected to reach the end of their current block sections. The weight δ_{AP} on arc $(0, A8)$ is the scheduled departure time of train T_A from the platform P, while the weight on arc $(AP, A8)$ (not depicted in Fig. 4) is its scheduled dwell time. Finally, the weights $-\gamma_{AP}$, $-\gamma_{Aout}$ and $-\gamma_{Bout}$ are the modified due dates of trains T_A and T_B at the scheduled stop and at the exit point of the network.

3. Conflict resolution algorithms

In this section, we describe some properties of CRP which can be used to speed up solution algorithms and then list the main components of our branch and bound procedure.

A key concept in the reduction of the computational effort of branch and bound procedures for the job shop problem is the concept of immediate selection, or *dynamic implication*. The idea is to prove that, given a partial selection S and an unselected pair of alternative arcs $((i,j),(h,k)) \in A$, no improvement to the current best solution is possible if arc (h,k) is selected. In such case, we say that arc (i,j) is *implied* by S, arc (h,k) can be forbidden and arc (i,j) selected, i.e. added to S.

Dynamic implication rules have been used to solve both the case with infinite capacity buffers [5,6] as well as that with no-store constraints [18]. However, the computation of dynamic implications may require excessive time when dealing with a large number of jobs and machines, as in practical size instances of CRP. In our branch and bound algorithm, we therefore include only the following two dynamic rules, both of which can be computed very efficiently. See [6] for the definition of *ascendant* and *descendant sets*.

Proposition 3.1. *Given a selection S, if $((i,j),(h,k))$ is an unselected pair of alternative arcs and $r_h + f_{hk} + q_k \geqslant UB$, then arc (h,k) is forbidden, and arc (i,j) is implied by S.*

Proposition 3.2. *If J is an ascendant set of c, all arcs $(\sigma(c), j)$, $\forall j \in J$, are forbidden. If J is a descendant set of c, all arcs $(\sigma(J), c)$, $\forall j \in J$, are forbidden.*

Carlier and Pinson [6] describe an algorithm for finding a maximal set of implied arcs, of complexity $O(|K|\log|K|)$, where K is the set of operations to be processed on a given machine. Their algorithm is based on a single machine scheduling problem with heads, tails and preemption, and computes the so called *Jackson preemptive schedule* (JPS) [14], which also returns a lower bound to the length of the longest path from 0 to n for any extension of S [4]. It can also be used in the alternative graph context, and has been successfully used in [18] for the solution of the job shop scheduling problem with no-store and no-wait constraints.

In this paper, we develop *static implication* rules for CRP which can be computed off-line on the basis of the physical track topology of the rail network. Static implications are particularly effective for CRP, and allow a significant reduction of the computation time of our solution algorithms. The following proposition allows the establishment of correspondence between the selection of arcs from different alternative pairs.

Proposition 3.3. *Consider a selection S and two unselected alternative pairs $((a,b),(c,d))$ and $((i,j),(h,k))$. If $f_{ab} + l^S(b,i) + f_{ij} + l^S(j,a) \geqslant 0$, then arc (h,k) is implied by selection $S \cup \{(a,b)\}$ and arc (c,d) is implied by selection $S \cup \{(i,j)\}$.*

Proof. The result immediately follows from the observation that the selection $S \cup \{(a,b),(i,j)\}$ contains a positive length cycle. □

Proposition 3.3 can be applied particularly with the empty selection $S = \emptyset$, when the graph $\mathcal{G}(\emptyset)$ is composed only of fixed arcs associated with the train paths. Therefore, the resulting static implication rules can be computed in pre-processing step, on the basis of the topology of the rail network and given the train patterns. A list of implied alternative arcs is associated with each alternative arc. During the execution of the solution procedure, the selection of an arc causes the selection of the entire set of arcs implied by it. The following proposition expresses Proposition 3.3 in terms of the physical topology of the rail network.

Proposition 3.4. *Consider two alternative pairs $((a,b),(c,d))$ and $((i,j),(h,k))$. Then, $f_{ab} + l^{\emptyset}(b,i) + f_{ij} + l^{\emptyset}(j,a) \geqslant 0$ if the following conditions hold.*

1. Nodes b and i are associated with train T_1 and are connected by a directed path of fixed arcs, i.e., T_1 executes b before i.
2. Nodes j and a are associated with train T_2 and are connected by a directed path of fixed arcs, i.e., T_2 executes j before a.
3. T_1 and T_2 pass through the two block sections $((a,b),(c,d))$ and $((i,j),(h,k))$ refer to.

Proof. The result immediately follows from Proposition 3.3. □

We next specify the two static implication rules used in our algorithms. Let us use B_1 and B_2 to denote the block sections associated with pairs $((a,b),(c,d))$ and $((i,j),(h,k))$, respectively. The conditions of Proposition 3.4 hold in the following two cases:

- B_1 and B_2 are adjacent block sections, traversed by T_1 and T_2 in the same order. In this case, nodes a and j coincide, as well as c and k (see Fig. 5). Here, arcs (a,b) and (h,k) imply each other, as well as arcs (c,d) and (i,j).
- T_1 and T_2 pass both through B_1 and B_2 in opposite orders (see Fig. 6). In this case, arc (a,b) implies arc (h,k) and arc (i,j) implies arc (c,d). Note that arc (h,k) does not imply arc (a,b), and arc (c,d) does not imply arc (i,j).

Next, we describe the main components of our branch and bound algorithm. A number of variants, particularly effective for CRP, allow a drastic decrease in computation time when using the algorithm of Mascis and Pacciarelli [18].

We compute static implications in a preprocessing phase of the algorithm, and associate to each alternative arc (i,j) the set of all arcs statically implied. We call this set Stat(i,j). Static implications are then utilized during the execution of initial algorithms, as well as in the branch and bound procedure.

Given an initial selection S^{in}, possibly empty, the procedure builds an extension of S^{in}, minimizing the length of the longest path from 0 to n. The procedure maintains a structure List, containing a set of partial selections initially equal to S^{in}. A partial selection S is added to List if an extension of S improving the current optimum UB exists. Proposition 3.4 is used in the preprocessing phase to compute static implications Stat(u,v) for each unselected alternative arc $(u,v) \in A$.

At each step the algorithm chooses a selection S from List, an unselected alternative pair $((i,j),(h,k))$ from S and builds the two selections $S \cup \{(i,j) \cup \text{Stat}(i,j)\}$ and $S \cup \{(h,k) \cup \text{Stat}(h,k)\}$. Propositions 3.1 and 3.2 are then exploited to compute dynamic implications and to enlarge the two selections. This step is repeated until it is impossible to enlarge them anymore. The Jackson preemptive schedule, computed for each resource of the network, then gives a lower bound on any extension of each of the two selections. Finally, each selection is added to List if the lower bound is smaller than the current optimum UB.

In this paper, we intend to evaluate the potential benefit of effective procedures for solving CRP. To

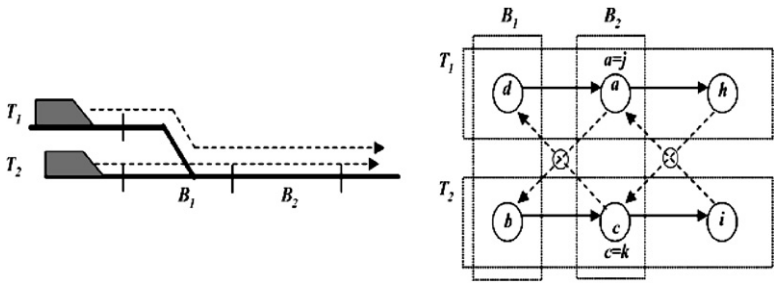

Fig. 5. Trains traveling in the same direction.

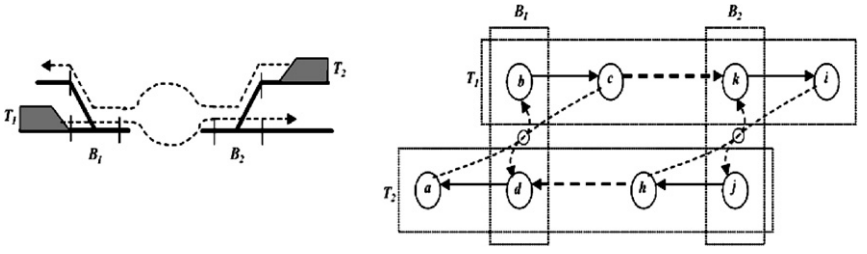

Fig. 6. Trains traveling in opposite directions.

this end, we implement two local simple dispatching rules, to simulate the typical behavior of a human dispatcher, and a greedy heuristic based on global information. The proposed algorithms are then tested as initial solutions of the branch and bound procedure.

The first dispatching rule is the First Come First Served (FCFS), which is commonly adopted in railway practice. It consists simply of giving precedence to the train arriving first at a block section. Starting from the initial selection S^{in}, FCFS computes the arrival time of the trains at each junction, sequencing the trains in order of arrival.

The second dispatching rule, called First Leave First Served (FLFS), is as follows. When two trains require the same block section, the time required for each train to enter and traverse it is first computed. Precedence then is given to the train which is able to leave the block section first. FLFS is a compromise between two commonly used dispatching rules of (i) giving priority to the fast trains over the slow ones and (ii) giving precedence to the train arriving first.

The third heuristic (see Fig. 7) is similar to the greedy algorithm AMCC described in [18]. It consists of forbidding one alternative arc at a time, the one which would introduce the largest delay. AMCC enlarges at each step a selection S, initially set equal to S^{in}, by choosing a pair of unselected alternative arcs $((i,j),(h,k))$ such that the quantity $l^S(0,h)+f_{hk}+l^S(k,n)$ is maximum among all unselected alternative arcs. Hence, selecting arc (h,k) would increase $l^S(0,n)$ more than any other unselected alternative arc. For this reason, AMCC forbids this arc and selects its alternative, i.e., arc (i,j) is added to S, together with the arcs in the set Stat(i,j). The difference with the AMCC presented in [18] is that here we consider only static implications and not dynamic ones. This results in a significant reduction of the computation time of the heuristic.

In our computational experience, the solutions found by AMCC are almost always better than those of the two dispatching rules. On the other hand, while the computation time of the two dispatching rules is negligible, AMCC is significantly slower and fails to find feasible solutions more frequently. Hence, none of the three heuristics outperforms the others overall. For these reasons, the initial value of the upper bound UB in the branch and bound code is set equal to the best value obtained by all the three heuristics.

We also evaluated several branching schemes and search strategies of the enumeration tree. The best configuration for CRP, in terms of computation times and number of branches, adopts the following binary branching scheme based on the AMCC rule:

Algorithm AMCC

begin
$S = S^{in}$.
while $A \neq \emptyset$ **do**
  begin
  Let $C_{\max}(S) = \max_{(u,v)\in A}\{l^S(0,u)+f_{uv}+l^S(u,n)\}$,
  Choose a pair $((i,j),(h,k)) \in A$ such that $l^S(0,h)+f_{hk}+l^S(k,n) = C_{\max}(S)$,
  Let $S' := S \cup \{(i,j)\}$, $A' := A - \{((i,j),(h,k))\}$,
  Select all arcs in the set $Stat(i,j)$,
  if (there is a cycle in $\mathcal{G}(S')$) **or** (an arc from $Stat(i,j)$ is forbidden) **then**
    begin
    Let $S' := S \cup \{(h,k)\}$, $A' := A - \{((i,j),(h,k))\}$,
    Select all arcs in the set $Stat(h,k)$,
    if (there is a cycle in $\mathcal{G}(S')$) **or** (an arc from $Stat(h,k)$ is forbidden)
    then STOP, the procedure failed in finding a feasible solution,
    end
  $S := S', A := A'$.
  end
end.

Fig. 7. Sketch of the AMCC heuristic.

Branch on the unselected pair $((i,j),(h,k))$ such that $l^S(0,i)+f_{ij}+l^S(j,n)$ is maximum. As a search strategy when exploring the enumeration tree, the best configuration consists of extracting a partial selection S from *List* as follows. It alternates three repetitions of the depth-first visit to a choice of the selection S with smallest lower bound among the last nine sets S inserted in List.

4. Computational experience

In this section, we report on our computational experiments on a large sample of practical size instances, and compare local and global conflict resolution strategies. Our experiments are based on the Schiphol dispatching area (Fig. 8), a bottleneck area of the Dutch railway network including the underground station of Schiphol, beneath the international airport of Amsterdam. The algorithms for conflict resolution are implemented in C++ language and executed on a laptop equipped with a 1.6 GHz Pentium M processor.

4.1. Test structure

The Schiphol rail network is around 20 km long and consists mainly of four tracks. It is composed of 86 block sections and includes 16 platforms and two traffic directions. Trains enter/leave the network from/to ten access points: two high speed line, HSL, (block sections 20 and 59), the station of Nieuw Vennep (Nvp) (block sections 33 and 70), the yard of Hoofddorp station (HfdY) (block sections 18 and 71), and two stations in Amsterdam, namely Amsterdam Lelylaan (Asdl) (block sections 46 and 50) and Amsterdam Zuid WTC (Asdz) (block sections 77 and 45). The Hoofddorp station (Hfd) has two platforms, which are dedicated to freight trains, while the Schiphol station (Shl) has six platforms, which are dedicated to passenger and freight trains. The two traffic directions are largely independent except for the underlined block sections 38, 40, 42, 44, 46, 47, 48, 49 and 50 around Amsterdam Lelylaan station and the block Section 4 at the border of Hoofddorp yard.

We study the network by simulating real-time traffic conditions for different kinds of initial perturbations, and computing a re-scheduling of train operations each time. We consider a provisional timetable for 2007 [11], which forecasts an increase in the traffic through this bottleneck area, thus making it an interesting test case for our study. The timetable is hourly cyclic and contains 54 trains circulating each hour. The time horizon of practical interest for railway managers is usually less than one hour for real-time purposes.

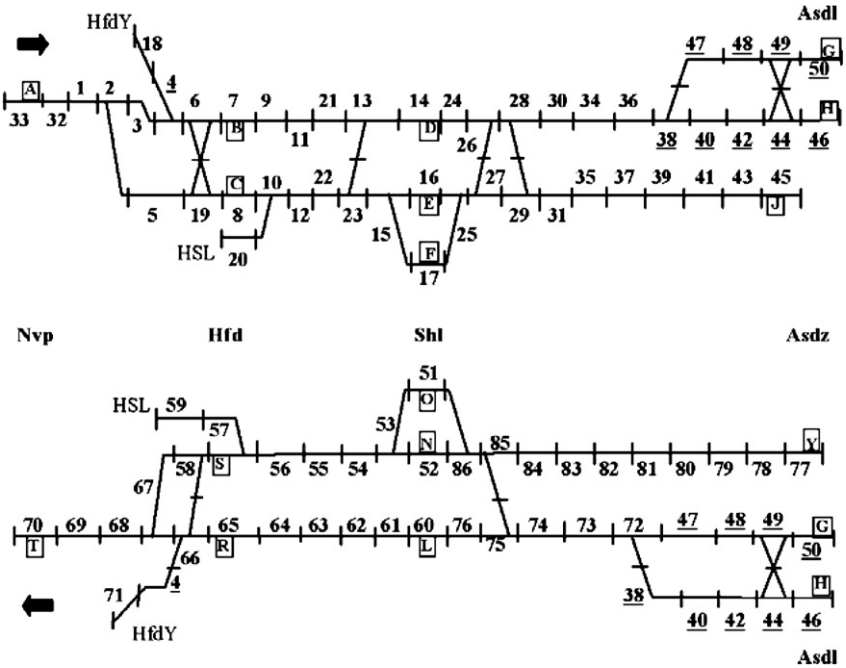

Fig. 8. The Schiphol railway network.

Given a timetable of one hour, we consider 60 perturbation schemes, obtained by varying 60 different sets of delayed trains. A random input delay is assigned to some trains, i.e., the trains enter the network late with respect to the timetable. More precisely, the 60 perturbation schemes are divided in six groups of ten instances each. The number of delayed trains in each group is the same, varying between 7 and 27. The 10 instances differ for the sets of delayed trains and for the delay values. The delayed trains are randomly chosen within the first half hour of the cyclic timetable. In five out of ten instances, the random value of the input delay is generated on the basis of Gaussian distribution in the range $[0, \max]$. In the remaining five instances it is chosen according to the uniform distribution in the same range.

Starting from the first set of 60 real-size instances, we generate other 240 more difficult instances to better evaluate the performance of the branch and bound algorithm. For each perturbation scheme, we introduce four variations of the timetable by simply anticipating the scheduled departure times of all trains from each station of the same amount, chosen in the set $\{30, 60, 90, 120\}$ seconds, and keeping constant the arrival times. Such timetable modifications enlarge the set of feasible solutions to the scheduling problem, since the constraints on minimum departure time are relaxed, thus making the instances potentially more difficult to solve for the branch and bound algorithm.

We denote each instance with a four field code *A–B–C–D* as follows: Under *A* we denote the number of delayed trains; *B* indicates the distribution type, uniform (U) or Gaussian (G); *C* represents the range $[0, \max]$ in which we choose the delay values, for max varying in the set $\{200, 300, 400, 600, 800, 1000, 1400, 1800\}$ seconds; *D* indicates the type of timetable, varying from 1 to 5, 1 being the original timetable.

In the first two sets of instances the number of trains is 54, corresponding to alternative graphs with more than 8000 pairs of alternative arcs. In order to further evaluate the limits of the branch and bound, we generated a third set of 300 instances by enlarging the timetable up to two hours for every instance of the first two sets, corresponding to 108 trains and graphs with more than 30 000 pairs of alternative arcs.

For all the 600 test cases, we compute the performance of the three initial heuristics and the branch and bound algorithm as described in Section 3 and truncated after 120 seconds of computation. This amount of computation time makes the algorithm compatible with the needs of real-time applications.

Table 1
Effects of initial heuristics

AMCC	Time horizon (hours)	Iterations		Time (seconds)		# Optimal solutions
		Best	Total	Best	Total	
No	1	149.45	387.49	2.58	4.39	297/300
Yes	1	9.49	116.63	0.69	1.93	297/300
No	2	358.58	794.34	26.74	36.70	294/300
Yes	2	31.77	92.20	17.17	20.13	294/300

In a first test phase, we analyze the time performance of the initial heuristics in the branch and bound procedure. FCFS and FLFS dispatching rules require a negligible computation time, whereas AMCC needs on average 0.5 and 14.6 seconds on one hour and two hour instances, respectively. In order to analyze the actual benefit of AMCC, we run the branch and bound algorithm with and without the initial AMCC heuristic.

Table 1 describes the influence of AMCC algorithm in detail. Each row of the table gives the average behavior for over 300 instances. The average number of iterations required to obtain the best solution and the total average number of iterations are shown in columns three and four respectively. The two subsequent columns show the average computation time required to reach the best solution and to prove the optimality. Finally, the last column reports on the number of times the branch and bound algorithm finds a proven optimal solution within the time limit. As reported in Table 1, the use of the AMCC almost halves the computation time for both sets of one hour and two hour instances.

4.2. Evaluation of the experiments

In this subsection, we evaluate the first two sets of experiments with one hour timetables. For each of the 300 instances, we compute the performance of the three initial heuristics and the branch and bound algorithm described in Section 3, truncated after 120 seconds of computation. In Fig. 9, we present the solutions obtained in terms of maximum and average secondary delays. We do not consider primary delays since they depend only on the input delay and cannot be avoided. For each of the four

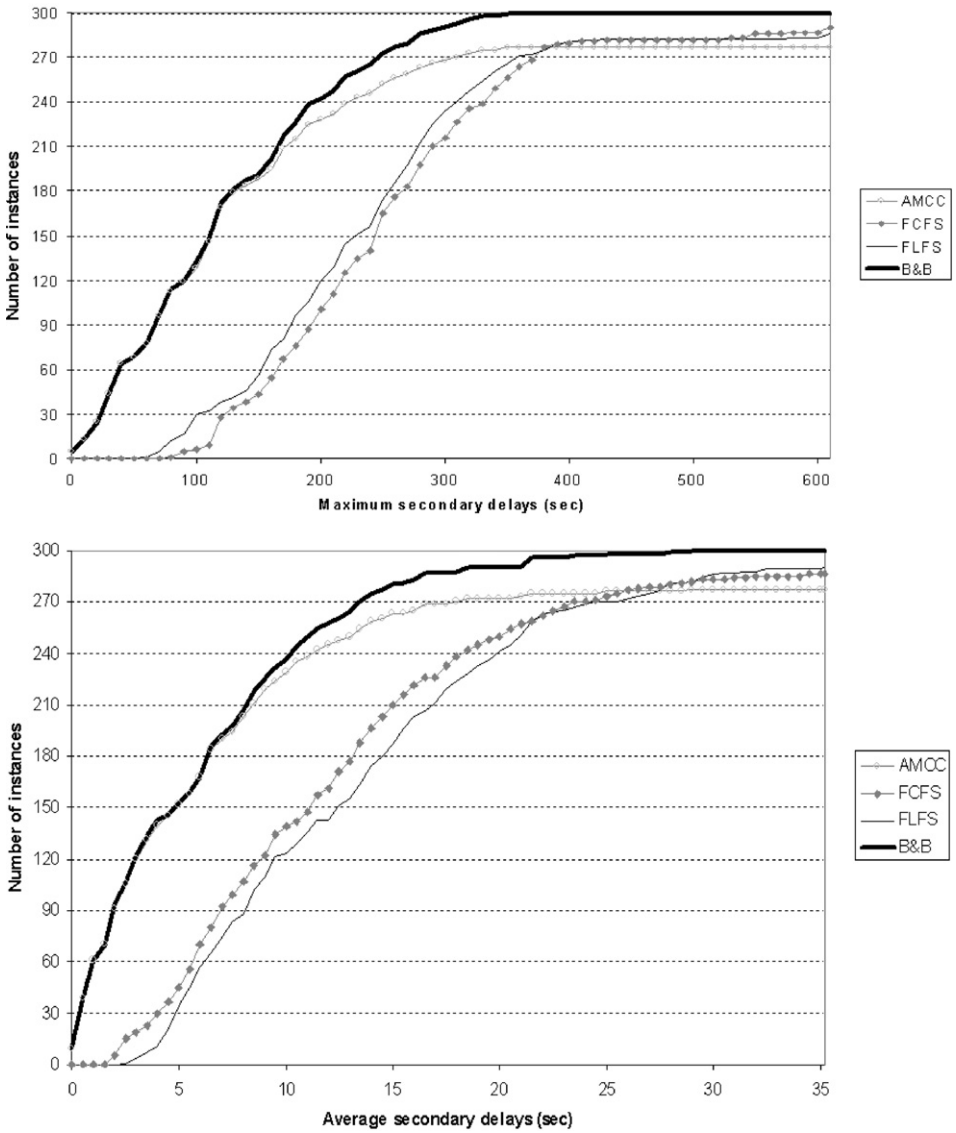

Fig. 9. Cumulative plots of maximum and average secondary delays.

considered algorithms, a point (x, y) in the curve corresponds to the number of instances y with a secondary delay smaller or equal to x over the 300 instances.

As expected, the branch and bound algorithm clearly outperforms the other algorithms. The improvement with respect to commonly used dispatching rules, such as FCFS and FLFS, confirms the advantage of global optimization even within the strict time limits imposed by real-time purposes. The branch and bound algorithm reduces the amount of maximum secondary delay by approximately 100 seconds with respect to the dispatching rules FCFS and FLFS. Unexpectedly, the branch and bound also exhibits better behavior when considering the average delays. The total amount of average secondary delay is approximately 50% less with respect to FCFS and FLFS.

The branch and bound algorithm finds proven optimal solutions in 297 cases out of 300 within the time limit of 120 seconds. The average number of nodes in the enumeration tree is 116, and the average computation time is 1.93 seconds. In the three open instances, the branch and bound algorithm is able to improve the solutions found by the three initial heuristics. The distance from the

lower bound is over 85% for the heuristics, while it drops to 26% when using the branch and bound algorithm.

The First Come First Served (FCFS) and the First Leave First Served (FLFS) rules, being the most simple local rules, give the worst results. While the worst solution found by the branch and bound algorithm exhibits a maximum secondary delay of 350 seconds, the worst solutions calculated by the dispatching rules have more than 600 seconds of secondary delay. Thus, simple dispatching rules may, in general, lead to poor results. Moreover, since the Schiphol network is not a deadlock free network, the heuristics may fail to find a feasible solution. In particular, FCFS finds a feasible solution 290 times out of 300, FLFS 286 times out of 300, AMCC 277 times out of 300 instances. However, when AMCC finds a feasible solution, it is almost always the optimal one.

4.3. Effects of static implications

Here, we show the effectiveness of static implications for CRP, both in the initial heuristics as well as in the branch and bound algorithm. The experiments are carried out on the first set of 60 instances with 54 trains and the original Schiphol timetable, and on other 60 instances with 108 trains, obtained by replicating the timetable in the second hour.

Table 2 describes the performance of our algorithms when including static implications both in the initial heuristics and in the branch and bound, compared to the case in which static implications are not used either in the initial heuristics or in the branch and bound. Both algorithms are truncated after 120 seconds. Every row of the table gives the average behavior over 60 instances. The last column reports on the number of times the algorithm finds a feasible solution.

The advantage of using static implications in the solution of CRP is shown clearly in Table 2. When using static implications, the algorithm is always able to attain and prove the optimality, and only requires a few steps to terminate a run. The main differences between one hour and two hour instances are in the increase of the number of iterations and CPU time.

When static implications are not used, the algorithm is unable to reach the optimal solution in more than 50% of the instances and, in some cases, is not even able to find a feasible solution. In fact, since the three heuristics do not use static implications, the number of times our heuristics find a feasible solution decreases significantly. When a time horizon of two hours is considered, the algorithm is unable to find a feasible solution in about 40% of instances and is only able to prove optimality in a few cases. Finally, we observe that the time limit of 120 seconds only allows exploration of a limited number of nodes when tackling two hour instances. This is partially due to the longer time required by the three heuristics to compute an initial solution without static implications.

4.4. "Hard" instances

In this subsection, we describe the most difficult instances over the three sets of experiments with timetables of one and two hours. These experiments show that CRP is a huge no-store job shop problem, but practical instances are, in most cases, easy to solve. In the vast majority of the instances, the branch and bound closes the problem at root or after a single branching of the enumeration tree and the optimal solution almost always coincides with the lower bound. Moreover, AMCC is very effective at finding good solutions. However, there are some difficult instances. In this subsection, we study the performance of the branch and bound algorithm on hard instances. We classify a solution *hard* if the branch and bound requires more than two nodes of the enumeration tree for finding the proven optimum. Specifically, there are only two hard instances from the first set of 60 instances,

Table 2
Effects of static implications

Static implication	Time horizon (hours)	Iterations		Time (seconds)		# Optimal solutions	# Feasible solutions
		Best	Total	Best	Total		
Yes	1	7.3	15.1	0.66	0.74	60/60	60/60
Yes	2	17.2	34.6	16.15	18.11	60/60	60/60
No	1	112.3	7316.9	25.45	67.76	28/60	50/60
No	2	0.0	21.0	117.68	119.40	8/60	38/60

and in both cases the optimal solution equals the lower bound. For this first set of test cases, the optimal solution is proved by our branch and bound algorithm in less than 5 seconds. There are other 12 hard instances from the second set of 240 instances and 24 hard instances in the third set of 300 instances with a timetable of two hours. In this last set of experiments, we restrict our attention to 38 hard instances and enlarge the time limit of the branch and bound algorithm to 7200 seconds.

The results of the hard instances are reported in Tables 3 and 4. In the first column, the code of the instance is given with the *A–B–C–D* fields code

Table 3
Hard instances with a time horizon of one hour

Perturbation ID	Lower bound	Initial solution	Final solution	Iterations		Time (seconds)	
				Best	Total	Best	Total
7-G-800-3	125	165	125*	146	146	3.1	3.1
7-G-800-4	95	145	95*	144	144	3.0	3.0
7-G-800-5	65	145	65*	151	151	3.1	3.1
26-G-600-1	48	310	162*	199	387	3.9	4.8
11-U-1000-1	212	320	212*	239	239	4.6	4.6
11-U-1000-2	182	290	182*	232	232	4.4	4.4
11-U-1000-3	152	260	175*	229	58498	4.4	656.4
11-U-1000-4	122	248	155	236	494563	4.4	7200.0
11-U-1000-5	114	207	155*	207	46624	4.0	575.4
11-G-1000-5	151	215	151*	218	218	4.3	4.3
16-U-1800-2	237	343	237*	248	248	4.6	4.6
16-U-1800-3	207	262	207*	189	189	3.6	3.6
16-U-1800-4	177	262	177*	203	203	3.9	3.9
16-U-1800-5	177	262	177*	207	207	4.0	4.0

Table 4
Hard instances with a time horizon of two hours

Perturbation ID	Lower bound	Initial solution	Final solution	Iterations		Time (seconds)	
				Best	Total	Best	Total
7-G-800-2	155	197	155*	361	361	39.9	39.9
7-G-800-3	125	236	125*	439	439	45.3	45.3
7-G-800-4	95	212	95*	430	430	45.0	45.0
7-G-800-5	65	212	70*	446	873	46.0	54.8
26-G-600-1	151	310	162*	494	961	50.5	60.4
26-G-600-2	121	280	196	465	144578	49.3	7200.0
26-G-600-3	91	185	185	0	156045	14.8	7200.0
26-G-600-4	61	155	155*	0	31	14.9	16.6
11-U-1000-1	212	320	212*	540	540	57.0	57.0
11-U-1000-2	182	290	182*	521	521	55.8	55.8
11-U-1000-3	152	260	175	497	146215	53.5	7200.0
11-U-1000-4	122	248	155	543	147400	57.0	7200.0
11-U-1000-5	114	207	155	472	145888	51.7	7200.0
11-U-1400-3	70	124	100*	287	566	33.9	41.5
11-U-1400-4	40	110	100*	278	546	33.3	41.4
11-U-1400-5	40	110	100*	287	564	33.9	42.1
11-U-1800-5	183	343	343	0	187610	16.0	7200.0
11-G-1000-3	211	236	211*	467	467	50.3	50.3
11-G-1000-4	181	215	181*	463	463	49.5	49.5
11-G-1000-5	151	215	151*	476	476	50.4	50.4
16-U-1800-2	237	343	237*	591	591	58.2	58.2
16-U-1800-3	207	262	207*	471	471	50.5	50.5
16-U-1800-4	177	262	177*	496	496	50.5	50.5
16-U-1800-5	177	262	177*	507	507	51.4	51.4

of Section 4.1. Columns two, three and four report on the lower bound, the best solution found by AMCC, FCFS, FLFS and the final solution found by the branch and bound respectively. The asterisk on the elements of column four indicates the proven optimality of the solution. Columns five and six (seven and eight) show the number of nodes in the enumeration tree (the computation time in seconds) which are needed to reach the best solution and the termination of the algorithm, respectively.

The branch and bound is able to close 11 out of 14 hard instances of the one hour time horizon, and the optimal solution is always found within 5 seconds. For two of the three remaining hard instances the optimal solution is proven within eleven minutes. One open instance remains, for which the algorithm is unable to improve the solution found in the first 4.4 seconds. A similar behavior can be observed for the two hour instances. In Table 4, six instances remain unsolved within two hours of computation time, but the best solution is always found in less than one minute. Moreover, the proven optimum equals the initial lower bound in 50% of instances, thus requiring less than one minute of total computation time.

5. Conclusions

This paper describes a new optimization algorithm for the real-time management of a complex rail network. We make use of a detailed discrete-event optimization model able to efficiently treat the no-store aspect of the train scheduling problem. We present a truncated branch and bound algorithm for finding a conflict-free schedule, given an initial perturbation that makes infeasible the timetable. As the objective function, we minimize the maximum secondary delay for all trains at all visited stations.

The computational experiments, carried out on a heavily congested area of the Dutch railway network, show very promising performance of the algorithm in finding near optimal solutions to practical-size instances within short computation times, compatible with real-time purposes. More importantly, from a practical point of view, the computational experience demonstrates that algorithms based on global information clearly outperform the commonly used dispatching rules, with respect to both maximum and average delays. Hence, such algorithms may be able to yield significant improvements in the quality of railway service.

Acknowledgements

We thank ProRail (The Netherlands) for providing the instances and the two anonymous referees for their helpful comments. This work is partially supported by the programs "Towards Reliable Mobility" of the Transport Research Centre Delft, the Dutch foundation "Next Generation Infrastructures" and by the European Commission, Grant number IST-2001-34705, project "COMBINE2- enhanced COntrol center for fixed and Moving Block sIgNalling systEm 2".

References

[1] B. Adenso-Díaz, M. Oliva González, P. González-Torre, On-line timetable rescheduling in regional train services, Transportation Research Part B 33 (1999) 387–398.
[2] U. Brännlund, P.O. Lindberg, A. Nöu, J.E. Nilsson, Railway timetabling using lagrangian relaxation, Transportation Science 32 (1998) 358–369.
[3] A. Caprara, M. Fischetti, P. Toth, Modeling and solving the train timetabling problem, Operations Research 50 (2002) 851–861.
[4] J. Carlier, The one-machine sequencing problem, European Journal of Operational Research 11 (1982) 42–47.
[5] J. Carlier, E. Pinson, An algorithm for solving the job-shop problem, Management Science 35 (2) (1989) 164–176.
[6] J. Carlier, E. Pinson, Adjustment of heads and tails for the job-shop problem, European Journal of Operational Research 78 (1994) 146–161.
[7] J.F. Cordeau, P. Toth, D. Vigo, A survey of optimization models for train routing and scheduling, Transportation Science 32 (4) (1998) 380–420.
[8] M.M. Dessouky, Q. Lu, J. Zhao, R.C. Leachman, An exact solution procedure to determine the optimal dispatching times for complex rail networks, IIE Transaction 38 (2) (2006) 141–152.
[9] M.J. Dorfman, J. Medanic, Scheduling trains on a railway network using a discrete event model of railway traffic, Transportation Research Part B 38 (2004) 81–98.
[10] A. Fay, A fuzzy knowledge-based system for railway traffic control, Engineering Application of Artificial Intelligence 13 (2000) 719–729.
[11] R. Hemelrijk, J. Kruijer, D.K. de Vries, Schiphol tunnel 2007. Description of the situation, Internal report, Holland Railconsult, Utrecht, The Netherlands, 2003.
[12] A. Higgins, E. Kozan, L. Ferreira, Optimal scheduling of trains on a single line track, Transportation Research Part B 30 (1996) 147–161.
[13] A. Higgins, E. Kozan, Heuristic techniques for single line train scheduling, Journal of Heuristics 3 (1997) 43–62.
[14] J.R. Jackson, Scheduling a production line to minimize maximum tardiness, Research Report 43, Management Science Research Project, University of California, Los Angeles, USA, 1955.
[15] D. Jovanovic, P.T. Harker, Tactical scheduling of train operations: The SCAN I system, Transportation Science 25 (1991) 46–64.

[16] A. Kauppi, J. Wikstrm, B. Sandblad, A.W. Andersson, Future train traffic control: Control by re-planning, Cognition Technology and Work 8 (1) (2006) 50–56.
[17] E. Kraft, A branch and bound procedure for optimal train dispatching, Journal of the Transportation Research Forum 28 (3) (1987) 263–276.
[18] A. Mascis, D. Pacciarelli, Job shop scheduling with blocking and no-wait constraints, European Journal of Operational Research 143 (3) (2002) 498–517.
[19] L. Nie, I.A. Hansen, System analysis of train operations and track occupancy at railway stations, European Journal of Transport and Infrastructure Research 5 (1) (2005) 31–54.
[20] E. Oliveira, B.M. Smith, A Job-Shop Scheduling Model for the Single-Track Railway Scheduling Problem, School of Computing Research Report 2000.21, University of Leeds, England, 2000.
[21] J. Pachl, Railway Operation and Control Mountlake Terrace, VTD Rail Publishing, 2002.
[22] L. Ping, N. Axin, J. Limin, W. Fuzhang, Study on intelligent train dispatching, in: Proceedings of IEEE Intelligent Transportation Systems Conference, Oakland, USA, 25–29 August 2001.
[23] İ. Şahin, Railway traffic control and train scheduling based on inter-train conflict management, Transportation Research Part B 33 (1999) 511–534.
[24] A.A.M. Schaafsma, Dynamic traffic management – innovative solution for the Schiphol bottleneck 2007, in: Proceedings of the First International Seminar on Railway Operations Modelling and Analysis, Delft, The Netherlands, 8–10 June 2005.
[25] B. Szpigel, Optimal Train Scheduling on a Single Track Railway, in: M. Ross (Ed.), Operational Research '72, Amsterdam, The Netherlands, 1973, pp. 343–352.