Transportation Research Part B 44 (2010) 175–192

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

A tabu search algorithm for rerouting trains during rail operations

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

[a] Department of Transport and Planning, Delft University of Technology, Stevinweg 1, 2628 CN 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 degli Studi di Siena, via Roma 56, 53100 Siena, Italy

ARTICLE INFO

Article history:
Received 20 March 2008
Received in revised form 22 May 2009
Accepted 23 May 2009

Keywords:
Real-time railway traffic management
Scheduling
Routing
Alternative graph
Tabu search

ABSTRACT

This paper addresses the problem of train conflict detection and resolution, which is dealt every day by traffic controllers to adapt the timetable to delays and other unpredictable events occurring in real-time. We describe a number of algorithmic improvements implemented in the real-time traffic management system ROMA (Railway traffic Optimization by Means of Alternative graphs), achieved by incorporating effective rescheduling algorithms and local rerouting strategies in a tabu search scheme. We alternate a fast heuristic and a truncated branch and bound algorithm for computing train schedules within a short computation time, and investigate the effectiveness of using different neighborhood structures for train rerouting. The computational experiments are based on practical size instances from a dispatching area of the Dutch railway network and include complex disturbances with multiple late trains and blocked tracks. Several small instances are solved to optimality in order to compare the heuristic solutions with the optimum. For small instances, the new tabu search algorithms find optimal solutions. For large instances, the solutions generated by the new algorithms after 20 s of computation are up to more than 15% better than those achieved within 180 s by the previous version of ROMA.

1. Introduction

European railway companies face the challenge of accommodating the expected growth of transport demand while improving train punctuality. Since possibilities for large investments are minimal, a more efficient use of existing infrastructure is necessary. The usual way railways manage their performance is through carefully designed plans of operations, followed by real-time policies to manage disturbances. The strategy consists of the off-line development of detailed timetables, defining several months in advance routes, orders and timing for all running trains. Designing a timetable is usually a long-term procedure, and sophisticated decision support tools, based on mathematical programming techniques, have been proposed to help railway managers to optimize the use of infrastructure capacity and to distribute time margins between train paths, which may enable to absorb minor delays occurring during operations. For extensive literature reviews on the timetabling problem, we refer the reader to the surveys of Ahuja et al. (2005), Caprara et al. (2006), Hansen (2006) and D'Ariano (2008).

During rail operations, major disturbances may influence the off-line plan of operations, thus causing primary delays that propagate as consecutive, or secondary, delays to other trains in the network. In such situations short-term adjustments (i.e., train schedule modifications defined shortly before execution) are worthwhile in order to minimize the negative effects of the disturbances. This real-time process is called train conflict detection and resolution (CDR), and consists of changing dwell

* Corresponding author. Address: Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, via della vasca navale 79, 00146 Roma, Italy. Tel.: +39 06 5733 3225.
E-mail addresses: f.corman@tudelft.nl (F. Corman), a.dariano@tudelft.nl, a.dariano@dia.uniroma3.it (A. D'Ariano), pacciarelli@dia.uniroma3.it (D. Pacciarelli), pranzo@dii.unisi.it (M. Pranzo).

0191-2615/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trb.2009.05.004

times, train speeds as well as train orders and routes. The primary goal of reordering strategies is to reduce delay propagation, while the combined adjustment of train orders and routes allows to thoroughly reorganize the use of available resources in order to minimize train delays. For example, manual train rerouting is commonly practiced at the Swiss Federal Railways to control traffic in heavily used areas (Lüthi et al., 2007).

A recent stream of research on CDR focuses on detailed formulations based on the alternative graph of Mascis and Pacciarelli (2002). The first alternative graph formulation of the train scheduling problem with fixed routes was developed within the European project COMBINE (Mascis et al., 2008). Mazzarello and Ottaviani (2007) report on the practical implementation of the COMBINE system, using simple routing and sequencing algorithms, on a pilot site in The Netherlands. Flamini and Pacciarelli (2008) address the problem of routing trains through an underground rail terminus and develop a heuristic algorithm for a bicriteria version of the problem in which earliness/tardiness and train headways have to be optimized. D'Ariano et al. (2007) propose a branch and bound algorithm for the CDR problem with fixed routing. Their computational experiments, carried on the Dutch railway bottleneck around Schiphol International airport and for multiple delayed trains, show that optimal or near-optimal solutions can be found within a short computation time. In a follow-up paper (D'Ariano et al., 2008), the traffic management system ROMA (Railway traffic Optimization by Means of Alternative graphs) is described. In ROMA, this branch and bound algorithm is incorporated in a local search framework such that train routes are changed when better solutions can be achieved. Computational tests, carried on the Dutch dispatching area between Utrecht and Den Bosch, include instances with multiple delayed trains and different blocked tracks in the network. The results show that significant delay reduction is achieved by rerouting and rescheduling train movements, even though the benefit is mainly due to the sequencing optimization rather than to rerouting, particularly when dealing with heavy disruptions in the network. The latter paper left open two relevant issues. The first concerns the extent at which different neighborhoods or more sophisticated search schemes might improve upon the local minima found by the local search algorithm. The second issue is to study algorithmic improvements, in order to increase the solution quality and to reduce the computation time. Both these issues motivate the present paper. Precisely, the main contributions of this paper are the following.

- We develop two new routing neighborhoods of different size in order to search for more effective routings, and study their structural properties.
- We develop several new strategies for searching within these neighborhoods and incorporate them in a tabu search scheme.
- To evaluate the effectiveness of a move, we tested three different methods: a lower bound and two upper bounds on the objective function for each neighbor.
- We carried out an extensive campaign of experiments based on practical size instances referring to the Dutch dispatching area between Utrecht and Den Bosch. In our instances we include timetable disturbances, passenger connections, multiple delayed trains and heavy network disruptions. We compare the tabu search solutions with those obtained by the branch and bound algorithm of D'Ariano et al. (2007) and by the local search algorithm of D'Ariano et al. (2008). We also solve several small instances to optimality to compare the heuristic solutions with the optimum. For small instances, the new algorithms allow to close the optimality gap. For large instances, the new algorithms achieve significantly better results with respect to the previous version of ROMA within remarkably reduced computation times.

This paper is organized as follows. Section 2 discusses some related works, Section 3 provides a formal description of the CDR problem, since slightly different formulations of this problem are considered in the related literature. Section 4 deals with the algorithm development. Section 5 reports on our computational experiments. Section 6 presents some conclusions and subjects for further research.

2. Literature review

The literature on CDR experienced a slow growth from the pioneering paper of Frank (1966) and only in the recent years the research focused on detailed models and effective algorithms for the combined adjustment of train orders and routes.

Zwaneveld et al. (2001) study the routing of trains through railway stations for timetable design purposes, given a detailed layout of the stations and a draft timetable. Carey and Crawford (2007) face the same problem on a corridor including several busy complex stations. In both papers, the problem is modeled as a Mixed Integer Linear Program (MILP). Their algorithms are quite efficient but are not designed to be used in real-time.

Semet and Schoenauer (2005) study the problem of repairing a slightly perturbed timetable in order to minimize the total accumulated delay, defined as the difference between the actual time of arrival and the scheduled one. A local reconstruction of the schedule is based on adjustments of departure and arrival times at stations and allocation of resources. The problem is solved with a permutation-based evolutionary algorithm. The permutation of trains is achieved with a greedy heuristic that iteratively inserts trains in an initially empty schedule preserving the overall feasibility. Experimental results are presented on a large real-world test case with a single train being delayed for 10 min at a large connecting node, requiring timetable modifications of neighboring trains.

Goverde (2005) deals with stability analysis on a network level using Petri Net approaches. He conducts railway capacity assessments and derive structural and performance properties of Petri Nets, including the propagation of delays for a perturbed timetable.

Burkolter (2005) models deadlock avoidance in aggregated railway networks by using Petri nets, formulates a timetabling problem with cycle times similar to a job shop scheduling problem and solves the resulting train scheduling problem by a local search and simulated annealing in order to find a solution with minimum cycle time.

Caimi et al. (2005) address the problem of generating robust train routings through a station, given a timetable and the station layout. The problem is formulated via a node packing model (as in Zwaneveld et al., 2001) and a fixed-point iteration algorithm is adopted to compute an initial solution. A local search scheme is then applied to increase the length of the time slot of a chosen route, i.e., the time interval during which a delayed train may arrive and find its designated route still available. Their computational experience is based on the station of Bern, in Switzerland, and on a timetable of 19 trains arriving from six major directions in half an hour. Computing optimized solutions requires some hours of computation, but offers the chance to find delay-tolerant routings and to decrease the impact of late trains on rail traffic.

Törnquist and Persson (2007) address the real-time CDR problem at a network level by using a MILP model and carry on several experiments with a single delayed train. The instances are then solved with a general MILP solver.

Rodriguez (2007) focuses on the real-time CDR problem through junctions and proposes a constraint programming formulation for the combined routing and sequencing problem. His results show that a truncated branch and bound algorithm can find satisfactory solutions for a junction within computation time compatible with real-time purposes.

A more practical point of view of the train rescheduling problem is presented e.g., in Lüthi et al. (2007). The authors focused on analyzing the impact of real-time control of railway operations in heavily used areas of the Swiss railway network. They also identify a need for further research on the development of faster rescheduling algorithms capable to produce feasible schedules.

Montigel et al. (2007) report on the practical implementation of real-time rescheduling solutions in regular and emergency operations at the Lötschberg Base Tunnel in Switzerland. They state that considerable improvements of the railway traffic management can be achieved by a more efficient use of the track infrastructure.

From the above literature review it can be observed that there is a clear trend in academic literature towards the development of detailed models, able to incorporate practical requirements such as passenger connections, complex delay patterns and heavy network disruptions. At the same time, practical experience requires the development of more efficient algorithms that are able to produce effective and reliable solutions within a limited computation time.

3. CDR problem formulation

A railway network consists of track segments and signals. Signals allow to regulate traffic in the network by enforcing speed restriction to running trains. The track segment between two signals is called *block section*, and can host at most one train at a time. Traffic regulations also impose a minimum distance separation among the trains, which translates into a *setup time* between the exit of a train from a block section and the entrance of the subsequent train in the same block section.

For each train running in the network, an off-line developed timetable specifies a *path*, i.e., a sequence of block sections to be traversed, and planned arrival/passing times at a set of relevant points along its path (e.g., stations, junctions and the exit point of the network). Each train speed profile depends on the adopted *rolling stock*, i.e., on the train type and its acceleration and braking curves. The passing of a train through a particular block section is called *operation* and requires a given *minimum running time*, which depends on rolling stock and infrastructure characteristics, and is known in advance since railway traffic regulations impose strict restrictions to train speed profiles. Many timetables plan trains to travel at less than maximum speed, which means that the scheduled running time of each operation is larger than its minimum running time. These time reserves offer a flexibility that can be used by the dispatchers to regulate rail traffic in real-time.

We consider a timetable which describes the movement of all trains in the network during subsequent time periods. At each station, a stopping train is not allowed to depart from a platform before its scheduled departure time and is considered late if arriving at the platform later than its scheduled arrival time. Constraints due to rolling stock circulation and crew scheduling must also be taken into account. In our model these constraints require that a train can leave a station only a given time after the arrival of another train, to allow for car coupling/decoupling or crew movements. Additional constraints related to passenger satisfaction can also be included in as minimum transfer times between connected train services. These are the times required to allow passengers alight from one train, move to another platform track and board the other train.

Timetables are designed to satisfy all traffic regulations. However, in real-time, unexpected events occur that make the timetables infeasible. Real-time traffic management copes with these infeasibilities by adjusting the timetable of each train, in terms of routing and timing, and by resequencing the trains at each merging/crossing point. Its main goal is to minimize train delays (i.e., the difference between the arrival time at each relevant point in the new schedule and that in the timetable) while satisfying traffic regulations constraints and the compatibility with the real-time position of each train. The latter information enables the computation of the *release time* of each train, which is the minimum time at which the train can enter the network or reach the end of its current block section. The total accumulated delay of a train at a relevant point in the network is the difference between the arrival time of the train at that point in the CDR solution and that in the

timetable. This delay can be divided into two parts. Given the position of a train at time t_0, the *initial delay* (or primary delay) is the minimum delay of the train over all feasible schedules and speed profiles, i.e., the delay that cannot be recovered by rescheduling the train movements or by speeding up the trains. In other words, the initial delay is computed by using all the available time reserve, by considering the minimum running time rather than the scheduled running time for each train at each block section. The *consecutive delay* (or secondary delay) is the difference between the total accumulated delay and the initial delay, which is caused by the interaction with the other trains when managing traffic in real-time. It would be zero by always giving priority to this train over the others.

Let us introduce some definitions to formalize the CDR problem. We denote *route* the path taken by a particular train, i.e., the sequence of operations to be executed by the train in order to reach its destination. Given an operation i in a route, we denote with $\sigma(i)$ the operation which follows i on its route. A *timing* of a route specifies the starting time t_i of each operation in the route. Denoting with f_i the minimum running time of operation i, a timing is feasible if $t_{\sigma(i)} \geqslant t_i + f_i$, for every operation in the route. A *conflict* occurs whenever two or more trains traversing the same block section do not respect the minimum setup time required for that block section. A set of feasible route timings is *conflict-free* if, for each pair of operations associated to the same block section, the minimum setup time constraints are satisfied. In other words, if i and j are two operations associated with the entrance of two trains in same block section and f_{ij} is the setup time when i precedes j, the setup time constraint requires that the train associated to i must leave the block section at least f_{ij} time units before the train associated to j can enter the block section, i.e., $t_j \geqslant t_{\sigma(i)} + f_{ij}$. Similarly, if j precedes i, $t_i \geqslant t_{\sigma(j)} + f_{ji}$ must hold.

The CDR problem consists of choosing a route for each train and a timing for each chosen route such that all timings are conflict-free, no train enters the network before its release time and consecutive delays are minimized. In this paper, we address the minimization of maximum and average consecutive delays in lexicographic order, computed as the maximum and the average among all consecutive delays of all trains at all their respective relevant points, respectively. Specifically, we say that $[a;b] < [c;d]$ if $a < c$ or $a = c$ and $b < d$, and we use this notation to denote the lexicographic comparison.

The CDR problem can be partitioned into two subproblems. A rerouting problem, in which a route among a set of *rerouting possibilities* is associated to each train, and a scheduling problem in which the starting time of each operation is decided. In what follows, we refer to the problem of *conflict detection and resolution with fixed routes* (CDRFR) to denote the scheduling problem when trains are not allowed to change route and the objective function is minimization of the maximum consecutive delay.

The CDRFR problem can be formulated as a particular *disjunctive program*, i.e., a linear program with logical conditions involving operations "and" ($\wedge$, conjunction) and "or" ($\vee$, disjunction), as in Balas (1979).

$$
\begin{array}{lll}
\min & t_n - t_0 & \\
\text{s.t.} & t_j - t_i \geqslant w_{ij} & (i,j) \in F \\
 & (t_j - t_i \geqslant w_{ij}) \vee (t_k - t_h \geqslant w_{hk}) & ((i,j),(h,k)) \in A
\end{array}
$$

Here, a variable t_i, for $i = 1, \ldots, n-1$, is the starting time of operation i and corresponds to the entrance time of a train in the associated block section. Operation 0 is a dummy operation called *start* that precedes all the other operations. Similarly, n is a dummy operation called *end* that follows all the other operations.

Fixed constraints in F model feasible timings for the routes. For each operation i in a route there is a precedence relation $t_{\sigma(i)} \geqslant t_i + w_{i\sigma(i)}$, with $w_{i\sigma(i)} = f_i$. However, fixed constraints are also used to impose other constraints, e.g., the release times and the rolling stock connections, and to compute the train delays. Disjunctions in A represent conflict-free solutions. For each pair i and j of operations associated with the entrance of two trains in same block section, we introduce the disjunction $(t_j - t_{\sigma(i)} \geqslant w_{\sigma(i)j}) \vee (t_i - t_{\sigma(j)} \geqslant w_{\sigma(j)i})$, with $w_{\sigma(i)j} = f_{ij}$ and $w_{\sigma(j)i} = f_{ji}$.

Associating a node i with each variable t_i, the CDRFR problem can be usefully represented by an alternative graph $\mathcal{G} = (N, F, A)$ as in (Mascis and Pacciarelli, 2002), where N is the set of nodes, F is the set of conjunctions, or fixed arcs, and A is the set of disjunctions, or pairs of alternative arcs. Each arc (i,j), either fixed or alternative, is weighted with the quantity w_{ij}. Hence, in the alternative graph there are $n+1$ nodes associated to variables $t_0, t_1, \ldots, t_n$, where $n-1$ equals the sum over all trains of the number of block sections traversed by each train plus the number of scheduled stops, as described in (D'Ariano et al., 2007). The number of constraints in F increases linearly with n. However, we observe that the complexity of the problem is mainly related to the number of alternative pairs, that can be viewed as binary variables in an integer linear program. Their number can be computed as follows. Let us denote by s the overall number of block sections and by x_i the number of trains traversing block section $i = 1, \ldots, s$. Since there is an alternative pair for each pair of those trains, the overall number of alternative pairs is then $\sum_{i=1,\ldots,s} x_i(x_i - 1)$. As for the complexity of the CDRFR problem, even the feasibility problem of deciding whether a conflict-free schedule exists, given the initial positions of the trains, is NP-complete. This conclusion directly follows from an analogous result for the no-swap blocking job shop scheduling problem (Mascis and Pacciarelli, 2002) and it is proved in the train scheduling context by Strotmann (2007).

A *selection* S is a set of alternative arcs obtained by choosing at most one arc from each pair in A. The selection S is *consistent* if the graph $G(S) = (N, F \cup S)$ contains no positive length cycles, and we denote with $l^S(i,j)$ the length of a longest path from i to j in $G(S)$. The selection S is *complete* if exactly one arc from each pair is selected. A *solution* is a complete consistent selection S, and its value is the length $l^S(0,n)$ of the longest path from 0 to n. An optimal solution is such that $l^S(0,n)$ is minimum, and it is indicated as S^*. A schedule for the CDRFR problem is then obtained from a solution S by associating to each operation i the starting time $t_i = l^S(0,i)$. Since $G(S)$ is acyclic, computing all values $l^S(0,i)$, for $i = 1, \ldots, n$, can be done in time

$O(|F \cup S|)$. By choosing suitable weights on the arcs entering node n, as described in (D'Ariano et al., 2007), the maximum consecutive delay of the solution is then $l^S(0,n)$.

A more detailed description of the alternative graph formulation of railway traffic management constraints can be found in (D'Ariano, 2008; D'Ariano et al., 2008; D'Ariano et al., 2007; Mascis et al., 2008).

As far as the CDR problem is concerned, we observe that changing a route implies changing the sets F and A in $\mathscr{G}$. We denote with $\mathscr{F}$ the family of route-sets and with $F \in \mathscr{F}$ a specific route-set, i.e., a set of fixed arcs defining train routes. We call $A(F)$ the corresponding set of pairs of alternative arcs and $\mathscr{G}(F)$ the alternative graph associated with F. Let us also define a set of nodes $N(F)$ as the subset of N such that for each node $i \in N(F)$ there is at least an arc of F incident to node i. Set $N(F)$ includes the two dummy nodes 0 (start) and n (end). For each node $i \in N(F)$ there is a directed path from node 0 to node i and from node i to node n in the graph $G(\emptyset) = (N(F), F)$. We call a node $i \in N$ *isolated* if $i \notin N(F)$.

A solution t for the CDR problem is feasible if the following constraints are satisfied:

$$\bigvee_{F \in \mathscr{F}} \left(\bigwedge_{(i,j) \in F} (t_j - t_i \geqslant w_{ij}) \bigwedge_{((i,j),(h,k)) \in A(F)} ((t_j - t_i \geqslant w_{ij}) \vee (t_k - t_h \geqslant w_{hk})) \right)$$

A specific solution to the CDR problem is then a pair $(F, S(F))$, where $S(F)$ is a complete consistent selection in $\mathscr{G}(F)$. The CDR problem is significantly more difficult to solve at optimality than the CDRFR problem. In fact, the disjunctions associated to the choice of a route-set F from family $\mathscr{F}$ correspond to adding a large number of integer variables and constraints in an integer program. Specifically, the number of disjunctions in the CDR problem formulation increases exponentially with the number of trains. It follows that only small instances of the CDR problem can be solved at optimality and to the best of our knowledge no effective lower bounds are known for large instances.

Fig. 1 shows a simple railway network composed of 14 block sections and three trains T_A, T_B and T_C. For the sake of clarity, we only show the location of the most relevant block signals. In the timetable, the default route of T_A is given by the sequence of operations A1, A2, A3, A9, A12, A13, A14, even if also the routes A1, A2, A3, A9, A10, A5, A13, A14 and A1, A2, A3, A4, A5, A13, A14 are available for this train in order to reach the exit at block Section 14. The route of T_B is B7, B8, B9, B10, B5, B6 and the route of T_C is C11, C8, C9, C10, C5, C6. T_B and T_C share the same path from block Section 8 to 6, which is crossed by the timetable default route of T_A on block Section 9. T_B is a slow train entering block Section 7 at release time 0. Its minimum running time on each block section is 20 time units. T_A and T_C are fast trains requiring a minimum running time of 10 time units on each block section. Their release times are 60 and 40, respectively. There are only two points that are relevant for the timetable, namely the end of block Sections 6 and 14. The timetable requires that T_A, T_B and T_C exit the network within time 131, 160 and 122, respectively. We observe that each train, individually, would be able to exit the network on time, i.e., the primary delay of each train at its relevant point is zero.

Fig. 1 also presents the alternative graph model of the CDRFR problem from the proposed example. We denote a node with the pair (train, block section) of the associated operation or with the pair (train, exit point), except for the dummy nodes. Alternative pairs are depicted using dashed arcs and we assume their length (i.e., the setup time) always equal to zero. The three fixed arcs departing from node 0 model the release time of each train, whereas the arcs entering node n model the objective function. Since the primary delay is zero, the weight of the latter arcs is equal to the exit time of the associated trains from the network as specified in the timetable (but with negative weight). Note that the current alternative graph $\mathscr{G} = (N, F, A)$ has three isolated nodes (i.e., the nodes A4, A5 and A10) that can be used in order to implement alternative routes.

Fig. 2 (top) shows the optimal solution to the alternative graph of Fig. 1. In this schedule T_B always precedes T_C, while T_A follows the other two trains on block Section 9 (see Fig. 3). The longest path, highlighted in bold in Fig. 2 (top), passes through the nodes 0, B7, B8, B9, B10, B5, B6, Bout,C6,Cout and n. The optimal value $l^S(0,n)$ of the CDRFR problem is therefore 8.

Consider now the CDR problem, in which train routes (i.e., the set F of $\mathscr{G}$) can be changed, and let us choose the new route A1, A2, A3, A4, A5, A13, A14 for T_A (see Fig. 3). Clearly, a different alternative graph $\mathscr{G}' = (N, F', A')$ has to be adopted to

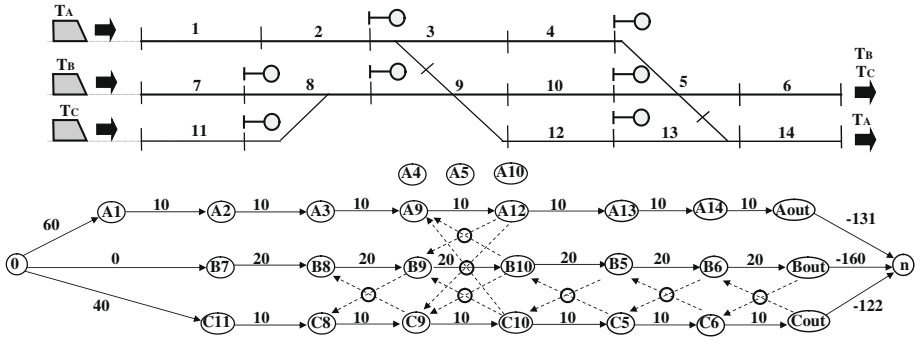

Fig. 1. A railway network with three trains and the alternative graph formulation.

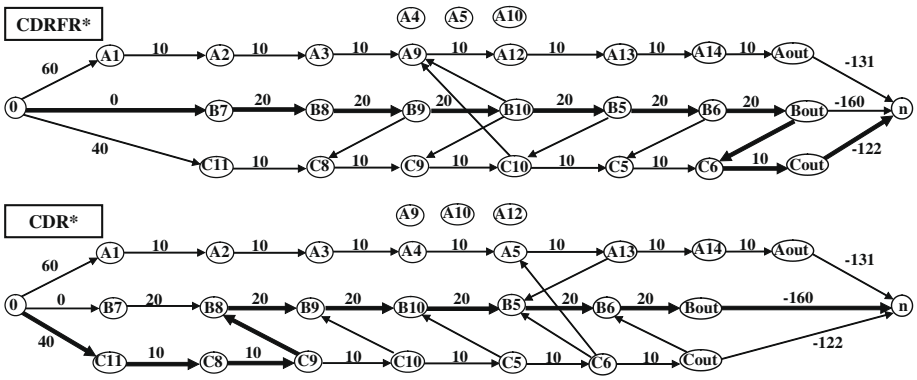

Fig. 2. Optimal CDRFR and CDR solutions for the proposed example.

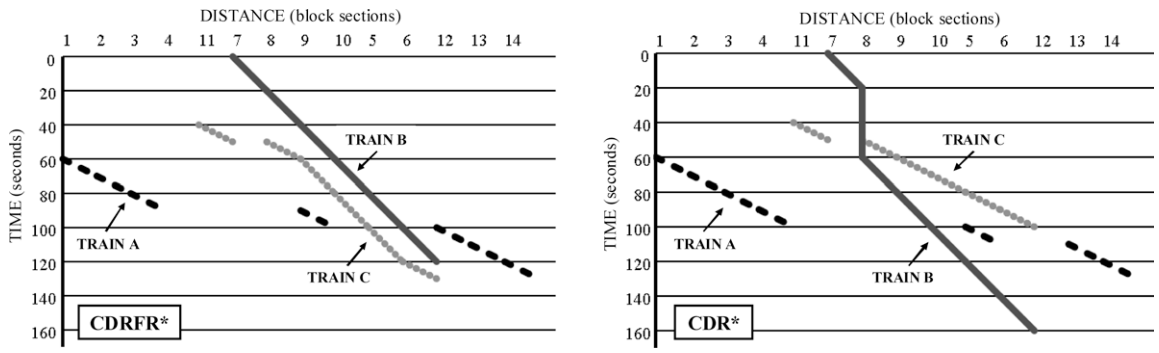

Fig. 3. Time–distance diagram of the optimal CDRFR and CDR solutions.

represent the new CDRFR problem, in which T_A crosses the routes of T_B and T_C at block Section 5. In the new alternative graph, the isolated nodes are $A9, A10$ and $A12$. Fig. 2 (bottom) shows the optimal solution to $\mathscr{G}'$, in which T_C precedes T_B, while T_A follows T_C and precedes T_B. The longest path of length 0, highlighted in bold in Fig. 2 (bottom), passes through the nodes 0, C11, C8, C9, B8, B9, B10, B5, B6, Bout and n. This solution is also optimal for the compound CDR problem, i.e., $(F^*, S^*(F^*))$.

4. Tabu search algorithm

In this section, we deal with the development of algorithms for solving the CDR problem. Since the combinatorial structure of the CDR problem is similar to that of the job shop scheduling problem with routing flexibility, we focus on the tabu search approach that achieved very good results with the latter problem (Mastrolilli and Gambardella, 2000).

The *tabu search* (TS) is a deterministic metaheuristic based on local search (Glover, 1986), which makes extensive use of memory for guiding the search. Basic ingredients of a tabu search are the concepts of move and tabu list, which restrict the set of solutions to explore. From the incumbent solution, non-tabu moves define a set of solutions, called the neighborhood of the incumbent solution. At each step, the best solution in this set is chosen as the new incumbent solution. Then, some attributes of the former incumbent are stored in a tabu list (TL), used by the algorithm to avoid being trapped in local optima and to avoid re-visiting the same solution. The moves in the tabu list are forbidden as long as these are in the list, unless an aspiration criterion is satisfied. The tabu list length can remain constant or be dynamically modified during the search.

We notice that, despite their similarity, there are significant differences between the CDR problem and the job shop scheduling problem, such as the absence of inter-machine buffers in the CDR problem, called no-store or blocking constraint (Hall and Sriskandarajah, 1996). As a result, most of the properties that are used in the job shop scheduling problem to design effective neighborhood structures do not hold for the CDR problem. Specifically, computing the value of the objective function after a local change, either train rerouting or reordering, may require a significant amount of time. Even the feasibility of a solution after a local change cannot be ensured as it occurs, e.g., in the job shop scheduling problem when reordering two consecutive operations laying on a critical path (Balas, 1969). For these reasons, our tabu search adopts a different searching scheme with respect to those mostly used for the flexible job shop problem (Mastrolilli and Gambardella, 2000).

In this paper, the neighborhood of an incumbent solution $(F, S(F))$ is composed of all sets of fixed arcs differing from F for exactly one train route. The basic move consists of locally changing the route of a single train in the set F in order to avoid

passing on a specific block section, thus obtaining a new set F' in the neighborhood. Given a neighbor F', the *evaluation* of the move corresponds to estimating the value of $(F', S(F'))$. When the best neighbor F' is chosen, according to some rule, the *implementation* of the move requires to minimize the maximum consecutive delay by solving the resulting CDRFR problem. This procedure is repeated until a stopping criterion is met. A general flowchart of the algorithm is shown in Fig. 4.

If no feasible solution $S(F)$ can be computed for a set F then the move is not allowed. This situation occurs e.g., when changing a train route causes a deadlock among a set of trains. In this case each train of the set claims the block section ahead which is not available due to the presence of another train in the set. A railway network is called deadlock-free if deadlocks can never occur.

The remaining part of this section is organized as follows. Section 4.1 briefly describes three methods for evaluating a move. Section 4.2 introduces three neighborhood structures of different size, analyzes their structural properties and defines strategies for exploring them. Finally, Section 4.3 gives the tabu search scheme.

4.1. Move evaluation

In this section, we define three methods for evaluating a move. The aim of move evaluation is to produce reliable estimates within a short computation time.

The first method computes a lower bound to the optimum of the CDRFR problem. This method requires to compute a single machine preemptive schedule with implications for each block section as described in (D'Ariano et al., 2007). We refer to this algorithm as LB.

The second method computes an upper bound to the optimum of the CDRFR problem. Letting T_i be the train with changed route, the aim of this method is to remove train T_i from the incumbent schedule and to reinsert it with the new route while leaving unchanged the part of the schedule not related to T_i. More formally, given the incumbent solution $(F, S(F))$ and a new route-set F', we compute $S(F')$ as follows. We first remove from $N(F)$ all the nodes associated to the old route of T_i, remove from $(F, S(F))$ all the fixed arcs and all the alternative arcs incident to a node associated to the old route, thus obtaining a reduced solution $(F'', S(F''))$. We then add to $(F'', S(F''))$ the nodes and the fixed arcs associated to the new route of T_i, thus obtaining the route-set F'. Then, we define a new alternative graph $\mathscr{G}(F') = (F', S(F''), A(T_i))$ in which $A(T_i)$ contains the pairs of alternative arcs associated to route T_i. A new solution is then computed on the resulting graph by using the greedy algorithm AMCC described in (Mascis and Pacciarelli, 2002), which selects one alternative arc at a time. At each iteration AMCC forbids the alternative arc which would introduce the largest consecutive delay in the current selection, thus selecting its paired alternative arc. We refer to this algorithm as UB1.

The third method computes a near-optimal solution to the CDRFR problem with the branch and bound algorithm described in (D'Ariano et al., 2007), truncating its execution time after 10 s. We refer to this algorithm as UB2.

Besides the use of UB2 for move evaluation purposes, we always use UB2 for move implementation. In fact, in our computational experiments (see Section 5), this method finds a proven optimal solution to the CDRFR problem in less than 2 s in most cases, but there are a few cases in which the computation time is overlong. Since we do not need proven optimality for move evaluation/implementation purposes, we truncate the execution of the branch and bound algorithm after 10 s.

4.2. Routing neighborhoods

This section describes three neighborhood structures of different size and combines two of them to form hybrid neighborhood structures. To this aim, we need to introduce some preliminary notation.

Given a solution S to the CDRFR problem and a node $i \in N(F) \setminus \{0, n\}$, let $\sigma^{-1}(i)$ denote the node which precedes i on its route. We say that i is a *critical node* in S if $l^S(0,i) + l^S(i,n) = l^S(0,n)$. A critical node i is a *waiting node* if $l^S(0,i) > l^S(0, \sigma^{-1}(i)) + w_{\sigma^{-1}(i)i}$, i.e., the associated train is waiting for some other train. For each waiting node i, there is always a node h_i in $\mathscr{G}(S)$, different from $\sigma^{-1}(i)$, such that $l^S(0,i) = l^S(0,h_i) + w_{h_i i}$. We call h_i the *hindering node* of i. By definition, for each waiting node $i \in N(F) \setminus \{0, n\}$ there is exactly one hindering node (possibly node 0).

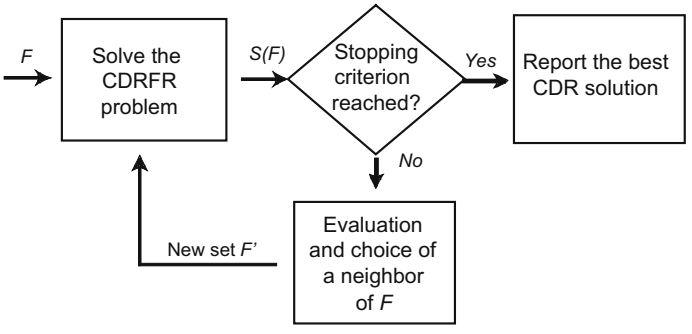

Fig. 4. Flowchart of the algorithm.

Given a node $i \in N(F) \setminus \{0, n\}$, we recursively define its *backward ramification* $R_B(i)$ as follows. If i is a waiting node, then $R_B(i) = R_B(\sigma^{-1}(i)) \cup R_B(h_i) \cup \{i\}$, otherwise $R_B(i) = R_B(\sigma^{-1}(i)) \cup \{i\}$. We define recursively the *forward ramification* $R_F(i)$ as follows. If i is the hindering node of a waiting node k (i.e., if $i = h_k$), then $R_F(i) = R_F(\sigma(i)) \cup R_F(k) \cup \{i\}$, otherwise $R_F(i) = R_F(\sigma(i)) \cup \{i\}$. Moreover, by definition, $R_B(0) = R_F(0) = \{0\}$ and $R_B(n) = R_F(n) = \{n\}$.

Given a solution $(F, S(F))$ to the CDR problem, we call *critical path set* $\mathscr{C}(F,S)$ the set of all critical nodes, *backward ramified critical path set* (BRCP) the set $\mathscr{B}(F,S) = \bigcup_{i \in \mathscr{C}(F,S)} [R_B(i)]$ and *forward backward ramified critical path set* (FBRCP) the set $\mathscr{R}(F,S) = \bigcup_{i \in \mathscr{C}(F,S)} [R_B(i) \cup R_F(i)]$.

For example, in the graph of Fig. 2 (bottom):

$\mathscr{C}(F,S) = \{0, C11, C8, C9, B8, B9, B10, B5, B6, Bout, n\}$,
$\mathscr{B}(F,S) = \{0, B7, B8, B9, B10, B5, B6, Bout, C11, C8, C9, n\}$;
$\mathscr{R}(F,S) = \{0, B7, B8, B9, B10, B5, B6, Bout, C11, C8, C9, C10, C5, C6, Cout, n\}$.

The intuition behind forward and backward ramifications is that there are several possibilities for reducing the maximum consecutive delay: (*i*) directly reroute the train T_i with the largest consecutive delay, (*ii*) reroute another train T_j which has precedence on T_i and contributes to its delay, (*iii*) anticipate the arrival time of T_j at the conflict point with T_i by rerouting another train T_k which has precedence on T_j before the conflict point with T_i. Trains T_i and T_j have at least one operation associated to a critical node. For the third case, trains are obtained from the backward ramification. Note that, by reversing the direction of every arc in a solution, we get a graph with the same maximum consecutive delay and critical path in which the former backward ramification is now a forward ramification. It follows that changing the route of a train in the forward ramification can also lead to a better solution.

We next describe three neighborhood structures. The two latter neighborhoods can be viewed as restricted versions of the first one.

$\mathscr{N}_C$. The *complete neighborhood* contains all the feasible solutions to the CDR problem in which exactly one train follows a different route compared to the incumbent solution. This is the largest neighborhood we consider.

$\mathscr{N}_{BRCP}$. With the *backward ramified critical path neighborhood* only the trains with at least one operation in $\mathscr{B}(F,S)$ can be rerouted. The idea is that the maximum consecutive delay of an optimal solution to the CDRFR problem can be reduced by removing one of the conflicts causing it. This requires either removing an operation from the critical path set (i.e., rerouting the associated train through a different block section) or anticipating its arrival at the conflict point. The latter result can be obtained by removing an operation from $\mathscr{B}(F,S)$ and then rescheduling train movements. This neighborhood structure has been studied in (D'Ariano et al., 2008) within a local search scheme.

$\mathscr{N}_{FBRCP}$. The *forward backward ramified critical path neighborhood* extends the previous neighborhood by considering also the forward ramifications, i.e., all the operations in $\mathscr{R}(F,S)$ and the associated trains. This neighborhood contains all the solutions to the CDR problem, in which one train associated to $\mathscr{R}(F,S)$ is rerouted. $\mathscr{N}_{FBRCP}$ is explored by alternating the rerouting of a train in the forward ramification to the rerouting of a train in the backward ramification.

In general, all the proposed neighborhoods enlarge common neighborhood structures for the job shop problem in order to avoid empty neighborhoods as far as possible. However, the search strategy adopted requires defining the sets F and $S(F)$ in succession. It is thus worthwhile studying specific properties of the different neighborhood structures.

A neighborhood structure $\mathscr{N}$ is called *opt-connected* if, starting from any feasible solution $(F, S(F))$ to the CDR problem, an optimal solution $(F^*, S^*(F^*))$ can be reached after a finite number of moves chosen in $\mathscr{N}$. In our case, $\mathscr{N}$ is explored by first defining a new route-set F and then computing $S(F)$. We must therefore take into account this exploration strategy when studying the different neighborhood structures.

We next show that $\mathscr{N}_C$ is opt-connected if for any route-set F' there exists a feasible schedule $S(F')$, which is always the case when the network is deadlock-free, i.e., when for each route-set F there exists a feasible $S(F)$. In such case, if $F \neq F^*$ then at least one train can be rerouted according to its route in F^*, thus leading from F to F^* after a number of moves smaller or equal to the number of trains. Then, in order to obtain $(F^*, S^*(F^*))$, it is sufficient to choose the selection $S^*(F^*)$ after the route-set F^* is reached. However, if deadlock situations may arise, opt-connectivity is not guaranteed.

An example of this situation is shown in Fig. 5, in which two trains, T_A and T_B, run in opposite direction on a single track line. Suppose that the optimal CDR solution requires T_A passing through block Sections 1, 2 and 4, and T_B passing through block Sections 4, 3 and 1. Consider now a feasible solution in which T_A and T_B are routed through block Sections 3 and 2, respectively. Clearly, changing only one route for T_A or T_B leads to a deadlock situation for which no feasible schedule exists.

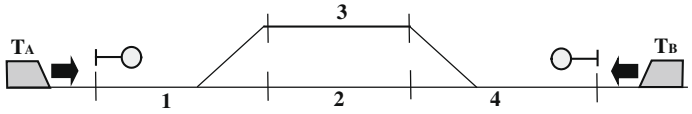

Fig. 5. A small example in which $\mathscr{N}_C$ is not opt-connected.

Thus, it is not possible to reach the optimal solution passing through a sequence of feasible intermediate solutions each differing from the previous one for at most one train route.

We next show that, even if for any route-set F' there exists a feasible schedule $S(F')$, the two restricted neighborhoods $\mathcal{N}_{BRCP}$ and $\mathcal{N}_{FBRCP}$ are not opt-connected even if the network is deadlock-free. To show this fact, consider the example described in Section 3. Analyzing the network in Fig. 1, it is straightforward to observe that only one route exists for trains T_B and T_C, while three routes are possible for train T_A (through block Section 4, 10 or 12), all leading to feasible schedules. The solution in Fig. 2 (top) can be improved only by changing the route of T_A, but no node of T_A belongs to $\mathcal{R}(F,S)$, thus implying that the two restricted neighborhoods are empty (the arcs in $\mathcal{R}(F,S)$ are depicted in bold in Fig. 6). In other words, starting from the solution in Fig. 2 (top), no move is allowed in $\mathcal{N}_{BRCP}$ or $\mathcal{N}_{FBRCP}$, and the optimal solution in Fig. 2 (bottom) cannot be reached.

The above discussion suggests research directions to design effective CDR algorithms. In what follows, we assume that for any route-set F' there exists a feasible schedule $S(F')$, and that the length of the tabu list is smaller than the number of trains that can be rerouted. The latter assumptions imply that there is always a non-tabu move in the complete neighborhood $\mathcal{N}_C$.

We next describe five strategies for exploring the neighborhood of the incumbent solution. The strategies are based on the intuition that the neighborhood $\mathcal{N}_{FBRCP}$ is likely to contain the most promising moves, but limiting the search to $\mathcal{N}_{FBRCP}$ can be too restrictive. On the other hand, since the evaluation of a single move is computationally expensive, using always $\mathcal{N}_C$ would be a too time consuming. Alternating the search in $\mathcal{N}_C$ and $\mathcal{N}_{FBRCP}$ enhances the chance to find better moves, makes the neighborhood opt-connected and enables some diversification in the search of new route-sets with respect to $\mathcal{N}_{FBRCP}$.

With all strategies, $\mathcal{N}_{FBRCP}$ and $\mathcal{N}_C$ are only partially explored to reduce the computation time at each iteration of the tabu search. At most ψ moves are randomly chosen in each neighborhood (with all trains and alternative routes having the same probability), where ψ is a parameter of the tabu search algorithm. Tabu moves are not evaluated unless an aspiration criterion is used. In the evaluation of a move it may happen that no feasible solution is found, in such cases the move is discarded from the neighborhood. Therefore, it is possible that the neighborhood becomes empty. When this occurs the algorithm implements a *restart move*, i.e., it changes γ routes at the same time, chosen randomly after clearing the tabu list, and implements the move with UB2. If UB2 fails in finding a feasible solution during the implementation of a move, a restart move is performed until a new feasible solution is found.

Complete. Candidate solutions in $\mathcal{N}_C$ are evaluated (either with LB, UB1 or UB2) and the best is chosen as the new incumbent.

Restart. Candidate solutions in $\mathcal{N}_{FBRCP}$ are evaluated (either with LB, UB1 or UB2) and the best is chosen as the new incumbent. If $\mathcal{N}_{FBRCP}$ is empty, $\gamma \geqslant 1$ consecutive moves are performed in $\mathcal{N}_C$ before searching again in the restricted neighborhood, where γ is a parameter of the tabu search algorithm. In each move, ψ randomly chosen routes are evaluated (either with LB, UB1 or UB2) and the best is chosen without implementing the move (i.e., without solving the CDRFR problem with UB2). A new schedule is then computed with the UB2 algorithm only after the γ moves.

Hybrid1. Candidate solutions in $\mathcal{N}_{FBRCP}$ are evaluated (either with LB, UB1 or UB2) and the best is chosen as the new incumbent. When $\mathcal{N}_{FBRCP}$ is empty, $\gamma > 1$ moves are performed in $\mathcal{N}_C$ before searching again in $\mathcal{N}_{FBRCP}$. In the latter case the move is chosen by evaluating ψ candidate solutions in $\mathcal{N}_C$ with UB2 and implementing the best one by taking into account the two objective functions in lexicographic order (i.e., the minimization of the [maximum; average] consecutive delays). After each move, a new schedule is computed with the UB2 algorithm.

Hybrid2. As in Hybrid1 but the candidate solutions in $\mathcal{N}_C$ are evaluated by taking into account only the average consecutive delay.

Hybrid3. Candidate solutions in $\mathcal{N}_{FBRCP}$ are evaluated (either with LB, UB1 or UB2) and the best is chosen as the new incumbent if it is an improving move. If a local minima is reached, ψ candidate solutions in $\mathcal{N}_C$ are evaluated (either with LB, UB1 or UB2) and the best solution found among those in $\mathcal{N}_{FBRCP} \cup \mathcal{N}_C$ is implemented. The two objective functions are considered in lexicographic order. After each move, a new schedule is computed with the UB2 algorithm.

Strategy Restart is a simple diversification strategy, in which γ moves are chosen without solving the CDRFR problem. Among the hybrid strategies, Hybrid1 and Hybrid3 minimize the two objective functions in lexicographic order. The choice

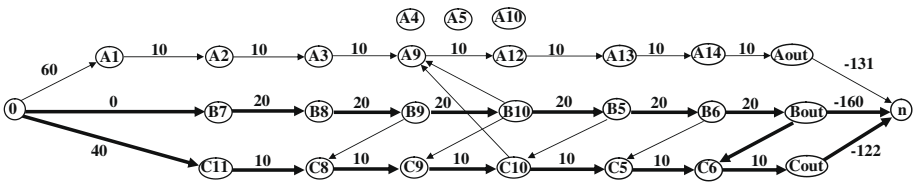

Fig. 6. FBRCP for the graph of Fig. 2 (top).

of a different objective function for strategy Hybrid2 is motivated by two observations. First, changing the objective function increases diversification, which is often the aim of restart actions in tabu search algorithms. Second, this choice allows to address directly the secondary objective function, which is useful when the primary objective function cannot be improved significantly (this is often the case when the restricted neighborhood is empty). Finally, the tabu search algorithm with strategy Hybrid3 is a *variable neighborhood tabu search*, tested on the median cycle problem by Moreno Pérez et al. (2003).

4.3. The tabu search scheme

The tabu search algorithm for the CDR problem with strategy Hybrid2 is described in Fig. 7. Similar schemes apply to the other neighborhood search strategies.

We indicate with $\mathcal{N}_{FBRCP}(IncSol, \psi)$ the neighborhood of the incumbent solution $IncSol = (F, S(F))$ containing at most ψ elements (ψ rerouting options). We denote with $\overline{T}^{S(F)} = \sum_{i:(i,n)\in F} \max\{0, l^{S(F)}(0,i) + w_{in}\}$ the sum of the positive consecutive delays of all running trains at their relevant points. Each solution is evaluated in $\mathcal{N}_{FBRCP}(IncSol, \psi)$ by considering the maximum

Algorithm TabuSearch
Input: an initial solution $(F, S(F))$
  $IncSol = (F, S(F))$
  $BestSol = (F, S(F))$
  $BestValue = [l^{S(F)}(0,n); \overline{T}^{S(F)}]$
  while time limit is not reached **do**
    if $\mathcal{N}_{FBRCP}(IncSol, \psi) = \emptyset$ **then** (explore the neighborhood $\mathcal{N}_C$)
      reset TL ($TL = \emptyset$)
      for γ iterations **do**
        $NextValue = [+\infty; +\infty]$
        while $\mathcal{N}_C(IncSol, \psi) \neq \emptyset$ **do**
          choose a non-tabu solution $(F'', S(F'')) \in \mathcal{N}_C(IncSol, \psi)$
          (i.e., such that the arcs in $F'' \setminus F$ are associated to a train not in TL)
          $\mathcal{N}_C(IncSol, \psi) = \mathcal{N}_C(IncSol, \psi) \setminus (F'', S(F''))$
          if $\overline{T}^{S(F'')} < NextValue[2]$ **then**
            $NextSol = (F'', S(F''))$
            $NextValue = [l^{S(F'')}(0,n); \overline{T}^{S(F'')}]$
          end if
        end while
        insert in TL the train having different route in $IncSol$ and $NextSol$
        $IncSol = NextSol$
        if $NextValue < BestValue$ **then**
          $BestSol = IncSol$; $BestValue = NextValue$
        end if
      end for
    else (execute a move in the neighborhood $\mathcal{N}_{FBRCP}$)
      $NextValue = [+\infty; +\infty]$
      while $\mathcal{N}_{FBRCP}(IncSol, \psi) \neq \emptyset$ **do**
        choose a non-tabu solution $(F', S(F')) \in \mathcal{N}_{FBRCP}(IncSol, \psi)$
        $\mathcal{N}_{FBRCP}(IncSol, \psi) = \mathcal{N}_{FBRCP}(IncSol, \psi) \setminus (F', S(F'))$
        if $[l^{S(F')}(0,n); \overline{T}^{S(F')}] < NextValue$ **then**
          $NextSol = (F', S(F'))$
          $NextValue = [l^{S(F')}(0,n); \overline{T}^{S(F')}]$
        end if
      end while
      insert in TL the train having different route in $IncSol$ and $NextSol$
      $IncSol = NextSol$
      if $NextValue < BestValue$ **then**
          $BestSol = IncSol$; $BestValue = NextValue$
      end if
    end if
  end while

Fig. 7. Pseudocode of the tabu search algorithm using strategy Hybrid2.

and average consecutive delays in lexicographic order. The value of a solution is therefore the pair $[l^{S(F)}(0,n);\overline{T}^{S(F)}\}]$. Solutions in $\mathcal{N}_C(IncSol,\psi)$ are evaluated by only considering the average delay.

The search process is halted when a time limit is reached. For all iterations, each neighbor $(F',S(F')) \in \mathcal{N}_{FBRCP}(IncSol,\psi)$ is generated by changing a train route of *IncSol*. If $\mathcal{N}_{FBRCP}$ is empty, the algorithm investigates $\mathcal{N}_C$ for γ iterations. The value of a neighbor $(F',S(F')) \in \mathcal{N}_C(IncSol,\psi)$ is evaluated by UB2, while the value of a neighbor $(F',S(F')) \in \mathcal{N}_{FBRCP}(IncSol,\psi)$ is computed by one of the three strategies described in Section 4.1. Specifically, when LB is used, only the maximum consecutive delay is evaluated while the average consecutive delay is not considered. Once the most promising route-set has been selected, a new incumbent solution is computed by solving the CDRFR problem with the UB2 algorithm and replaces, possibly, the current best solution. The route changed from the old to the new incumbent solution is added to the tabu list for λ iterations, where λ is the length of the tabu list. In our experiments, we also tested a tabu move that forbids all the routes associated to the rerouted train. In this case the tabu list contains the last rerouted trains rather than routes.

5. Computational experiments

This section presents our tests on the dispatching area of Utrecht Den Bosch, a bottleneck of the Dutch railway network. Section 5.1 describes the test case while Section 5.2 reports the advantages of using alternative algorithmic components in our tabu search. Section 5.3 then presents the performance of our neighborhood search strategies. We study complex perturbations (with several trains delayed at their entrance in the network) and disruptions (some tracks are blocked and therefore cannot be used). We also consider timetables with and without passenger connection constraints. Routing and scheduling algorithms are implemented in C++ language and executed on a PC equipped with a processor Intel Pentium D (3 GHz), 1 GB Ram and Linux operating system.

5.1. Test case description

The Utrecht Den Bosch railway infrastructure is shown in Fig. 8. This railway area is around 50 km long and consists of 191 block sections, including 21 platforms. There are two main tracks, divided into one long corridor for each traffic direction, a dedicated stop for freight trains and seven passenger stations: Utrecht Lunetten, Houten, Houten Castellum, Culemborg, Geldermalsen, Zaltbommel and Den Bosch. Each traffic direction has nine entrances: Utrecht, Dordrecht, Geldermalsen Yard, Nijmegen, Betuweroute, Oss, Den Bosch Yard, Eindhoven and Tilburg. There are several potential conflict points along each corridor due to different train speeds and critical crossing/merging points. Each train has a default route and a set of local rerouting options (highlighted in grey color in Fig. 8). Overall, 356 possible train routes are considered.

We evaluate a hourly cyclic timetable variant for year 2007 referred to peak hour. In the dispatching area around Geldermalsen 26 passengers and freight trains are scheduled in both directions. At Den Bosch station, the number of trains per hour increases up to 40. It follows that 1 h of traffic corresponds to a scheduling instance of around 3600 alternative pairs, the exact value depending on the route chosen for each train. We also include constraints on the minimum transfer time between connected train services. Precisely, rolling stock connections are provided in Zaltbommel and Den Bosch stations while passenger connections are modeled in Den Bosch station for the traffic directions from Oss to Utrecht and vice versa.

We next describe the disturbance schemes evaluated in this paper, in terms of entrance delays and blocked tracks. The timetable perturbations are in a time window of maximum entrance delay varying from 1000 up to 1800 s and the average entrance delay is around 320 s. The values of the entrance delay are randomly chosen in a time window of typical train delays for this railway network. The delayed trains are chosen among those scheduled in the first 30 min of the timetable while the time period of traffic prediction used in the experiments is 1 h long. We generate 24 instances with passenger and rolling

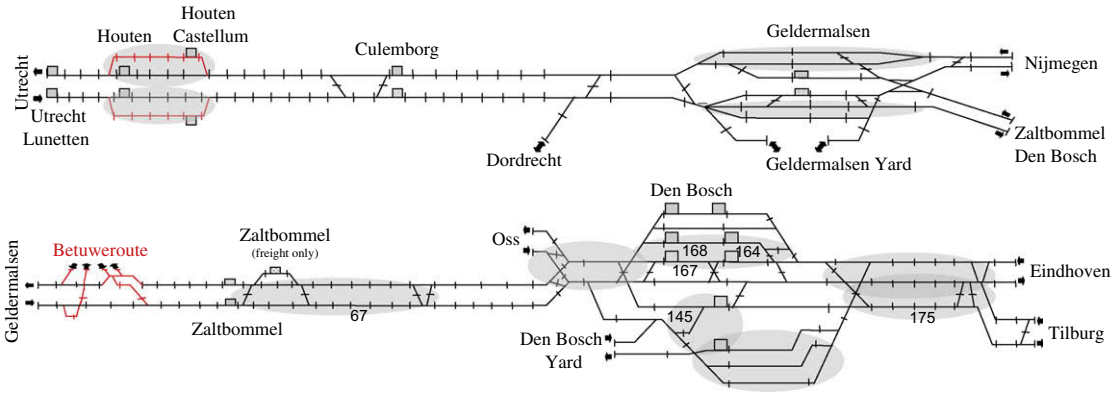

Fig. 8. Utrecht Den Bosch railway infrastructure.

stock connection constraints plus 24 instances with rolling stock connection constraints but without passenger connection constraints.

Table 1 presents three configurations of randomly generated blocked tracks (corresponding to the most time consuming disruptions studied by D'Ariano et al. (2008) and by D'Ariano (2008)). Each disruption (column 1) is obtained by making unavailable the set of block sections shown in column 2 (and also reported in Fig. 8). Column 3 shows the percentage of routes made unavailable by the blocked tracks.

5.2. Assessment of tabu search components

This section compares several alternative choices to configure our tabu search components. Table 2 shows all the alternative configurations considered in our assessment. Specifically, we analyze (i) the number ψ of neighbors to be evaluated at each iteration, (ii) the tabu list length λ, (iii) the number γ of moves to be performed in case of restart, (iv) the implementation of an aspiration criterion, (v) the tabu list structure, (vi) the algorithm for choosing the best candidate in the neighborhood. The assessment is based on 18 pilot instances, chosen among the 24 instances with rolling stock connection constraints and without passenger connection constraints. No disruption has been considered to avoid a reduction of the number of alternative routes.

We assess the following aspiration criterion: the tabu status of a move is ignored if the move improves the current best solution. This aspiration criterion requires the evaluation of tabu moves as an additional computational effort (tabu moves are not evaluated when aspiration is not used). Clearly, this effort is worthwhile only if this computational cost is compensated by significant improvements to the best solution.

As for the tabu list structure, we explore two possibilities: inserting a train in the tabu list or inserting a route in the tabu list. With the first option (Tabu train) all the routes of a tabu train are forbidden, while with the second, less restrictive, option (Tabu route) only the routes in the tabu list are forbidden for each train, while the others are allowed.

As for the neighbor evaluation algorithm, we analyze the three heuristics LB, UB1 and UB2 described in Section 4.1.

The best configuration for each neighborhood exploration strategy is as follows. For the Complete and Hybrid3 strategies the best values of (ψ, λ, γ) are $(8, 3, 5)$ and the best tabu structure is Tabu train. For the Restart, Hybrid1 and Hybrid2 strategies the best values of (ψ, λ, γ) are $(8, 27, 5)$ while the preferred tabu structure is Tabu route. Finally, the best configuration for all the strategies requires UB1 algorithm and aspiration criterion off.

It is interesting to show the results of our assessment for the three neighbor evaluation method more in details. Fig. 9 shows the average values found at time $t = 0, \ldots, 180$ s for the three methods. The average is computed on the values found by the five strategies using their best configuration on the 18 pilot instances. Table 3 shows the average computation time (in seconds) required by a single run of algorithms LB, UB1 and UB2, and the total number of calls to these algorithms within 180 s.

From the obtained results, it turns out that UB1 is always the best method, since it allows the evaluation of a larger number of moves within the same computation time and returns good quality solutions at each iteration. A drawback of UB1 is that it sometimes fails in finding a feasible solution (in such cases the neighbor is discarded from the neighborhood). LB is slightly faster than UB2 but less accurate, thus being less effective. Finally, UB2 produces the most accurate estimates but, being the most time consuming method, it causes the tabu search being considerably slower than when using UB1.

5.3. Disturbance management

In this section, we evaluate the five neighborhood search strategies described in Section 4.2. We analyze separately the 48 scenarios with perturbations described in Section 5.1 and the 144 scenarios associated to the same 48 perturbations and the three disruptions of Table 1.

Table 1
Description of the three disruptions.

Disruption	Unavailable block sections	% Unavailable routes
1	67 145 167	65.2
2	168 164 67 175	66.6
3	67	40.4

Table 2
Description of the tabu search parameters evaluated.

Neighborhood size (ψ)	Tabu length (λ)	Restart moves (γ)
3/ 5/ 8/ 10	3/ 5/ 8/ 10/ 15/ 20/ 27	1/ 3/ 5/ 8/ 10/ 20
Aspiration criterion	Tabu structure	Neighbor evaluation
On/ off	Tabu train/ tabu route	LB/ UB1/ UB2

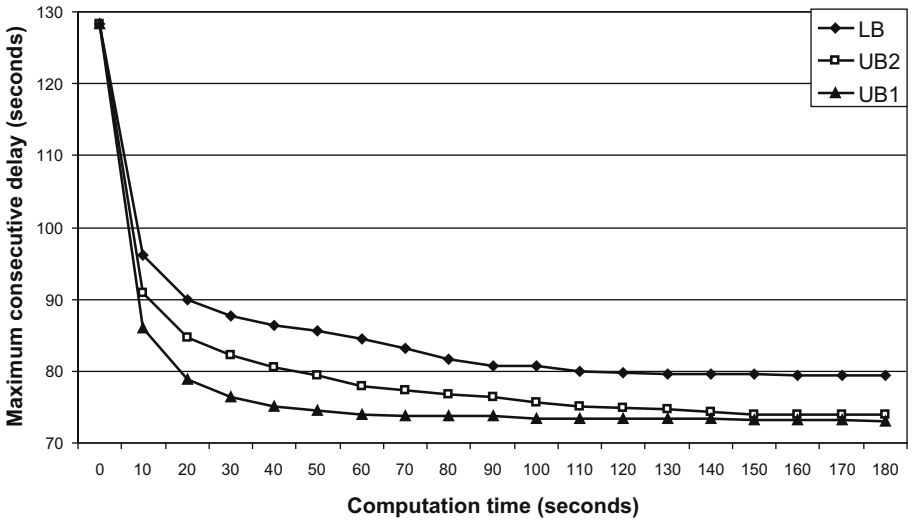

Fig. 9. Neighbors evaluation in terms of maximum consecutive delays.

Table 3
Other information on the neighbors evaluation.

Algorithm	Average computation time	Number of calls
LB	0.37	320
UB2	0.43	294
UB1	0.18	540

For each search strategy, the configuration of the tabu search algorithm is the best performing in the set of experiments of Section 5.2. In order to reduce the dependency from the random behavior of the search algorithms, we perform four runs for each instance by using different values of random seed, for a total of 768 runs for each version of the tabu search algorithm.

We compare the results obtained by the five versions of the tabu search algorithm with the performance of the local search algorithm of D'Ariano et al. (2008), based on the $\mathcal{N}_{BRCP}$ neighborhood, and of the proven optimal CDRFR solutions. In the latter case, each train can only travel on its default route, if available. For the scenarios with disruption, each non available default route is replaced with its shortest available alternative.

Figs. 10 and 11 illustrate the average results, in terms of the maximum and average consecutive delays, over the 48 perturbation schemes for the case without and with disruptions, respectively. At time $t = 0$, we report the average results on the proven optimal CDRFR solutions. The average solutions of the other rerouting algorithms are depicted after multiples of 10 s of computation, up to 3 min. These figures present, clearly, the benefits of rerouting trains to recover delays. In fact, the delay reduction when passing from the optimal CDRFR solutions to the best CDR solutions is evident. For the perturbation instances in Fig. 10, the drop after 180 s is about one half for both maximum and average consecutive delays.

All the five versions of the tabu search algorithm clearly outperform the local search algorithm. In Fig. 10, the reduction of the hybrid strategies after 180 s is about 20% for the maximum consecutive delay and 25% for the average consecutive delay. Similar qualitative results are obtained for the disruption instances in Fig. 11, with the difference that reduced rerouting possibility and higher density of trains in some block sections limit the improvement achieved with respect to the initial solution and to the local search. Even more effective is the improvement after 10–20 s of computation, which is a time limit more suitable for real-time purposes. After 10 s in Fig. 10 and after 20 s in Fig. 11, Hybrid1 and Hybrid2 find solutions that cannot be improved by the local search within 180 s of computation, with reference to both maximum and average consecutive delays.

All algorithms require a larger computation time to improve upon the initial solution when dealing with disruptions compared to the case without disruptions, despite the smaller number of rerouting options available in the former case. However, it should be noticed that also human dispatchers are slower in managing disruptions than perturbations, and scenarios with severe disruptions are those in which dispatchers need more support from computerized tools.

As for the comparison between the five versions of the tabu search algorithm, from an overall point of view the hybrid strategies are preferable with respect to the Restart and Complete strategies, thus demonstrating the advantage of using hybrid search. The Restart strategy is able to find good solutions within a short-time, but the improvement in terms of average delay is slower in the last 2 min of computation. The Complete strategy is slower in improving the initial solution with respect to Hybrid1 and Hybrid2, at least in terms of maximum consecutive delay, and it is dominated by Hybrid3

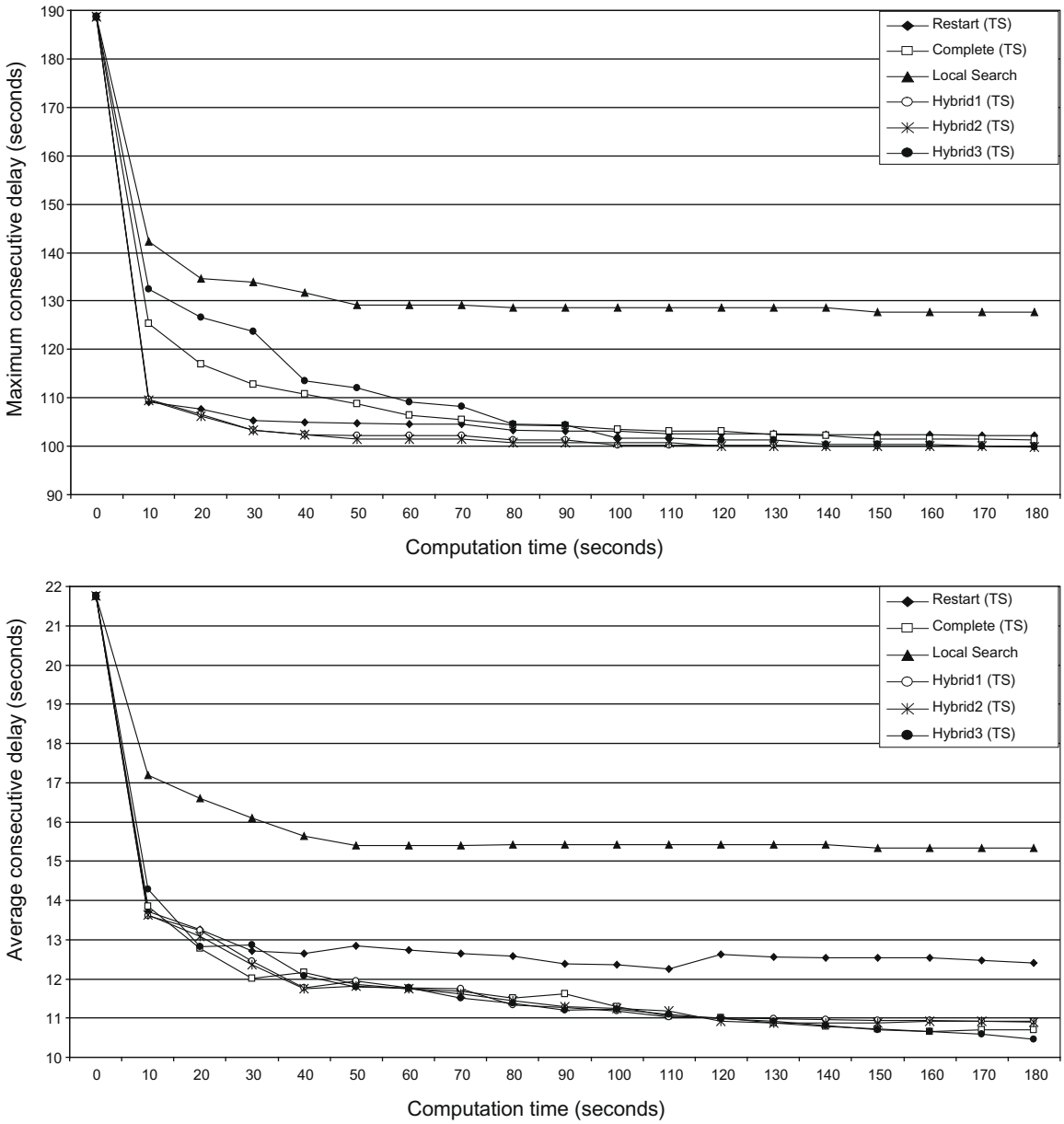

Fig. 10. Comparison of the algorithms in case of perturbations without disruptions.

after 180 s. This behavior confirms the intuition that searching with priority in $\mathcal{N}_{FBRCP}$ is preferable to searching in $\mathcal{N}_C$. Hybrid3 is slower with respect to Hybrid1 and Hybrid2 but the former dominates all the others after 180 s. In fact, differently from the other strategies, Hybrid3 searches for the best neighbor within 2ψ solutions at each local minimum. This makes Hybrid3 potentially two times slower than the others but offers more chances to find better neighbors as the time increases.

Table 4 presents the general behavior of different tabu search strategies after 180 s of computation. Results are divided in two groups. The first group (first five lines of results in table) refer to 48 perturbations scenarios without disruptions. The second group (last five lines in table) refer to 144 perturbations scenarios with disruptions.

Column 2 shows the average number of times the algorithm improves the current best solution over the 48 perturbations and 144 disruptions, while column 3 indicates the average number of train routes which are changed in the overall best solution compared to the routes used to compute the initial CDRFR solution. It is interesting to observe that the best solution is improved more frequently in the case of strategies Complete and Hybrid3. On the other hand, the final solutions for the other strategies present a smaller number of changed routes compared to the default routes, and should be therefore easier to

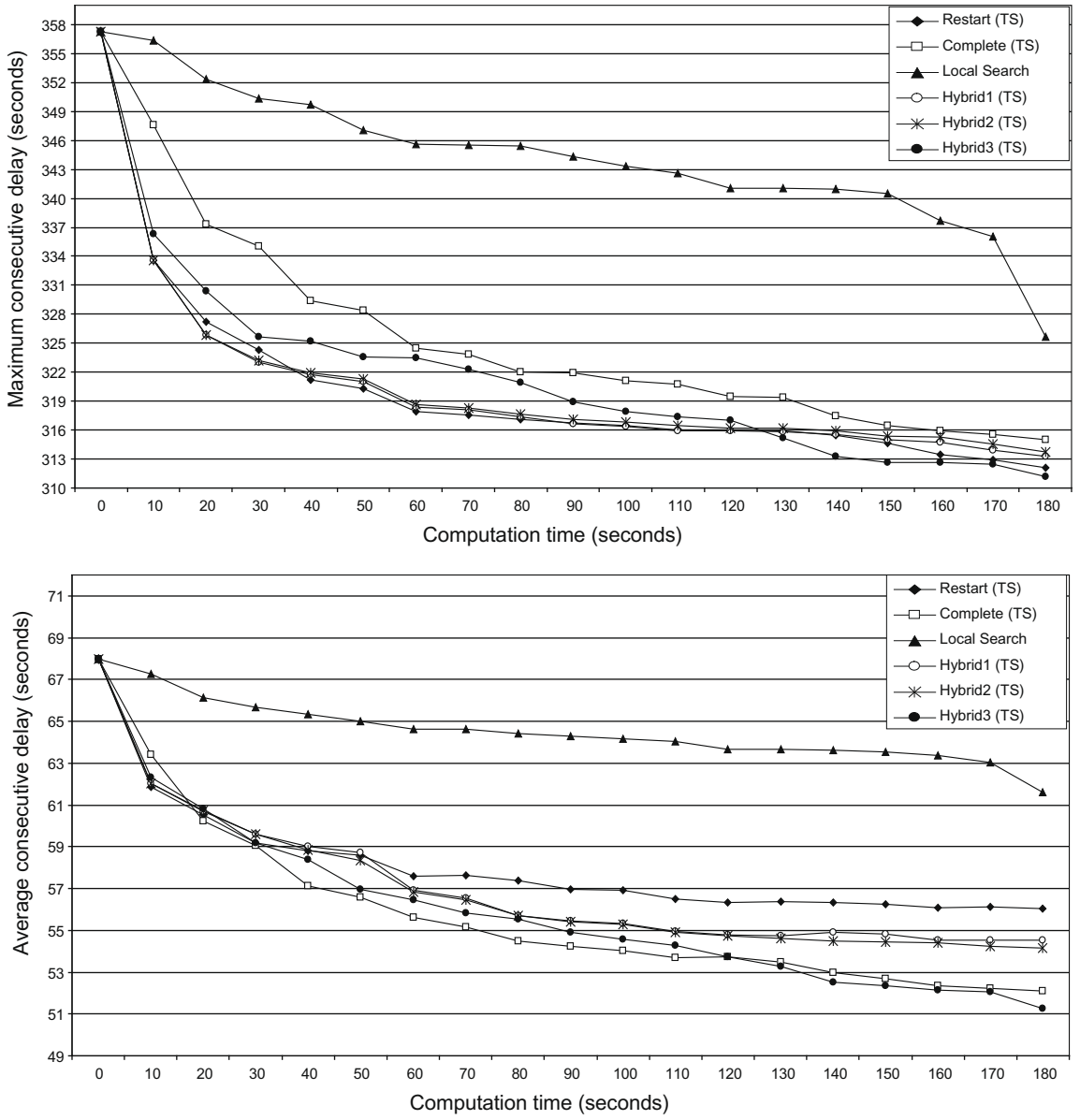

Fig. 11. Comparison of the algorithms in case of perturbations with disruptions.

implement by traffic controllers during rail operations. This behavior is probably due to the fact that strategies Complete and Hybrid3 search in the larger $\mathcal{N}_C$ neighborhood more frequently compared to the other strategies, as shown in Column 4.

Columns 4 and 5 report the average number of implemented moves chosen in $\mathcal{N}_C$ and $\mathcal{N}_{FBRCP}$, respectively. In Column 5 under strategy Hybrid3 we report only those moves that do no require searching in $\mathcal{N}_C$, i.e., only the moves that improve the current solution. It can be observed that, though only eight in the first group and three in the second, these moves allow to improve the current best solution in most cases. The higher number of moves carried out by strategy Restart is due to the fact that searching in $\mathcal{N}_C$ is particularly fast with this strategy since UB2 is executed only once out of γ moves, which leaves more time to search in $\mathcal{N}_{FBRCP}$.

Column 6 shows the average time (in seconds) spent by the tabu search to run algorithm UB2 within 180 s of computation. Column 7 reports the average number of times UB2 is truncated after 10 s of computation without finding a proven optimal solution. It can be noted that for the first group of instances this time limit is only rarely reached. On the other hand, the instances with disruptions are significantly harder, as also indicated in Column 6. This is probably due to the need of scheduling trains on a single track in both directions between Zaltbommel and Den Bosch, which makes these instances of the CDRFR problem particularly hard to solve to optimality.

Table 4
Comparison of tabu search neighborhood search strategies.

Neighborhood search strategy	Timetable perturbations without disruptions					
	Improving moves	Changed routes	Moves in $\mathcal{N}_C$	Moves in $\mathcal{N}_{FBRCP}$	Comput time UB2	Time limits UB2
Complete	7.5	14.6	138	0	15.2	0.1
Restart	4.8	9.3	81	185	23.0	0.3
Hybrid1	5.6	9.0	21	125	24.8	0.2
Hybrid2	5.5	8.9	21	121	26.1	0.8
Hybrid3	7.1	14.4	99	8	39.8	0.7
Complete	7.6	12.6	73	0	70.4	4.6
Restart	5.0	6.6	23	88	80.6	4.6
Hybrid1	5.8	7.0	10	57	87.3	4.6
Hybrid2	5.6	7.0	10	57	87.1	4.6
Hybrid3	7.0	10.3	25	3	102.4	5.4

To summarize, our tabu search algorithm clearly outperforms previously known algorithms (D'Ariano et al., 2008; D'Ariano et al., 2007) both in terms of computation time and solution quality. Overall, the new version of ROMA is able to provide an effective support for the delay management, both in case of timetable perturbations and disruptions, within a time limit compatible with the needs of rail operations.

5.4. Comparison with the optimal CDR solutions

We next compare the performance of the tabu search algorithm with the optimal CDR solutions. In this section we focus on minimization of maximum consecutive delay. To this end, we use a new set of 15 instances of limited size for which we have been able to compute a proven optimal solution to the CDR problem. Specifically, we consider the 13 trains running between Utrecth Lunetten and Den Bosch within the first 15 min of the timetable and compute the proven optimum to the CDRFR problem for every route-set. The procedure requires to solve to optimality more than 10^6 instances of the CDRFR problem for each instance of the CDR problem.

In Table 5 we compare the proven optimum to the CDR problem (denoted as CDR* in table) with the solutions found by the local search and the tabu search.

The first column describes each perturbation scenario. We denote each scenario with a three-field code A–B–C as follows. *A* indicates the type of distribution used to generate the train delays, uniform (U) or Gaussian (G). *B* represents the range $[0, max]$ in which we choose the delay values (in case of Gaussian distribution we accept a value only if it is within the range), for *max* varying in the set $\{600,\ 800, 1000, 1200, 1400, 1800\ s\}$. *C* is a serial number to distinguish scenarios with the same values for *A* and *B*. The other columns report the computation time (in seconds) and the maximum consecutive delay (in seconds) for the following algorithms: the branch and bound algorithm of D'Ariano et al. (2007) with no time limits (*CDRFR is the optimum to the CDRFR problem with default routes), the local search algorithm of D'Ariano et al. (2008), the tabu search algorithm with strategy Hybrid1 and the total enumeration algorithm (*CDR is the proven optimum to the CDR problem). For the tabu search, we report the computation time required to find the best solution and the total execution time, while we only show the total execution time for the other algorithms.

Table 5
Comparison between heuristic and optimal solutions.

Perturb instance ID	CDRFR*		Local search		Tabu search hybrid1			CDR*	
	Comp. time	Max. delay	Comp. time	Max. delay	Time to best	Comp. time	Max. delay	Comp. time	Max. delay
G-1800-1	0.01	145	0.02	145	1.00	180	88	42742	88
G-1800-2	0.01	105	0.03	103	0.81	180	31	42,010	31
G-1800-3	0.01	55	0.05	53	0.58	180	27	43,496	27
U-1000-1	0.01	79	0.02	79	0.41	180	10	44,711	10
U-1000-2	0.01	79	0.03	79	0.38	180	52	38,757	52
U-1000-3	0.01	76	0.03	74	0.88	180	26	39,409	26
U-1400-1	0.01	36	0.03	34	0.19	180	6	37,548	6
U-1400-2	0.01	51	0.05	51	0.45	180	44	38,795	44
U-1400-3	0.01	143	0.03	143	0.58	180	90	41,090	90
U-1800-1	0.01	105	0.03	103	0.70	180	0	37,354	0
U-1800-2	0.01	169	0.03	108	0.03	180	108	39,050	108
U-800-1	0.01	95	0.05	92	0.61	180	3	41,858	3
U-800-2	0.01	85	0.05	83	0.66	180	6	42,589	6
U-600-1	0.01	141	0.03	15	0.09	180	0	37,007	0
U-1200-1	0.01	99	0.03	99	0.89	180	63	43,565	63
Average	**0.01**	**97.53**	**0.03**	**84.07**	**0.55**	**180**	**36.93**	**40,665**	**36.93**

Despite the existing gap between the optimal CDRFR and CDR solutions (60.6 s on average), the tabu search is always able to find the optimum within one second of computation. The local search is only able to recover 13.46 s, which correspond to a relative error of more than 127%.

As for the other versions of the tabu search algorithm, the Complete and the three hybrid strategies always find the optimal CDR solution within a few seconds of computation. The Restart strategy finds an optimal solution only in 12 cases out of 15. The error after 180 s is 26 s for instance G-1800-3, 20 s for U-1000-3 and 1 s for U-800-2.

6. Conclusions

This paper investigates the effectiveness of advanced strategies to solve the compound train rerouting and rescheduling problem. Novel tabu search neighborhood structures and search strategies have been developed to this aim. The computational results, carried on real-world railway instances, demonstrate the high potential of rerouting trains in order to minimize consecutive delays. The new algorithms improve significantly the performance of the previous version of ROMA both in terms of solution quality and computation time. A specific analysis has been carried out on instances for which the proven optimum is known, showing that four of the five versions of our tabu search are able to find the optimum within a few seconds.

Further research is required on a number of issues. From the computational point of view, the development of exact CDR algorithms, able to find optimal solutions for large instances within acceptable computation time, is worthwhile. From the theoretical and practical points of view, it would also be interesting to study other performance indicators, such as the schedule robustness or the minimization of energy consumption, which require to deal with variable train dynamics and speed regulation. Another interesting research topic is to analyze the long-term effects of timetable modifications, i.e., the train delay propagation over different dispatching areas and time horizons.

Acknowledgements

This work has been partially funded by the Dutch government, program TRANSUMO "Reliability of transport chains", the Dutch foundation "Next Generation Infrastructures", and the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation".

The authors are also grateful to ProRail, the Dutch rail infrastructure manager, that provided the instances and to the anonymous referees for their helpful comments.

References

Ahuja, R., Cunha, C., Sahin, G., 2005. Network models in railroad planning and scheduling. In: Greenberg, H., Smith, J. (Eds.), TutORials in Operations Research, vol. 1. pp. 54–101.

Balas, E., 1969. Machine sequencing via disjunctive graphs: an implicit enumeration approach. Operations Research 17 (6), 941–957.

Balas, E., 1979. Disjunctive programming. Annals of Discrete Mathematics 5 (1), 3–51.

Burkolter, D.M., 2005. Capacity of railways in station areas using Petri nets. Dissertation number 16254, Swiss Federal Institute of Technology, Zurich, Switzerland.

Caimi, G., Burkolter, D. Herrmann, T., 2005. Finding delay-tolerant train routings through stations. In: Fleuren, H. (Ed.), Selected Papers of the Annual International Conference of the German Operations Research Society Jointly Organized with the Netherlands Society. pp. 136–143.

Caprara, A., Kroon, L., Monaci, M., Peeters, M., Toth, P., 2006. Passenger railway optimization. In: Barnhart, C., Laporte, G. (Eds.), Handbooks in Operations Research and Management Science, vol. 14. pp. 129–187.

Carey, M., Crawford, I., 2007. Scheduling trains on a network of busy complex stations. Transportation Research Part B 41 (2), 159–178.

D'Ariano, A., 2008. Improving real-time train dispatching: models, algorithms and applications. TRAIL Thesis Series, T2008/6, The Netherlands.

D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains in a railway network. European Journal of Operational Research 183 (2), 643–657.

D'Ariano, A., Corman, F., Pacciarelli, D., Pranzo, M., 2008. Reordering and local rerouting strategies to manage train traffic in real-time. Transportation Science 42 (4), 405–419.

Flamini, M., Pacciarelli, D., 2008. Real time management of a metro rail terminus. European Journal of Operational Research 189 (3), 746–761.

Frank, O., 1966. Two-way traffic on a single line of railway. Operations Research 14 (5), 801–811.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13 (5), 533–549.

Goverde, R.M.P., 2005. Punctuality of railway operation and timetable stability analysis. TRAIL Thesis Series T2005/10, The Netherlands.

Hall, N.J., Sriskandarajah, C., 1996. A survey on machine scheduling problems with blocking and no-wait in process. Operations Research 44 (3), 510–525.

Hansen, I.A., 2006. State-of-the-art of railway operations research. In: Allan, J., Brebbia, C., Rumsey, A., Sciutto, G., Sone, S., Goodman, C. (Eds.), Computers in Railways X. WIT Press, Southampton, UK, pp. 565–577.

Lüthi, M., Medeossi, G., Nash, A., 2007. Evaluation of an integrated real-time rescheduling and train control system for heavily used areas. In: Hansen, I.A., Radtke, A., Pachl, J., Wendler, E. (Eds.), Proceedings of the 2nd International Seminar on Railway Operations Modelling and Analysis, Hannover, Germany.

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

Mascis, A., Pacciarelli, D., Pranzo, M., 2008. Scheduling models for short-term railway traffic optimization. In: Hickman, M., Mirchandani, P., Voß, S. (Eds.), Lecture Notes in Economics and Mathematical Systems 600 (1): Computer-Aided Systems in Public Transport. Springer, pp. 71–90.

Mastrolilli, M., Gambardella, L.M., 2000. Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling 3 (1), 3–20.

Mazzarello, M., Ottaviani, E., 2007. A traffic management system for real-time traffic optimisation in railways. Transportation Research Part B 41 (2), 246–274.

Montigel, M., Kleiner, C., Achermann, E., 2007. Rescheduling in the ETCS Level 2 environment. In: Hansen, I.A., Radtke, A., Pachl, J., Wendler, E. (Eds.), Proceedings of the 2nd International Seminar on Railway Operations Modelling and Analysis, Hannover, Germany.

Moreno Pérez, J.A., Moreno-Vega, J.M., Rodríguez Martín, I., 2003. Variable neighborhood tabu search and its application to the median cycle problem. European Journal of Operational Research 151 (2), 365–378.

Rodriguez, J., 2007. A constraint programming model for real-time train scheduling at junctions. Transportation Research Part B 41 (2), 231–245.
Semet, Y., Schoenauer, M., 2005. An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 1761–11768.
Strotmann, C., 2007. Railway scheduling problems and their decomposition. Ph.D. Thesis, Universität Osnabrück, Germany.
Törnquist, J., Persson, J.A., 2007. N-tracked railway traffic re-scheduling during disturbances. Transportation Research Part B 41 (3), 342–362.
Zwaneveld, P.J., Kroon, L.G., van Hoesel, S.P.M., 2001. Routing trains through a railway station based on a node packing model. European Journal of Operational Research 128 (1), 14–33.