A tabu search algorithm for rerouting trains during rail operations

Francesco Corman\textsuperscript{1}, Andrea D’Ariano\textsuperscript{1,2}, Dario Pacciarelli\textsuperscript{2}, Marco Pranzo\textsuperscript{3}

RT-DIA-127-2008 Giugno 2008

(2) Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, via della vasca navale, 79 - 00146 Roma, Italy.
(3) Dipartimento di Ingegneria dell’Informazione, Università di Siena, via Roma, 56 - 53100 Siena, Italy.

This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB “Advanced tracking system in intermodal freight transportation”
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 during operations. Traffic controllers do not frequently use rerouting, although it would enable a better use of railway infrastructure. This is mainly due to the complexity of rescheduling train operations after major routing modifications. We describe a number of algorithmic improvements implemented in the real-time traffic management system ROMA (Railway traffic Optimization by Means of Alternative graphs). We incorporate 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. Our computational experiments are based on practical size instances from a dispatching area of the Dutch railway network. We study complex disturbances with multiple late trains, and also address the problem with blocked tracks, which force several trains to use the same track in both traffic directions. Comparison with optimal rescheduling solutions, provided by an exact algorithm with fixed routing, and local search rerouting solutions demonstrates the effectiveness of the new tabu search algorithm to reduce delays and to improve the use of infrastructure capacity.
1 Introduction

European railway companies must 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 infrastructure is necessary. The usual way railways manage the performance is through carefully designed plans of operations, followed by real-time policies to manage disturbances. The strategy consists of the offline 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 Assad [2], Cordeau et al. [8], Crainic [9], Ahuja et al. [1], Caprara et al. [5], Hansen [19] and D’Ariano [10].

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, the timetable no longer provides an optimal use of infrastructure capacity, and 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 times 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 reoptimizing the use of available railway capacity after disturbances.

The literature on CDR experienced a slow growth from the pioneering paper of Frank [15] 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. [30] 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 [6] face the same problem on a corridor including several busy complex stations. Their algorithms are quite efficient but are not designed to be used in real-time. In both papers, the problem is modeled as a Mixed Integer Linear Program (MILP).

Semet and Schoenauer [28] study the problem of repairing a slightly perturbed timetable to minimize the total accumulated delay. 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 minutes at a large connecting node, requiring timetable modifications of neighbouring trains.

Caimi et al. [4] 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. [30]) 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 require some hours of computation, but offer the chance to find delay-tolerant routings and to decrease the impact of late trains.

Törnquist and Persson [29] 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 [27] focuses on the real-time CDR problem through junctions and proposes a constraint programming formulation for the combined routing and sequencing problem. Its 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 different research direction focuses on detailed formulations based on the alternative graph of Mascis and Pacciarelli [22]. The first alternative graph formulation of the train scheduling problem with fixed routes was developed within the European project COMBINE [23]. Mazzarello and Ottaviani [25] 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 [14] 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. [12] 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 short computation time. In a follow-up paper [11], 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 achievable. Computational tests, carried on the Dutch dispatching area between Utrecht and Den Bosch, consider 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 relevant questions. 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 question is to study algorithmic improvements, in order to increase the solution quality and reduce the computation time. These issues motivate the present paper.

In this work, we address the minimization of maximum and average consecutive delays in lexicographic order. We explore the effectiveness of extensive rerouting strategies by incorporating the search for new routes in a tabu search scheme, in order to escape from local minima. We also report on several algorithmic improvements that allow to speed up the algorithm considerably. Our tabu search is based on a fast heuristic evaluation of the effectiveness of a move and on a hybrid neighborhood scheme. Specifically, we alternate the search for promising moves in two neighborhoods of different size, similarly to the variable neighborhood tabu search developed by Moreno Pérez et al. [26].

Computational experiments are performed on practical size instances referring to the Dutch dispatching area between Utrecht and Den Bosch. We compare tabu search solutions with those obtained by the branch and bound scheduling algorithm with fixed
routing of [12] and by the local search of [11]. We manage timetable disturbances with passenger connections, multiple delayed trains and heavy network disruptions.

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

2 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. 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). The passing of a train through a particular block section is called operation and requires a given 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. Traffic regulations also impose a minimum distance separations among the trains, which translates into a minimum setup time between the exit of a train from a block section and the entrance of the subsequent train in the same block section.

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. Following the notation of [12], the delay of a train at a relevant point in the network is divided into two parts. The primary or initial delay is caused by failures and disturbances and cannot be recovered rescheduling train movements. The secondary or consecutive delay is caused by the interaction with the other trains when managing traffic in real-time and it would be zero by always giving priority to this train over the others.

Let us introduce some definition to formally define the CDR problem. We denote route the traversing of a given path for a given train, i.e., the sequence of operations to be executed in order to traverse the path. Given an operation $i$ on 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 running time of operation $i$, a timing is feasible if $t_{\sigma(i)} \geq t_i + f_i$, 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., $t_j \geq t_{\sigma(i)} + f_{ij}$. Similarly, if $j$ precedes $i$ $t_i \geq t_{\sigma(j)} + f_{ji}$ holds.

The CDR problem consists of choosing a route for each train and conflict-free timings for all chosen routes such that setup times between operations are satisfied, 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. Specifically, we say that $[a; b] < [c; d]$ if $a < c$ or if $a = c$ 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 times of each operation are decided. In what follows, we refer to the problem of conflict detection and resolution with fixed routes (CDRFR) to denote the scheduling problem when train routes are fixed, with the objective function of minimizing the maximum consecutive delay.

![Image of a railway network with three trains](image)

Figure 1: A small railway network with three trains

Figure 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 running time on each block section is 20 time units. $T_A$ and $T_C$ are fast trains requiring a running time of 10 time units on each block section. Their release time is 60 and 40, respectively. There are only two points that are relevant for the timetable, namely the exit 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.

We use the alternative graph model [22] to formulate the CDRFR problem. An alternative graph is a triple $G = (N, F, A)$, where $N$ is a set of nodes, $F$ is a set of fixed arcs, and $A$ is a set of pairs of alternative arcs. Each arc $(i, j)$, either fixed or alternative, is weighted with a given quantity $w_{ij}$.

We denote with $N(F)$ a subset of $N$ such that for each node $i \in N(F)$ there is at least an arc of $F$ adjacent (connected) to node $i$. Set $N$ includes two dummy nodes 0 and
n, called start and end respectively, such that 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 \( G(\emptyset) = (N, F, \emptyset) \). We call a node isolated if this is element of \( N \) but not element of \( N(F) \). Note that the dummy nodes 0 and \( n \) are isolated only if there are no other nodes in \( N \).

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 \( l^S(0, n) \). An optimal solution is indicated as \( S^* \).

In our formulation of the CDRFR problem, an operation \( i \) is associated to a node \( i \) of the alternative graph. Two additional dummy nodes 0 and \( n \) represent the start and end of the schedule and each train route is modeled with a chain of fixed arcs from 0 to \( n \). A fixed arc \((i, j)\) corresponds to the precedence relation \( t_j \geq t_i + w_{ij} \). Fixed arcs model feasible timings for the routes, i.e., for each operation \( i \) in a route there is a fixed arc \((i, \sigma(i))\) with weight \( w_{\sigma(i)} = f_i \). However, fixed arcs are used also to impose other constraints, e.g. the release times and the rolling stock connections, and to compute the train delays. Alternative arcs are needed to 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 pair of alternative arcs \(((\sigma(i), j), (\sigma(j), i))\), with \( w_{\sigma(i)j} = f_{ij} \) and \( w_{\sigma(j)i} = f_{ji} \). A more detailed description of the alternative graph formulation of railway traffic management constraints can be found in [23, 12, 13, 11, 10]. With this notation, a solution to the CDRFR problem is obtained from a complete consistent selection \( S \) by associating to each operation \( i \) the starting time \( t_i = l^S(0, i) \). Therefore, in what follows we call solution to CDRFR the set \( S \). By choosing suitable weights on the arcs entering node \( n \), as described in [12], the maximum consecutive delay of the solution is then \( l^S(0, n) \).

As far as the CDR problem is concerned, we observe that changing a route implies changing the set \( F \) in \( G \) and changing the set \( A \) accordingly. Hence, when dealing with this problem, we call \( F \) a route-set, \( S(F) \) a complete consistent selection in \( G \) for a given \( F \) and \((F, S(F))\) a solution to the CDR problem.

**Figure 2:** Alternative graph formulation of the proposed example

Figure 2 shows the alternative graph model of the CDRFR problem from the example of Figure 1. In Figure 2, we denote a node with the pair (train, block section) of the associated operation or with the pair (train, exit point), except for dummy nodes. Alternative pairs are depicted using dashed arcs and for simplicity 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. Note that the current alternative graph \( G = (N, F, A) \) has three isolated nodes (i.e., the
nodes $A_4$, $A_5$ and $A_{10}$) that can be used in order to implement alternative routes.

Figure 3: Optimal rescheduling solution of the proposed example

Figure 3 shows the optimal rescheduling solution to the alternative graph of Figure 2. In this schedule $T_B$ always precedes $T_C$, while $T_A$ follows the other two trains on block section 9. The longest path passes through the nodes 0, $B_7$, $B_8$, $B_9$, $B_{10}$, $B_5$, $B_6$, $B_{out}$, $C_6$, $C_{out}$, 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 $G$) can be changed, and let us choose the new route $A_1$, $A_2$, $A_3$, $A_4$, $A_5$, $A_{13}$, $A_{14}$ for $T_A$ (see the network of Figure 1). Clearly, a different alternative graph $G' = (N,F',A')$ has to be adopted to 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 $A_9$, $A_{10}$ and $A_{12}$. Figure 4 shows the optimal solution to $G'$, in which $T_C$ precedes $T_B$, while $T_A$ follows $T_C$ and precedes $T_B$. The longest path, of length 0, passes through the nodes 0, $C_{11}$, $C_8$, $C_9$, $B_8$, $B_9$, $B_{10}$, $B_5$, $B_6$, $B_{out}$ and $n$. This solution is also optimal for the compound CDR problem, i.e., $(F^*,S^*(F^*))$.

3 Tabu search algorithm

In this section, we deal with the development of algorithms for solving the CDR problem. Specifically, we study a tabu search approach for this problem, since its combinatorial structure is similar to that of the job shop scheduling problem with routing flexibility and the tabu search approach achieved very good results with the latter problem [24].

The Tabu Search (TS) is a deterministic metaheuristic based on local search [16, 17], 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, 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 [20, 18]. 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 reliable estimates of the global impact of 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 the critical path [3]. For these reasons, our tabu search adopts a different searching scheme with respect to those mostly used for the flexible job shop problem [24].

In this paper, the basic move consists of locally changing the route of a single train in order to avoid passing on a specific block section. Given a route-set $F$, we evaluate the quality of a solution by computing a new solution $S(F)$ to the CDRFR problem. If no feasible solution $S(F)$ can be computed for a route-set $F$ then the move is not allowed, which occurs e.g. when changing a train route leads to a deadlock situation. Specifically, we analyze three methods for computing $S(F)$ as described in Section 3.1. The first method produces a near-optimal solution to the CDRFR problem by the branch and bound algorithm described in [12]. The other two methods aim at reducing the computation time and produce upper and lower bounds to the optimum of the CDRFR problem. Since in all the three cases the evaluation of a single move is still computationally expensive, we restrict the number of solutions that have to be evaluated by reducing the neighborhood size. Moreover, we avoid the use of aspiration criteria, which in our computational experience do not improve the solution quality (as reported in Section 4.2).

The remaining part of this section is organized as follows. Section 3.1 briefly describes the different scheduling algorithms adopted to compute upper and lower bounds to the CDRFR problem, i.e. when the route-set $F$ is given. Section 3.2 introduces three neighborhood structures of different size while Section 3.3 gives the tabu search scheme.

### 3.1 Upper and lower bounds to the CDRFR problem

As for the lower bound, we compute for each block section a lower bound derived from the single machine Jackson’s Preemptive Schedule (JPS) [21] and adapted to deal with the CDRFR problem [12]. The idea is to view a block section as a machine, which can process at most one train at a time for a given processing time. Each train cannot enter the block section before a known release time, and the maximum consecutive delay cannot be smaller than the block section exit time, plus a known tail. Release times, processing times and tails can be quickly computed on $G(\emptyset)$. Then, a lower bound to the maximum consecutive delay can be computed in $O(c \log c)$ time units for each block section with the algorithm described in [7], with $c$ equal to the number of trains traversing the block section. In Section 4, we refer to this algorithm as LB.

As for the upper bound, we use the following strategy for computing $S(F')$ for a given incumbent solution $(F, S(F))$ and a new route-set $F'$, differing from $F$ for the route of a single train $T_i$. We first exclude from $G(S)$ all the nodes associated to the old route
of $T_i$, all the fixed arcs in $F$ and all the alternative arcs in $S(F)$ incident in a node $i \in N(F)$, thus obtaining a reduced solution $(F'', S(F''))$. We then add to $(F'', S(F''))$ the fixed arcs associated to the new route of $T_i$, thus obtaining the route-set $F'$, and the associated pairs of alternative arcs $A(T_i)$. The resulting alternative graph can be indicated as $(F', S(F''), A(T_i))$. A new solution is then computed on the resulting graph by using the greedy algorithm AMCC described in [22], which selects one alternative arc at a time. The idea is to forbid at each iteration the alternative arc which would introduce the largest consecutive delay in the current selection, thus selecting its paired alternative arc. In Section 4, we refer to this algorithm as UB.

A near-optimal solution to the CDRFR problem can be computed with the branch and bound algorithm described in [12], truncating its execution after a given time limit, that in our experiments is fixed equal to 10 seconds. At each node of the enumeration tree, the branch and bound algorithm uses the single machine Jackson’s Preemptive Schedule of [7] for computing a lower bound and possibly pruning the enumeration tree, as well as static and dynamic implication rules to speed up the computation [12]. In what follows, we refer to this algorithm as OPT.

### 3.2 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 CDRFR problem and a node $i \in N(F) \setminus \{0, n\}$, $\sigma^{-1}(i)$ denotes 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 $l^S(0, i) > l^S(0, \sigma^{-1}(i)) + w_{\sigma^{-1}(i)}$. For each waiting node $i$, there is always a node $h_i$ in $G(S)$, different from $\sigma^{-1}(i)$, such that $l^S(0, i) = l^S(0, h_i) + w_{h_i}$. We therefore call $h_i$ the hindering node of $i$. Notice that for each waiting node $i \in N(F) \setminus \{0, n\}$ there is exactly one hindering node.

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

For example, in the graph of Figure 3:

$C(F, S) = \{0, B7, B8, B9, B10, B5, B6, Bout, C6, Cout, n\};$

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

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

In the graph of Figure 4:

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

In this paper, we study the three neighborhood structures listed below. The two latter neighborhoods can be viewed as restricted versions of the first one.

\( N_C \). The complete neighborhood contains all the feasible solutions to the CDR problem in which one train follows a different route compared to the incumbent solution. This is the largest neighborhood we consider. To limit the number of neighbors to be evaluated, \( N_C \) is only partially explored as follows. A move is obtained by choosing a non-tabu train and a route different from the current one at random (i.e., all trains and alternative routes having the same probability), until a number \( \psi \) of alternative route-sets is obtained, where \( \psi \) is a parameter of the tabu search algorithm.

\( N_{BRCP} \). The backward ramified critical path neighborhood considers only the operations in \( B(F, S) \) and the associated trains. 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 \( B(F, S) \) and then rescheduling train movements. This neighborhood structure has been studied in [11] within a local search scheme.

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

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 \( \mathcal{N} \) 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 \( \mathcal{N} \). In our case, \( \mathcal{N} \) is explored by first defining a new route-set \( F \) and then computing \( S(F) \). We must therefore take into account this assumption when studying the different neighborhood structures.

We next show that \( N_C \) 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. We observe that 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-connectedness is not guaranteed.
An example of this situation is shown in Figure 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. 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.

Figure 5: A small example in which $\mathcal{N}_C$ is not opt-connected

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. To show this fact, consider the example described in Section 2. Analyzing the network in Figure 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 Figure 3 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. In other words, starting from the solution in Figure 3, no move is allowed in $\mathcal{N}_{BRCP}$ or $\mathcal{N}_{FBRCP}$, and the optimal solution in Figure 4 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 re-routed. The former assumptions imply that there is always a non-tabu move in the complete neighborhood $\mathcal{N}_C$.

For each route-set $F$, an optimal or near-optimal schedule $S(F)$ has therefore to be computed in order to avoid missing the optimal CDR solution, i.e., choosing the route set $F^*$. For this reason, independently from the neighborhood structure and move evaluation strategy adopted, whenever a route-set $F$ is changed the algorithm produces a near-optimal solution to the CDRFR problem by using the truncated branch and bound method. The computational experiments of Section 4 will show that this time limit is rarely reached. This is therefore the solution we start from in the next iteration.

In order to achieve the opt-connected property, we implement the Complete neighborhood search strategy, that explores the complete neighborhood $\mathcal{N}_C$, and the four restricted neighborhood search strategies described below.

**Restart.** The best candidate solution is searched in $\mathcal{N}_{FBRCP}$ unless this is empty. Otherwise, $\gamma \geq 1$ consecutive moves are performed in $\mathcal{N}_C$ before searching again in the restricted neighborhood, where $\gamma$ is a parameter of the tabu search algorithm. A new schedule is computed with the OPT algorithm after all $\gamma$ routes are changed.

**Hybrid1.** When $\mathcal{N}_{FBRCP}$ is empty, $\gamma > 1$ moves are performed in $\mathcal{N}_C$ before searching again in $\mathcal{N}_{FBRCP}$. The move is chosen by evaluating $\psi$ candidate solutions in $\mathcal{N}_C$ and selecting the best one with the two objective functions in lexicographic order.
Hybrid2. When $\mathcal{N}_{FBRCP}$ is empty, $\gamma > 1$ moves are performed in $\mathcal{N}_C$. Each move is carried out by evaluating $\psi$ candidate solutions in $\mathcal{N}_C$ and selecting the one with the smallest average delay. Each solution is obtained by first rerouting a single, randomly chosen, non-tabu train and then computing a near-optimal schedule with the OPT algorithm of Section 3.1.

Hybrid3. When a local minima is reached in the restricted neighborhood, i.e., all moves in $\mathcal{N}_{FBRCP}$ are non-improving, $\psi$ candidate solutions are evaluated in $\mathcal{N}_C$ 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. Each solution in $\mathcal{N}_{FBRCP} \cup \mathcal{N}_C$ is obtained by first rerouting a single, randomly chosen, non-tabu train and then computing a near-optimal schedule with the OPT algorithm of Section 3.1.

The four restricted strategies are combination of $\mathcal{N}_C$ and $\mathcal{N}_{FBRCP}$. Moreover, strategy Restart is a simple restart strategy, i.e., the $\gamma$ moves are chosen at random without evaluating their quality. It can be observed that this is a commonly used diversification strategy in tabu search algorithms. Among the hybrid strategies, Hybrid1 and Hybrid3 minimize the two objective functions in lexicographic order. The choice of a different objective function for strategy Hybrid2 is motivated by two observations. First, changing the objective function improves diversification, which is often the aim of the 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 (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. [26].

### 3.3 The tabu search scheme

The tabu search algorithm for the CDR problem with strategy Hybrid2 is described in Figure 6. 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 $\psi$ elements ($\psi$ rerouting options). We denote with $\sum \max\{0, l^{S(F)}(0, n)\}$ the sum of the consecutive delays of all running trains at their relevant points, i.e., the sum of the values $l^{S(F)}(0, i) + w_i$ for all arcs entering node $n$. Each solution is evaluated in $\mathcal{N}_{FBRCP}(IncSol, \psi)$ by considering the maximum and average consecutive delays, in lexicographic order, while each solution is evaluated in $\mathcal{N}_C(IncSol, \psi)$ by only considering the average consecutive delay. The value of a solution is therefore the pair $[l^{S(F)}(0, n); \sum \max\{l^{S(F)}(0, n)\}]$.

The search process is halted when a time limit (bounding the computational effort) is exceeded. 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 applies strategy Hybrid2 for $\gamma$ iterations. The value of a neighbor $(F', S(F')) \in \mathcal{N}_{FBRCP}(IncSol, \psi)$ is computed by using one of the three strategies described in Section 3.1. Specifically, when the Jackson Preemptive Schedule lower bound is used, the primary objective function is
Algorithm TabuSearch
Input:
an initial solution \((F, S(F))\)
begin
\(IncSol = (F, S(F))\)
\(BestSol = (F, S(F))\)
\(BestValue = [l^{S(F)}(0,n); \sum \max \{l^{S(F)}(0,n)\}]\)
while time limit is not reached do
begin
if \(N_{FBRCP}(IncSol, \psi) = \emptyset\) then (explore the neighborhood \(N_C\))
begin
starting from \(IncSol\), execute \(\gamma\) moves in \(N_C\) and generate \((F'', S(F''))\)
\(IncSol = (F'', S(F''))\)
\(NextValue = [l^{S(F'')}\(0,n); \sum \max \{l^{S(F'')}\(0,n)\}]\)
update TL
end
else (execute a move in the neighborhood \(N_{FBRCP}\))
begin
\(NextValue = [+\infty; +\infty]\)
while \(N_{FBRCP}(IncSol, \psi) \neq \emptyset\) do
begin
choose a solution \((F', S(F'))\) \(\in N_{FBRCP}(IncSol, \psi)\)
\(N_{FBRCP}(IncSol, \psi) = N_{FBRCP}(IncSol, \psi) \setminus (F', S(F'))\)
if \([l^{S(F')}(0,n); \sum \max \{l^{S(F')}(0,n)\}] < NextValue\) then
begin
\(NextSol = (F', S(F'))\)
\(Nextvalue = [l^{S(F')}(0,n); \sum \max \{l^{S(F')}(0,n)\}]\)
end
end
insert in TL the train having different route in \(IncSol\) and \(NextSol\)
\(IncSol = NextSol\)
end
if \(NextValue < BestValue\) then
begin
\(BestSol = IncSol\)
\(BestValue = NextValue\)
end
end
end

Figure 6: Pseudocode of the tabu search algorithm using strategy Hybrid2
evaluated and 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 truncated branch and bound algorithm. The best solution in $N_{FBRCP}(IncSol, \psi)$ is the new incumbent solution, which replaces, possibly, the current optimal solution. The train having a different route from the old to the new incumbent solution is added to the tabu list ($TL$) for $\lambda$ iterations, where $\lambda$ is the length of the tabu list. A tabu-train is not allowed to be rerouted, even if its rerouting would lead to an unexplored solution. In our experiments, we found beneficial using a small tabu list and forbidding all the routes associated to a tabu train. In the next section, we will discuss in detail the effects of this choice against the alternative choice of inserting a single train route in the tabu list.

4 Computational experiments

This section presents our computational experiments on the dispatching area of Utrecht Den Bosch, a bottleneck of the Dutch railway network. Section 4.1 describes the test case while Section 4.2 reports the advantages of using alternative algorithmic components in our tabu search, namely the aspiration criterion, the tabu list structure and the method for neighbors evaluation. Finally, Section 4.3 presents the performance of our neighborhood search strategies, by studying complex perturbations (i.e., several trains delayed at their entrance in the network) and disruptions (i.e., some tracks are blocked and cannot be used). We also consider timetables with and without passenger connections. In all the experiments, the parameters ($\psi, \lambda, \gamma$) are fixed equal to (8, 3, 5) respectively. These values were defined by running a preliminary set of experiments with 18 pilot instances and for several triples ($\psi, \lambda, \gamma$). The above values were those achieving the best results on average for all the algorithmic components addressed in this section. 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.

4.1 Test case description

The Utrecht Den Bosch railway infrastructure is shown in Figure 7. 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 Figure 7). Overall, 356 possible train routes are considered.

We evaluate a timetable variant for year 2007, hourly, cyclic and extended over the entire railway area. During a 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 one hour 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 seconds and have an average entrance delay of around 320 seconds. 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 minutes of the timetable while the time period of traffic prediction used in the experiments is one hour long. We generate 24 instances with passenger and rolling stock connection constraints plus 24 instances with rolling stock connection constraints but without passenger connection constraints.

Table 1: Description of the three disruptions

<table>
<thead>
<tr>
<th>Disruption</th>
<th>Unavailable Block Sections</th>
<th>%Unavailable Routes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>67 145 167</td>
<td>65.2</td>
</tr>
<tr>
<td>2</td>
<td>168 164 67 175</td>
<td>66.6</td>
</tr>
<tr>
<td>3</td>
<td>67</td>
<td>40.4</td>
</tr>
</tbody>
</table>

Table 1 presents three configurations of randomly generated blocked tracks (corresponding to the most time consuming disruptions studied in [11, 10]). Each disruption (column 1) is obtained by making unavailable a set of block sections (column 2). Column 3 shows the percentage of routes made unavailable by the blocked tracks. The block sections with problems are also reported in Figure 7.

4.2 Assessment of tabu search components

This section compares different alternative choices to configure our tabu search. Specifically, we analyze (i) the implementation of an aspiration criterion, (ii) the tabu list structure, (iii) the algorithm for choosing the best candidate in the neighborhood. Each comparison is based on strategies Restart and Complete.
As for the first point, we only explore two possibilities, i.e., using or not using this feature. We consider only a simple aspiration criterion, in which tabu moves are explored and implemented when leading to an improvement of the current best solution. The results are compared in Figure 8 to the case without aspiration criterion. The value of the best solution found at time \( t \) is shown for the two cases, for \( t = 0, \ldots, 180 \) seconds. We conclude that the computational cost of the aspiration criterion is not compensated by significant improvements to the best solution. This criterion is therefore not used in the following experiments.

![Figure 8: Comparison between using or not using an aspiration criterion](image)

As for the second point, we explore the two possibilities of inserting a train in the tabu list or inserting a train route in the tabu list, the former case being more restrictive. The results of this comparison are shown in Figure 9. The value of the best solution found at time \( t \) for the two cases shows, clearly, that the former choice is preferable. This is thus the option used in the following experiments.

As for the third point, we evaluate three algorithms for choosing the best candidate in the neighborhood of the incumbent solution. These algorithms are denoted as LB, UB and OPT (as in Section 3.1). Figure 10 shows the values of the best solution found at time \( t = 0, \ldots, 180 \) seconds for the three cases. Specifically, LB is a fast but not accurate evaluation criterion while UB is more fast and accurate, even if this latter sometimes fails in finding a feasible solution. In case of infeasibility, the neighbor is discarded from the neighborhood. From the obtained results, OPT produces the most accurate estimates but, being the most time consuming method, causes the tabu search being approximately three times slower than when using algorithm UB. The described behavior motivates our choice of using UB for the purpose of neighbors evaluation.
Figure 9: Comparison between two tabu list structures

Figure 10: Assessment of different methods for neighbors evaluation
4.3 Disturbance management

In this section, we evaluate the neighborhood search strategies described in Section 3.2. We analyze separately the management of 48 perturbations with multiple late trains and the joint effects of these 48 perturbations and the three disruptions described in Section 4.1. Except for the search strategy, the configuration of the tabu search algorithm is the best performing in the set of experiments of Section 4.2.

Table 2: Comparison of tabu search neighborhood search strategies

<table>
<thead>
<tr>
<th>Neighborhood Search Strategy</th>
<th>Timetable Perturbations</th>
<th>Timetable Disruptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Improving Moves</td>
<td>Changed Routes</td>
</tr>
<tr>
<td>Complete</td>
<td>6.9</td>
<td>14.5</td>
</tr>
<tr>
<td>Restart</td>
<td>5.1</td>
<td>15.1</td>
</tr>
<tr>
<td>Hybrid1</td>
<td>6.3</td>
<td>13.7</td>
</tr>
<tr>
<td>Hybrid2</td>
<td>7.1</td>
<td>12.7</td>
</tr>
<tr>
<td>Hybrid3</td>
<td>7.1</td>
<td>12.7</td>
</tr>
</tbody>
</table>

Table 2 presents the general behavior of different tabu search strategies. 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 original (default) routes prescribed in the timetable. In case of disruptions, if a default route is not available this is initially replaced with its shortest available alternative route. It is interesting to observe that the best solution is improved more frequently in the case of strategies Hybrid2 and Hybrid3. Their final solutions also present a smaller number of changed routes compared to the default routes, and should be therefore easier to implement by traffic controllers during rail operations.

Column 4 reports the number of time limits reached by OPT which is truncated after 10 seconds of computation. It can be noted that when dealing with timetable perturbations this time limit is only rarely reached. On the other hand, the instances with disruptions are significantly harder, as also indicated in the last column of the table for algorithm OPT. 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 particularly hard to solve to optimality.

Columns 5 and 6 report the average number of moves executed in the neighborhoods $N_C$ and $N_{FBRCP}$, respectively. It has to be observed that we consider under strategy Hybrid3 and $N_{FBRCP}$ only those moves that do no require searching in $N_C$. The higher number of moves carried out by strategy Restart is due to the fact that searching in $N_C$ is particularly fast with this strategy and leaves more time to search in $N_{FBRCP}$. The last column shows the average time spent by the tabu search configurations to run algorithm
OPT within 180 seconds of computation. Quite surprisingly, this time is shorter with strategy Complete, despite the higher number of OPT executions. Therefore, when rerouting a train with this strategy, the resulting instances of the CDRFR problem are easier to solve compared to the other strategies. On the other hand, the other four strategies require similar amount of time to run algorithm OPT.

We now compare the results obtained by our tabu search algorithm, for the five search strategies, with the performance of the local search algorithm of [11], based on the $\mathcal{N}_{BRCP}$ neighborhood, and of the proven optimal CDRFR solutions with default routes.

Figures 11 and 12 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 with default routes. The average solutions of the other rerouting algorithms are depicted each 10 seconds of computation, up to three minutes.

The latter figures present, clearly, the benefits of rescheduling and rerouting trains to recover delays. Also evident is the delay reduction when passing from the optimal CDRFR solutions to the achieved CDR solutions. The drop is more than one third for maximum consecutive delay and more than one half for the average consecutive delay.

The hybrid strategies give the best overall results, outperforming strategies Restart and Complete, thus demonstrating the advantage of using a hybrid search. However, the delay reduction has a price in terms of computation time. For the perturbation instances, the proven optima to the CDRFR problem are obtained, on average, within two seconds of computation, while the tabu search algorithm needs at least 10 seconds for achieving a 30% reduction of maximum and average consecutive delays. The local search algorithm needs more than 20 seconds to achieve similar results. Moreover, the strategies Hybrid2 and Hybrid3 achieve a 40% reduction of both delays after 20 seconds of computation, and the strategy Hybrid2 needs 40 seconds to halve the average consecutive delay.

The disruption instances are significantly more time consuming compared to the perturbations instances, despite the smaller number of rerouting options. Specifically, the local search improves slowly the initial solution during the 180 seconds of computation, while the strategies Hybrid1 and Hybrid2 attain 10% reduction of the maximum average delay only after 40 seconds. Furthermore, the strategies Hybrid2 and Hybrid3 need around 90 seconds to obtain a 20% average consecutive delay reduction. However, it should be noticed that the human dispatchers need more support to manage severe disruptions. A restricted line capacity also limits the delay reduction that can be achieved.

To summarize, our tabu search algorithm, with hybrid search strategies, is able to provide an effective support for the delay management, in case of timetable perturbations and disruptions, within a time limit compatible with rail operations.
Figure 11: Comparison of the six algorithms in case of timetable perturbations
Figure 12: Comparison of the six algorithms in case of timetable disruptions
5 Conclusions

This paper investigates the effectiveness of advanced strategies to solve the compound train rerouting and rescheduling problem. The computational results, carried on real-world railway instances, demonstrate the high potential of rerouting trains in order to minimize consecutive delays. Novel tabu search neighborhood structures and search strategies have been studied in detail. Focused neighborhoods and local search yield promising results within short computation time when dealing with perturbations, while more sophisticated hybrid strategies are required to effectively manage disruptions in real-time. Furthermore, hybrid strategies outperform the local search and the stand-alone neighborhoods.

Further research is required on a number of issues. From the computational point of view, the development of CDR algorithms, able to find better solutions within shorter computation time, is worthwhile, as well as the development of exact algorithms and effective lower bounds to the CDR problem, which would allow to further improve the quality of the solutions found by the tabu search algorithm. 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.

Acknowledgments

We thank ProRail (The Netherlands) for providing the instances. This work is partially supported by the programs “Towards Reliable Mobility” of the Transport Research Centre Delft, the Dutch foundation “Next Generation Infrastructures” and ProRail.

References


