Computers & Operations Research 78 (2017) 480–499

Contents lists available at ScienceDirect

Computers & Operations Research

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

A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations

Marcella Samà [a], Andrea D'Ariano [a,*], Francesco Corman [b], Dario Pacciarelli [a]

[a] Roma Tre University, Department of Engineering, Section of Computer Science and Automation, via della Vasca Navale, 79 – 00146 Rome, Italy
[b] Delft University of Technology, Department of Maritime and Transport Technology, Section of Transport Engineering and Logistics, Mekelweg, 2 – 2628CD Delft, The Netherlands

A R T I C L E I N F O

Available online 23 February 2016

Keywords:
Real-time railway traffic management
Train scheduling and routing
Alternative graph
Disjunctive programming
Branch-and-bound
Variable neighbourhood search
Tabu search

A B S T R A C T

This paper focuses on the development of metaheuristic algorithms for the real-time traffic management problem of scheduling and routing trains in complex and busy railway networks. This key optimization problem can be formulated as a mixed integer linear program. However, since the problem is strongly NP-hard, heuristic algorithms are typically adopted in practice to compute good quality solutions in a short computation time. This paper presents a number of algorithmic improvements implemented in the AGLIBRARY optimization solver in order to improve the possibility of finding good quality solutions quickly. The optimization solver manages trains at the microscopic level of block sections and at a precision of seconds. The solver outcome is a detailed conflict-free train schedule, being able to avoid deadlock situations and to minimize train delays. The proposed algorithmic framework starts from a good initial solution for the train scheduling problem with fixed routes, obtained via a truncated branch-and-bound algorithm. Variable neighbourhood search or tabu search algorithms are then applied to improve the solution by re-routing some trains. The neighbourhood of a solution is characterized by the set of candidate trains to be re-routed and the available routes. Computational experiments are performed on railway networks from different countries and various sources of disturbance. The new algorithms often outperform a state-of-the-art tabu search algorithm and a commercial solver in terms of reduced computation times and/or train delays.

1. Introduction

In the last years, European railway companies are experiencing increasing difficulties to face the ever increasing transport demand while ensuring good quality of service to passengers, also due to the limited space and funds to build new infrastructure in bottleneck areas. These facts stimulated the interest for new effective Operations Research (OR) solutions for real-time train scheduling. This problem is faced by dispatchers, which have to modify orders, passing times and routes of trains (on-line train dispatching problem) in order to counter delays and keep traffic smooth.

In the train scheduling literature, there is a well-known difference between the level of sophistication of the theoretical results and algorithms and that of the methods that are employed in practice. While the theory typically address simplified problems, achieving optimal or near-optimal performance, the practice must face all the complexity of real-time operations, often with little attention to the performance level. This difference is especially evident for real-time scheduling, and train scheduling is not an exception. As a result, the poorly performing scheduling methods that are used in practice has a direct impact on the quality of service offered to the passengers, and the negative effects of disruptions on the regularity of railway traffic may last for hours after the end of the disruption (Kecman et al. [32]). However, there are recently many signals that the scheduling gap could be drastically reduced in the next few years. On the theoretical side, recent approaches to train scheduling tend to incorporate an increasing level of detail and realism in the models while keeping the computation time of the algorithms at an acceptable level. On the practical side, the railway industry is interested in assessing the suitability of these methods to the practical needs of real-time railway traffic management.

The design and implementation of advanced mathematical models is a prerequisite to the development of innovative decision support systems for solving the on-line train dispatching problem. This paper is concerned with the modelling of the conflict detection and resolution (CDR) problem for railway networks. The CDR

* Corresponding author.
E-mail address: dariano@ing.uniroma3.it (A. D'Ariano).

http://dx.doi.org/10.1016/j.cor.2016.02.008
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

problem is the real-time problem of computing a conflict-free and deadlock-free schedule compatible with the actual status of the network and such that the circulating trains arrive and depart with the smallest possible delay. To solve the CDR problem, a number of algorithmic improvements are implemented in the AGLIBRARY solver, a set of OR-based models and algorithms for complex practical scheduling problems developed at Roma Tre University. This solver is the main solution engine of the ROMA dispatching support system [22], used for instance in the EU project ON-TIME [20]. The solver is based on the alternative graph model introduced by Mascis and Pacciarelli [38] and on the following framework: a good initial solution for the scheduling problem with fixed routes is computed by the (truncated) branch-and-bound algorithm in [25]. Metaheuristics are then applied to improve the solution by re-routing some trains. This action corresponds to the concept of a move, from a metaheuristics perspective. In [13], a tabu search algorithm has been applied to solve practical-size railway instances for a Dutch test case in the Netherlands.

Previous research left open a number of relevant algorithmic issues. The first issue concerns the extent at which different search strategies and alternative solution methods might outperform the tabu search algorithm. A second issue is to quantify the algorithmic improvements, looking at a reduction of the computation time and at an improvement of solution quality. Both these issues motivate the development of the new metaheuristics proposed in this paper. The paper contributions are next outlined:

- We present routing neighbourhoods that differ from each other for the set of candidate trains to be re-routed in each move and for the available routing alternatives.
- We alternate the search for promising moves via systematic changes of a combination of neighbourhood structures, similarly to Moreno Pérez et al. [41], and present strategies for searching within these neighbourhoods based on variable neighbourhood search schemes [30].
- We use fast train scheduling heuristics for the evaluation of each neighbour.
- We apply the proposed algorithms to the management of complex CDR problems, characterized by busy traffic, multiple delayed trains and temporarily disrupted railway resources. The new metaheuristics are compared with a state-of-the-art tabu search algorithm [13] and with a commercial solver. Significantly better results are obtained in terms of a reduced time to compute the best-known (sometimes proven optimal) solutions, and for some CDR instances also in terms of an improved solution quality.
- We evaluate the algorithms over various real-world test cases, which feature different railway network characteristics and traffic flows.

Section 2 gives an overview of the literature related to the real-time railway traffic management. Section 3 formally defines the CDR problem and Section 4 presents mathematical formulations for this problem. Section 5 describes the algorithms of AGLIBRARY and the new metaheuristics proposed in this paper. Section 6 reports on the performance of the algorithms on various practical case studies from Italy, the Netherlands and UK. Section 7 summarizes the main paper findings and outlines future research directions. An appendix illustrates the neighbourhoods investigated in this work with a numerical example.

2. Literature review

The study of real-time train scheduling and routing problems received increasing attention in the literature in the last years. Early approaches (starting from the pioneering work of [50]) tend to solve very simplified problems that ignore the constraints of railway signalling, and that are only applicable for specific traffic situations or network configurations (e.g. a single line or a single junction), see the literature reviews in the following papers: Ahuja et al. [1]; Cacchiani et al. [5]; Cordeau et al. [10]; Fang et al. [29]; Hansen and Pachl [31]; Lusby et al. [36]; Meng and Zhou [40]; Pellegrini and Rodriguez [44]; Pellegrini et al. [43]; Törnquist and Persson [51]. Among the reasons for this gap between early theoretical works and practical needs are the inherent complexity of the real-time process and the strict time limits for taking and implementing decisions, which leave small margins to a computerized Decision Support System (DSS).

Effective DSSs must be able to provide the dispatcher with a conflict-free disposition schedule, which assigns a travel path and a start time to each train movement inside the considered time horizon and, additionally, minimizes the delays (and possibly the main broken connections) that could occur in the network. The main pre-requisite of a good DSS is the real-time ability to deal with actual traffic conditions and safety rules for practical networks. In other words, the solution provided by a DSS must be feasible in practice, since the human dispatcher may have not enough time to check and eventually adjust the schedule suggested by the DSS. A recognized approach to represent the feasibility of a railway schedule is provided by the blocking time theory, acknowledged as standard capacity estimation method by UIC in 2004 (Hansen and Pachl [31]), which represents a safe sequence of train movements in the railway network with the so-called blocking time stairways.

With the blocking time theory approach, the schedule of a train is individually feasible if a blocking time stairway is provided for it, starting from its current position and leaving each station (or each other relevant point in the network) not before the departure time prescribed by the timetable. A set of individually feasible blocking time stairways (one for each train) is globally feasible if no two blocking time stairways overlap. The timetable prescribes the set of trains that are expected to travel in the network within a certain time window, the stops for each train and a pair of (arrival, departure) times for each train and each stop. At other relevant points (e.g. at the exit from the network or specific relevant points between two consecutive stations) can be defined minimum and/or maximum pass through times.

Many models and algorithms for train re-scheduling have already been proposed in the literature, but only a few of them with successful application in practice. So far, the most successful attempt in the literature to incorporate the blocking time theory in an optimization model is based on the alternative graph model introduced by Mascis and Pacciarelli [38]. This model is a generalization of the disjunctive graph for job shop scheduling, in which each operation denotes the traversal of a resource of the network by a job (train). Effective applications to real-time train scheduling are described in D'Ariano et al. [25], Mannino and Mascis [37], Mazzarello and Ottaviani [39]. However, other promising approaches have been provided in the literature, either based on mathematical formulations (Cadarso and Marín [3]; Caimi et al. [7]; Lamorgese and Mannino [34]; Pellegrini et al. [43]; Rodriguez [46]; Şahin [47]; Törnquist and Persson [51]; Wegele et al. [54]) or on algorithmic approaches (Almodovar et al. [2]; Cai and Goh [6]; Cheng [8]; Chiu et al. [9]; Liu and Kozan [35]; Törnquist Krasemann [52]; Wegele and Schnieder [53]). Another important aspect when dealing with rail operations is the passenger behaviour (Cadarso et al. [4]; Corman et al. [19]; Dollevoet et al. [28]; Kroon et al. [33]), even if this latter aspect is not considered explicitly in this paper.

The alternative graph model allows to directly model the individual and global train schedule feasibility concepts expressed by the blocking time theory. This enables the detailed recognition of timetable conflicts in a general railway network with mixed traffic for a given look-ahead horizon, even in presence of heavy disturbances and network disruptions. Several later studies have confirmed the ability of the model to take into account different practical needs, such as train priorities (Corman et al. [11]), energy consumption issues (Corman et al. [12]), passenger and rolling stock transfer connections (Corman et al. [15]; D'Ariano et al. [23]), train re-routing (Corman et al. [13]; D'Ariano et al. [23]), management of complex and busy stations (Corman et al. [17,18]), traffic coordination between dispatching areas (Corman et al. [14,16]). Clearly, an alternative graph model of the CDR problem can be easily translated into a mixed integer program, and then solved with a commercial or academic software. However, the CDR problem is inherently strongly NP-hard, so it is often not possible to compute a feasible solution quickly via any commercial software for practical-size CDR instances. Such NP-hard problems are typically solved via heuristic algorithms that enable the computation of good quality solutions in a computation time compatible with real-time operations.

This paper presents a number of algorithmic improvements implemented in the AGLIBRARY optimization solver in order to improve the possibility of finding good quality solutions quickly. A set of specialized algorithms based on the alternative graph model is included in AGLIBRARY. This re-scheduling system includes solution algorithms ranging from fast heuristic procedures that can be chosen by the user to sophisticated branch-and-bound algorithms for train scheduling [25] or metaheuristics for train re-routing [13,16,23]. AGLIBRARY has been tested on various railway networks managed by the Dutch infrastructure manager ProRail (the railway networks Leiden–Schiphol–Amsterdam; Utrecht–Den Bosch; Utrecht–Den Bosch–Nijmegen–Arnhem), by the British infrastructure manager NetworkRail (part of the east coast main-line nearby London) and by the Italian infrastructure manager RFI (the regional line Campoleone–Nettuno), even if in principle the software can tackle any national or international traffic management system standard. AGLIBRARY has also been used in other application contexts, including steelmaking-continuous casting production scheduling [42], real-time air traffic scheduling and routing at a terminal control area [24,26,48,49], real-time management of containers at a container terminal [21,55].

The new AGLIBRARY algorithms proposed in this paper can be potentially applied to improve the results obtained both in railway traffic management and in the other application contexts. However, the algorithmic structures and parameters would need to be customized for each particular railway network and test case from other application fields. This work is focused on the customization and application of the new algorithms to solve the CDR problem in various railway networks, including issues related to the management of complex station areas, connection constraints, train re-scheduling and re-routing variables. The new algorithms are compared with previously-published algorithms and with a commercial solver.

3. Problem definition

Signals, interlocking and Automatic Train Protection (ATP) systems control the train traffic by imposing safety constraints between trains, setting up train routes and enforcing speed restrictions on running trains. Fixed block ATP systems ensure safety through the concept of *block section*, a part of the infrastructure that is exclusively assigned to at most one train at a time. Train movements can be modelled by a set of characteristic times, as follows. The *running time* of a train on a block section starts when its head (the first axle) enters the block section and ends when the head of the train reaches the end of the block section.

Safety regulations impose a minimum separation between consecutive trains traveling on the same block section, which translates into a minimum *headway time* between the start of the running times of two consecutive trains on the same block section. This time depends on the length of the block section, as well as on other factors like the speed and length of the trains and includes the time between the entrance of the train head in a block section and the exit of its tail (the last axle) from the previous one, plus additional time margins to release the occupied block section and to take into account the sighting distance.

Proactive re-scheduling of railway traffic must take into account several facts. The network is composed of block sections and platform stops at stations. A train is not allowed to depart from a platform stop before its scheduled departure time and is considered late if arriving at the platform later than its scheduled arrival time. At a platform stop, the scheduled stopping time of each train is called *dwell time*.

Disturbances affect rail traffic. We can distinguish between light traffic *perturbations* from neighbouring dispatching areas and heavy traffic *disruptions*. The former are light disturbances caused by a set of delayed trains in a dispatching area, while the latter are much stronger disturbances of the scheduled times and routes (e.g. due to some block sections being unavailable for a certain amount of time). Other kinds of disturbances include extensions to dwell times due to passengers boarding, connection constraints, or technical problems; and running time prolongation because of headway conflicts between trains or technical failures.

Moreover, the railway infrastructure is increasingly becoming utilised, towards a saturation level. This results in a strong sensitivity to initial delays, which are due to breakdowns, failures, extended dwell time at stations due to passengers; those phenomena are almost unavoidable. In saturated networks, those initial delays are particularly hard to be managed, and easily generate knock-on (or consecutive) delays which spread over the network in time and space, affecting more trains.

Delays propagate between trains when solving potential conflicting routes. Namely, a *potential conflict* between two trains arises if the trains request a same block section within a time interval smaller than the minimum time headway between them, which is needed for safety reasons and smooth running. The solution of the potential conflict is to fix the order of trains over the block section; in that case, one of the approaching trains might be forced to decelerate and thus experiencing a knock-on delay. Unscheduled braking and stopping of trains increases the running time and may cause an additional delay. Similarly, trains can be held at stations due to unavailability of outbound routes, or conversely prevented to enter stations as far as platforms and inbound routes are not available. In general, delays may propagate to other trains causing a domino effect of increasing traffic disturbances.

The conflict detection and resolution (CDR) problem studied in this paper can be defined as follows: given a railway network, a set of train routes and passing/stopping times at each relevant point in the network, and the position and speed of each train being known at a given starting time t_0 of traffic prediction, find an optimized plan of operations that solves all potential conflicts between trains, does not result in deadlock situations (i.e. a set of trains that are circularly waiting for each other, making any planned movement impossible), it is compatible with initial positions of trains, and such that the selected train timing, ordering and routing decisions are feasible, no train appears in the network before its expected entrance time (including the initial delays), no train departs from a relevant point before its scheduled departure time, and trains arrive at the relevant points with the smallest possible consecutive delay.

4. Problem formulation

The CDR problem is characterized by train routing and scheduling decisions. The general problem can thus be divided into two sub-problems: (i) the selection of a route for each train, and (ii) the scheduling decisions once the routes have been fixed for each train. This section describes the problem decomposition in scheduling and routing variables. The general alternative graph model is given for the CDR problem with fixed routes. A Mixed Integer Linear Programming (MILP) formulation is proposed for the alternative graph model. An extended MILP formulation is then given for the overall CDR problem. Binary variables are introduced both for the train scheduling and routing decisions. An illustrative example of the alternative graph model is reported in the appendix of this paper.

4.1. Alternative graph model

The alternative graph (AG) generalizes the classical disjunctive graph in order to take into account constraints arising in real-world scheduling applications. Regarding the CDR problem, AG allows to easily and efficiently check the feasibility of a train scheduling solution (all operations are to be processed with no deadlock and conflict situations), as well as the quality of a train scheduling solution (the maximum consecutive delay collected at each relevant location). The CDR problem is based on fixed and alternative constraints.

Fixed constraints model the individual feasibility of a train schedule, i.e. the blocking time stairway. A timing variable is associated to the entrance of each train in each resource (block section, platform of station route). The schedule is individually feasible if the entrance in a resource is at least a sufficient amount of time after the entrance in the former resource, respecting all the safety and operational constraints. Assuming that the route of a train has been fixed, if at the current time the train occupies a certain resource, it cannot enter the next resource in its route before the time needed to traverse the remaining part of the current resource. Since the timetable prescribes a departure (or a pass through) time for the train at each relevant point in its route, the train cannot enter the next resource of the route before a minimum prescribed time.

Alternative constraints model the global feasibility of a set of blocking time stairways (one for each circulating train). Given a resource traversed by two trains, the second train cannot enter the resource before the entrance time of the previous train plus its blocking time, i.e. the time interval in which the resource is reserved for the first train. If a precedence constraint has not been fixed between the two trains on that resource (either by the timetable, or the dispatcher, or the physical network topology), then two orderings are possible and one of them has to be chosen in a train scheduling solution. This fact is represented in the alternative graph by a pair of alternative constraints, one of which must be chosen in a solution.

The AG formulation of the CDR problem with fixed routes (i.e. in which the route is prescribed and cannot be changed) is a triple $G=(N, F, A)$ where $N=\{0, 1, \ldots, n-1, n\}$ is a set of $n+1$ nodes, F is a set of *fixed* directed arcs and A a set of pairs of *alternative* directed arcs.

Each node, except the start 0 and end n nodes, is associated with the start of an operation krj, where k indicates the train, r the route chosen and j the resource it traverses. The start time t_{krj} of operation krj is the entrance time of train k with route r in resource j.

The fixed arcs are used to model running, dwell, connection, arrival, departure, and pass through times of trains. Let the resources p and j be two consecutive resources processed by train k with route r, the fixed arc $(krp, krj) \in F$ models a job constraint between the nodes krp and krj. The weight $w^F_{krp\_krj}$ represents a minimum time constraint between t_{krp} and t_{krj}: $t_{krj}-t_{krp} \geq w^F_{krp\_krj}$. A fixed arc $(umv, krz) \in F$ is used to enforce a connection constraint between train k with route r and train u with route m, i.e. $t_{krz}-t_{umv} \geq w^F_{umv\_krz}$.

The alternative arcs are used to model the headway times between two consecutive trains. Each pair of alternative arcs $((krj, ump), (umi, krp)) \in A$ models train ordering decisions between train k with route r and train u with route m on resource p. Note that j [respectively i] is the next resource processed by train k [u] when using route r [m]. The two arcs of the pair are associated with the weights $w^A_{krj\_ump}$ and $w^A_{umi\_krp}$. In any solution, only one arc of each pair can be selected. If alternative arc (krj,ump) [(umi,krp)] is selected in a solution, the constraint $t_{ump}-t_{krj} \geq w^A_{krj\_ump}$ [$t_{krp}-t_{umi} \geq w^A_{umi\_krp}$] has to be satisfied. This corresponds to fixing the order of trains, first k and then u [first u and then k].

A solution to the CDR problem with fixed routes is represented by the following graph structure. A graph selection S is a set of alternative arcs obtained by selecting exactly one arc from each alternative pair in A and such that the resulting graph $\mathcal{G}(F, S)=(N, F \cup S)$ does not contain positive weight cycles. This allows to associate train orders and times to all operations.

The objective function is the minimization of the maximum consecutive delay, i.e. the largest positive deviation from the scheduled times at relevant locations. In the alternative graph, the maximum consecutive delay minimization is measured as a makespan minimization. Given a selection S and any two nodes krp and uml, we let $l^S(krp, uml)$ be the weight of the longest path from krp to uml in $\mathcal{G}(F, S)$. By definition, the start time t_{krp} of $krp \in N$ is the quantity $l^S(0, krp)$, which implies $t_0=0$ and $t_n=l^S(0, n)$.

To summarize, the alternative graph model corresponds to the following mathematical formulation:

$$\begin{aligned}
\min \quad & t_n \\
\text{s.t.} \quad & t_{krj}-t_{krp} \geq w^F_{krp\_krj} \quad (krp, krj) \in F \\
& t_{krz}-t_{umv} \geq w^F_{umv\_krz} \quad (umv, krz) \in F \\
& (t_{ump}-t_{krj} \geq w^A_{krj\_ump}) \vee (t_{krp}-t_{umi} \geq w^A_{umi\_krp}) \quad ((krj, ump), \\
& (umi, krp)) \in A
\end{aligned} \tag{1}$$

4.2. MILP formulations

A natural mathematical formulation of the CDR problem with fixed routes can be obtained from the alternative graph formulation (1) by translating each alternative pair into a pair of constraints and by introducing a binary variable representing the choice of one of the two constraints. The CDR problem with fixed routes can be viewed as a particular *disjunctive program*:

$$\begin{aligned}
\min \quad & t_n \\
\text{s.t.} \quad & t_{krj}-t_{krp} \geq w^F_{krp\_krj} \quad (krp, krj) \in F \\
& t_{krz}-t_{umv} \geq w^F_{umv\_krz} \quad (umv, krz) \in F \\
& t_{ump}-t_{krj} \geq w^A_{krj\_ump}+Mx_{(krj,ump),(umi,krp)} \quad ((krj, ump), (umi, krp)) \in A \\
& (t_{krp}-t_{umi} \geq w^A_{umi\_krp}+M(1-x_{(krj,ump),(umi,krp)}) \quad ((krj, ump), \\
& (umi, krp)) \in A \\
& x_{(krj,ump),(umi,krp)} \in \{0, 1\}
\end{aligned} \tag{2}$$

The variables are the following: $|N|$ real variables t_{krj} associated to the start time of each operation $krj \in N$, and $|A|$ binary variables $x_{(krj,ump),(umi,krp)}$ associated to each alternative pair $((krj, ump), (umi, krp)) \in A$. The constant M is a sufficiently large number, e.g. the sum of all arc weights.

We model the variables and constraints of the CDR problem for the different routes of each train as follows. The formulation (2) can be extended to the problem with routing flexibility by enlarging sets N, F and A to contain all possible train routes. In addition to the $|N|+|A|$ variables of the CDR problem, $|C|$ binary variables y are associated to the routes of the set of trains considered. The CDR problem with routing flexibility can also be viewed as a particular *disjunctive program*:

$$
\begin{aligned}
\min\ & t_n \\
\text{s.t.}\ & t_{krj}-t_{krp} \geq w^F_{krp\_krj}+M(1-y_{kr}) \quad (krp,krj)\in F \\
& t_{krz}-t_{umv} \geq w^F_{umv\_krz}+M(2-y_{um}-y_{kr}) \quad (umv,krz)\in F \\
& t_{ump}-t_{krj} \geq w^A_{krj\_ump}+M(2-y_{um}-y_{kr})+Mx_{(krj,ump),(umi,krp)} \\
& \quad ((krj,ump),(umi,krp))\in A \\
& t_{krp}-t_{umi} \geq w^A_{umi\_krp}+M(2-y_{um}-y_{kr})+M(1-x_{(krj,ump),(umi,krp)}) \\
& \quad ((krj,ump),(umi,krp))\in A \\
& \sum_{a=1}^{R_b} y_{ab}=1 \quad b=1,\ldots,Z \\
& y_{ab}\in\{0,1\} \\
& x_{(krj,ump),(umi,krp)}\in\{0,1\}
\end{aligned} \tag{3}
$$

In the CDR problem formulation (3), Z is the number of trains, and R_b the number of routes for each train $b=1,\ldots,Z$. The binary variable y_{ab} indicates if route a is chosen (1) or not (0) for train b. For each train b, only a single route, among the R_b routes, can be chosen in any CDR solution. The following constraint holds for train b: $\sum_{a=1}^{R_b} y_{ab}=1$.

When a route r is chosen for train k (i.e. $y_{kr}=1$), each fixed constraint related to route r and train k must be satisfied. For each fixed arc $((krp,krj))\in F$, $t_{krj}-t_{krp}\geq w^F_{krp\_krj}$ must hold. A fixed arc $(umv,krz)\in F$ enforces a connection constraint between train k with route r and train u with route m.

Regarding the alternative constraints, if $y_{um}=y_{kr}=1$ and the routes of trains u and k use the same infrastructure resource p, a potential conflict exists on that resource and an ordering decision has to be taken. This is modelled by introducing the binary variable $x_{(krj,ump),(umi,krp)}$ for the alternative pair $((krj,ump),(umi,krp))\in A$. There are two possible scheduling decisions for each alternative pair $((krj,ump),(umi,krp))\in A$: if $x_{(krj,ump),(umi,krp)}=1$ then $t_{krp}-t_{umi}\geq w^A_{umi\_krp}$ must be satisfied (i.e. $(umi,krp)\in S$); if $x_{(krj,ump),(umi,krp)}=0$ then $t_{ump}-t_{krj}\geq w^A_{krj\_ump}$ must be satisfied (i.e. $(krj,ump)\in S$).

5. Train scheduling and re-routing algorithms

This section describes the algorithmic approaches proposed in this paper to solve the CDR problem. Section 5.1 presents the general framework of the AGLIBRARY solver that is based on a combination of train scheduling and re-routing algorithms. Section 5.2 presents the algorithms used to compute a train schedule for given routes. Section 5.3 describes the neighbourhoods for the search of new train routes starting from a routing and scheduling solution. Section 5.4 is devoted to the scheduling heuristic
procedures used to evaluate the neighbours (the new routing combinations). The routing neighbourhoods and the scheduling algorithms are then used in Sections 5.5 and 5.6 that describe the former and new re-routing metaheuristics of the AGLIBRARY solver.

5.1. Solution framework

Fig. 1 shows the architecture of the AGLIBRARY solver. The input data are given via an XML file, defining the timetable of scheduled arrival and departure times, the current status of the infrastructure components (block sections and platforms), the

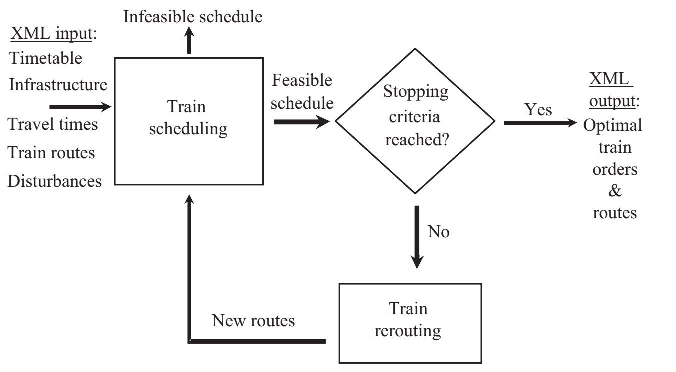

Fig. 1. Architecture of the AGLIBRARY solver.

running time of each train, an off-line defined default route and a set of re-routing options for each train, and a set of disturbances (initial delays, and eventually, disruptions). Given the input data, the AGLIBRARY solver iterates between the computation of a train schedule for a given set of routes, and the selection of a new set of routes. The basic idea is to first compute a train scheduling solution given fixed routes, and then search for better train routes. The solver solution is provided with another XML file, describing the CDR solution in terms of train orders and routes.

If no feasible train schedule is found in a given computation time, the human dispatcher must recover infeasibility manually by taking some decisions that are not allowed to the solver, e.g. the cancellation of a train service. When a feasible train schedule is found, the train re-routing module verifies whether a new set of routes, leading to a potentially better solution, exists or not. Whenever re-routing is performed, the train scheduling module computes a new schedule. The iterative procedure continues till a stopping criterion is met and returns the best CDR solution. We next describe the algorithms we use for each module.

5.2. Branch-and-bound scheduling algorithm

The CDR problem with fixed routes is solved by the branch-and-bound (BB) algorithm of D'Ariano et al. [25], truncated at a given maximum computation time. A near-optimal solution is computed in a short time by this algorithm for practical-size instances. In particular, the algorithm is based on a binary branching scheme in which the branching decision is a sequencing order between two trains in a resource. In the alternative graph, this sequencing decision corresponds to the selection of an alternative arc from each pair $((krj,ump),(umi,krp))\in A$. The branching decision is thus on the arcs (krj,ump) and (umi,krp).

5.3. Routing neighbourhoods

Metaheuristic algorithms are generic solution procedures based on exploring the solution space by means of considering an incumbent solution and iteratively changing it in favour of a new incumbent solution. This action corresponds to the concept of a *move* from a solution to a possibly better one, and it is in general guided by some approximation or evaluation of the objective value, and/or properties of the solution. Commonly more tentative solutions are considered, and a single one is chosen as incumbent. The *neighbourhood* describes the moves that will be considered, based on a certain incumbent solution.

This subsection describes the neighbourhood structures used by the CDR algorithms presented in this paper. To this aim, we need to introduce the following notations. Let $S(F)$ be a CDR solution with the routes defined in F and the sequencing decisions defined in S, and let $\mathcal{G}(F,S)$ be the graph of this solution. The search for a better solution is based on the computation of a new graph $\mathcal{G}'(F', S')$. This graph differs from the former $\mathcal{G}(F, S)$ by a different route for some trains, and different orders and times of operations. This corresponds to a neighbour, in metaheuristics terms. The longest path in $\mathcal{G}'(F', S')$ is denoted as $l^{S'(F')}(0, n)$. We observe that F' improves over F in terms of the objective function value if $l^{S'(F')}(0, n) < l^{S(F)}(0, n)$.

The neighbourhoods studied in this paper are based on observations on the graph $\mathcal{G}(F, S)$ regarding the nodes that represent train operations delayed due to the resolution of potential conflicts between trains. These nodes are *critical* when they are on the longest path from the start node 0 to the end node n in the graph $\mathcal{G}(F, S)$, that is called the *critical path set* $\mathcal{C}(F, S)$. Given a solution $S(F)$, $krp \in N(F) \backslash \{0, n\}$ is a *critical node* of train k with route r if $l^{S(F)}(0, krp) + l^{S(F)}(krp, n) = l^{S(F)}(0, n)$. A critical node krp is a *waiting node* if $l^{S(F)}(0, krp) > l^{S(F)}(0, \nu(krp)) + w^F_{\nu(krp),krp}$, where the node $\nu(krp)$ precedes the node krp on route r. For each waiting node krp, there is at least one *hindering node* $\eta(krp)$ in $\mathcal{G}(F, S)$, different from node $\nu(krp)$, such that $l^{S(F)}(0, krp) = l^{S(F)}(0, \eta(krp)) + w^F_{\eta(krp),krp}$.

Given a node $krp \in N(F) \backslash \{0, n\}$, we recursively define the *backward ramification* $R_B(krp)$ as follows. If node krp is waiting, then $R_B(krp) = R_B(\nu(krp)) \cup R_B(\eta(krp) \cup \{krp\}$, otherwise $R_B(krp) = R_B(\nu(krp)) \cup \{krp\}$. Similarly, we recursively define the *forward ramification* $R_F(krp)$ as follows. If node krp is the hindering of a waiting node abc, then $R_F(krp) = R_F(\sigma(krp)) \cup R_F(abc) \cup \{krp\}$, where the node $\sigma(krp)$ follows the node krp on route r. Otherwise, $R_F(krp) = R_F(\sigma(krp)) \cup \{krp\}$. By definition, $R_B(0) = R_F(0) = \{0\}$ and $R_B(n) = R_F(n) = \{n\}$. Given $\mathcal{C}(F, S)$, we define a *ramified critical path* set as $\mathcal{F}(F, S) = \bigcup_{krp \in \mathcal{C}(F,S)}[R_B(krp) \cup R_F(krp)]$, and a *backward ramified critical path* set as $\mathcal{B}(F, S) = \bigcup_{krp \in \mathcal{C}(F,S)}[R_B(krp)]$. We study the following five neighbourhood structures.

- *Complete K-Route neighbourhood* $\mathcal{N}_{CKR}$ contains all the feasible solutions to the CDR problem in which K trains follow a different route compared to the incumbent solution. To limit the number of neighbours to be evaluated, $\mathcal{N}_{CKR}$ is only partially explored as follows. A move is obtained by choosing different routes from the ones of the current solution at random (i.e. all alternative routes having the same probability) for K trains, until a number ψ (parameter) of alternative routing solutions is obtained.
- *Ramified Critical Path Operations neighbourhood* $\mathcal{N}_{RCPO}$ considers only the routing alternatives for the trains associated to the nodes in $\mathcal{B}(F, S)$ plus $\mathcal{F}(F, S)$. The idea is that the maximum consecutive delay of a solution to the CDR problem can be reduced by removing some train conflicts causing it. This requires either removing, anticipating or postponing some train operations from the ramified critical path set. The latter result can be obtained by re-routing the trains associated with the ramified critical path operations through different resources (i.e. by re-routing some trains associated to the nodes in $\mathcal{B}(F, S)$ or $\mathcal{F}(F, S)$), and then re-scheduling train movements.
- *Waiting Operations Critical Path neighbourhood* $\mathcal{N}_{WOCP}$ is a restriction of $\mathcal{N}_{RCPO}$ that considers the routing alternatives for the trains associated to the waiting nodes in $\mathcal{C}(F, S)$.
- *Delayed Jobs neighbourhood* $\mathcal{N}_{DJ}$ considers only the trains (jobs) that have a consecutive delay at some relevant locations on the incumbent solution.
- *Free-Net Waiting Operations Jobs neighbourhood* $\mathcal{N}_{FNWJ}$ considers only the trains (jobs) that have some waiting nodes in the graph of the incumbent solution in which all alternative arcs are unselected. The alternative graph with no alternative arc selected corresponds to the *free-net traffic situation* in which each train travels in the absence of conflicts.

The appendix of this paper will illustrate some neighbourhood structures for an illustrative example.

5.4. Heuristic evaluation of routing neighbours

The choice of a best neighbour in the neighbourhood requires the computation of a new CDR solution $S'(F')$ starting from an incumbent solution $S(F)$ that is characterized by the train routing decisions in F' and the train sequencing decisions in S'. To this aim, we use fast heuristics based on a two-step graph building procedure in which the graph $\mathcal{G}(F, S)$ is translated into the graph $\mathcal{G}'(F', S')$. In the first step, a sub-graph of $\mathcal{G}'(F', S')$ is generated by considering all the nodes in $N(F^I)$ associated to the routes modelled by the arcs in $F^I = F \bigcap F'$, all the fixed arcs $\in F^I$ and all the alternative arcs in $S(F)$ incident in nodes in $N(F^I)$. This corresponds to keeping a subset of decisions from the incumbent solution into the neighbour solution. In the second step, the fixed arcs in $F^R = F' \backslash F^I$ and the nodes in $N(F^R)$ are added to the sub-graph. Finally, $\mathcal{G}'(F', S')$ is obtained by adding a selection of alternative arcs $S'(F^R)$ to the sub-graph.

The selection $S'(F^R)$ is computed by selecting the best solution among two greedy algorithms based on the idea of repeatedly enlarging a selection by choosing an unselected pair at a time from the set A and by selecting one of the two arcs until a feasible schedule is found or an infeasibility (i.e. a positive weight cycle in the graph) is detected [45]. The first greedy algorithm AMSP (*Avoid Most Similar Pair*) chooses an unselected alternative pair $((krj, ump), (umi, krp)) \in A$ maximizing the quantity $l^{S'}(F^R)(0, krj) + w^A_{krj,ump} + l^{S'(F^R)}(ump, n) + l^{S'(F^R)}(0, umi) + w^A_{umi,krp} + l^{S'(F^R)}(krp, n)$. The other greedy algorithm AMCC (*Avoid Most Critical Completion Time*) chooses the alternative pair $((krj, ump), (umi, krp)) \in A$ such that the quantity $l^{S'(F^R)}(0, krj) + w^A_{krj,ump} + l^{S'(F^R)}(ump, n)$ is maximum among all the unselected alternative arcs. Both algorithms select the arc of the pair causing the minimum consecutive delay.

5.5. Tabu search re-routing algorithm

The *Tabu Search* (TS) is a deterministic metaheuristic based on local search, which makes extensive use of memory for guiding the search. A basic ingredient is the *tabu list* that is used to avoid being trapped in local optima and revisiting the same solution. From the incumbent solution, non-tabu moves define a set of solutions, named the *incumbent solution neighbourhood*. At each step, the best solution in this set is chosen as the new incumbent solution. Some attributes of the former incumbent are then stored in the tabu list. 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.

The Tabu Search (TS) used in this paper for the CDR problem is the algorithm of Corman et al. [13]. Two neighbourhood strategies for the maximum consecutive delay minimization problem are investigated namedRestart and *Complete*. Each neighbourhood strategy restricts the set of moves to be explored in order to speed up the search of the best move. In particular, the Complete strategy explores ψ (parameter) randomly chosen neighbours in $\mathcal{N}_{CKR}$ with $K = 1$, while the Restart strategy selects at most ψ promising moves in $\mathcal{N}_{RCPO}$, unless this neighbourhood is empty. When no potentially better solution is found on the incumbent solution neighbourhood, the search alternates the neighbourhood strategy with a diversification strategy, which consists of changing at random the route of μ (parameter) trains at the same time.

In this paper, all neighbours are evaluated via the scheduling heuristics of Section 5.4. The best neighbour is set as the move to be made, and re-evaluated via the branch-and-bound scheduling algorithm of Section 5.2; the resulting best CDR solution is set as

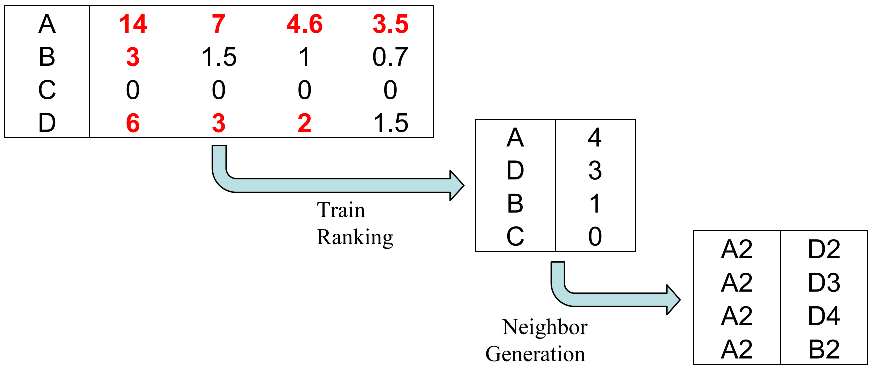

Fig. 2. Example of neighbourhood search strategy with $|J| = 4$, L=4 and K=2.

the new incumbent solution. The inverse of the chosen move is stored in a tabu list of length λ (parameter). The moves in the tabu list are forbidden for λ iterations and no aspiration criteria is used. From the tuning performed in [13], the best exploration strategies have the following parameters: for the Complete strategy the best values of (ψ, λ, μ) are (8; 3; 5); for the Restart strategy the best values of (ψ, λ, μ) are (8; 27; 5).

5.6. Variable neighbourhood search re-routing algorithms

Variable neighbourhood search (VNS) metaheuristics are presented in order to efficiently solve the CDR problem. This type of metaheuristic is based on the combination of neighbourhood structures. Systematic changes of the neighbourhood structures are proposed both in a local search phase in order to compute a local minimum, and in a perturbation phase in order to escape from a local minimum [30].

In this work, the classic ingredients of the VNS algorithm are combined with new routing neighbourhood structures and new specific neighbourhood search strategies to search for better train routing combinations. A move is obtained by choosing a set of routes different from the ones of the incumbent solution. Various variable neighbourhood schemes from [30] and routing neighbourhoods of Section 5.3 are implemented, differing in the set of candidate trains that are re-routed in each move.

The choice of investigating these search methods is motivated by the following facts: (i) the train scheduling solution with fixed routes can be improved in terms of multiple train routing modifications, (ii) there is a need of improving upon the local minima found by local search. From (i), we need to explore possibilities to generate new solutions starting from some reference solutions. From (ii), we need to develop strategies to spread the search for better quality solutions. For these reasons, we investigate VNS intensification and diversification strategies. The main algorithmic ingredients are next introduced and optionally incorporated in various versions of the variable neighbourhood search algorithm.

Build neighbourhood: Starting from an incumbent solution, the $\mathcal{N}_{CKR}$ neighbourhood is generated, in which exactly K trains are re-routed in the graph $\mathcal{G}(F,S)$ of the incumbent solution.

Shaking procedure: This is a typical diversification procedure that consists in changing the route of K trains randomly in the $\mathcal{N}_{CKR}$ neighbourhood of the incumbent solution (*IncSol*), and in computing a new incumbent solution (*IncSol*$'$) via the scheduling heuristics of Section 5.4 and the new set of routes.

Neighbourhood search strategy: This procedure is proposed in order to limit the local search to the evaluation of up to L neighbours in the current neighbourhood. Starting from an incumbent solution and the $\mathcal{N}_{CKR}$ neighbourhood of this solution, a restricted neighbourhood is generated by using a given neighbourhood structure $\mathcal{N}_i$. The selection of L neighbours is achieved in the following steps:

1. *Train ranking*: Each train gets a score based on the criterion specified in $\mathcal{N}_i$. The train ranking is based on one of the neighbourhood structures of Section 5.3. In $\mathcal{N}_{RCPO}$, each train on the ramified critical path gets a score based on the maximum value $l^{S'(F^R)}(0,krp)+l^{S'(F^R)}(krp,n)\ \forall(krp)$ in the ramified critical path of the graph of the incumbent solution. In $\mathcal{N}_{WOCP}$, each train gets a score based on the sum of the consecutive delays collected at each critical node in the graph of the incumbent solution. In $\mathcal{N}_{DJ}$, each train gets a score based on the maximum consecutive delay collected at some relevant locations for each job. In $\mathcal{N}_{FNWJ}$, each train gets a score based on the sum of the consecutive delays collected at each waiting node in the graph of the incumbent routing solution in which all alternative arcs are unselected (i.e. free-net traffic situation) but the one generating the waiting node. The scores are used to decide how many times each train has to be re-routed in the L neighbours.
2. *Route ranking*: The routes of each train get a score based on the distance from the route of the incumbent solution. The larger is the difference between the routes, the higher is the score. The route ranking thus suggests for each train to select the most different routes.
3. *Neighbour generation*: This is the assignment of the routes to the trains in each neighbour. The trains to be re-routed are selected via the train ranking and the train routes are selected via the route ranking. A combinatorial combination of the routes is used in order to generate L different neighbours. In each neighbour, exactly K trains are re-routed compared to the incumbent solution. The neighbours are ordered based on the train ranking, and in case of tie on the route ranking.

Numerical example regarding the neighbourhood search strategy: Fig. 2 presents a numerical example of the neighbourhood search strategy, in which four trains ($J=\{A,B,C,D\}$) can be re-routed in a railway network. Trains A and D have a default route and three alternative routes, while trains B and C have a default route and five alternative routes. As an example, the routes of B are named $B1$, $B2$, $B3$, $B4$, $B5$, $B6$, with the first one $B1$ being the default route. Detailed information regarding the traffic flows and the example network is reported in the paper appendix.

In the given incumbent CDR solution, the default route is used by all trains (i.e. the routes $A1$, $B1$, $C1$, $D1$). The parameters of the procedure are set to the following values: L=4 and K=2 (i.e. the neighbourhood is restricted to 4 neighbours and 2 trains are re-routed in each neighbour).

The train ranking procedure determines a score matrix in which each row represents a train and each column reports a score used to compute the number of times each train should be

```
Algorithm Variable Neighbourhood Search
Input: $IncSol\ \mathcal{G}(F,S)$, $K_{max}$, $T_{max}$, $L$, $\mathcal{N}_1$, $\mathcal{N}_2$
$\mathcal{N}_i \leftarrow \mathcal{N}_1$,
While $(T < T_{max})$ & $(f(IncSol) > 0)$ & $(OtherStoppingCriterion)$ do
    Begin
    $K \leftarrow 1$,
    While $(K \leq K_{max})$ do
        Begin
        $BuildNeighbourhood(IncSol,\ K)$,
        $IncSol' \leftarrow GenerateNewIncumbentSolution(IncSol,\ K,\ L,\ \mathcal{N}_i)$,
        $(IncSol,\ K) \leftarrow MoveOrNot(IncSol,\ IncSol',\ K)$,
        If $(K = K_{max})$ do
            Begin
            $\mathcal{N}_i \leftarrow NeighbourhoodChange(IncSol,\ \mathcal{N}_i,\ \mathcal{N}_1,\ \mathcal{N}_2)$,
            End
        $T \leftarrow$ CPU time()
        End
    End
```

Fig. 3. General sketch of the metaheuristics.

considered for re-routing. This score matrix is depicted in the left-hand side of Fig. 2. Specifically, the first column reports the score for each train based on the neighbourhood structure $\mathcal{N}_{DJ}$ (e.g. the value 14 in this example is the maximum consecutive delay collected for $A1$, see Appendix). The other columns report the score of the first column divided by the number of column, e.g. 14/2=7, 14/3=4.6, 14/4=3.5.

The procedure takes the highest KL scores that are provided in the score matrix (the scores are in bold in Fig. 2). In other words, from the largest values in the score matrix we obtain the number of times each train has to be re-routed in the four neighbours (e.g. train A will be re-routed in all four neighbours). This information is reported in the centre table of Fig. 2, in which the four trains are ordered according to the number of times they will be re-routed.

The route ranking orders the list of re-routing alternatives for each train based on maximizing the difference with the incumbent route, in terms of the number of different operations between the train routes. As an example, the route ranking of A is $A2$, $A3$, $A4$ and the most different route is $A2$; the route ranking of D is $D2$, $D3$, $D4$ and the most different route is $D2$; the most different route for train B is $B2$.

The neighbour generation procedure assigns the routes to the trains in each neighbour. In this example, train A appears in all the neighbours with one of its alternative routes; trains D and B appear with respectively three routes and one route. The neighbours are obtained as follows. We first re-route the two trains with higher train ranking (A and D) and then re-route the remaining train (B) with the train with the highest train ranking (A). Among the neighbours with the same re-routed trains, we first consider the current most different route for the train with the current higher train ranking, and then consider the current most different route for the next train with the current higher train ranking, and so on. Every row of the table in right-hand side of Fig. 2 corresponds to a candidate move.

Best improvement strategy: This is a local search procedure in the restricted neighbourhood in which all the candidate moves are evaluated via the train scheduling heuristics. This procedure lasts until all L neighbours in the restricted neighbourhood have been evaluated. At each step of the procedure, a neighbour is considered, a new graph is built with the new routes of the neighbour, and a new CDR solution is computed via the scheduling heuristics for the new set of routes. The best neighbour is set as the move to be made, and re-evaluated via the branch-and-bound scheduling algorithm of [25]; the resulting best CDR solution is set as the new incumbent solution.

Move Or Not procedure: This procedure is responsible for possibly performing a move. In case the best solution found in the neighbourhood is better than the incumbent, the resulting train scheduling problem is solved by the branch-and-bound algorithm of [25], and the best solution is set as the new incumbent solution. Otherwise, the best solution in the neighbourhood is chosen as incumbent, or some diversification strategy is employed depending on the adopted VNS scheme.

Neighbourhood change: This procedure is used to diversify the search by alternating K_{max} applications of the neighbourhood search strategy with $\mathcal{N}_1$ and K_{max} applications of the neighbourhood search strategy with $\mathcal{N}_2$.

The metaheuristics proposed in this paper are an adaptation of the VNS schemes described in Hansen et al. [30]. Specifically, we consider the four algorithmic schemes named “VND”, “General VNS”, “Basic VNS”, “Reduced VNS”. The general structure of the studied metaheuristics is the following. The metaheuristics start from an incumbent solution *IncSol* of the CDR problem, computed via the branch-and-bound scheduling algorithm of [25] by assigning a default (off-line) route to each train. In each metaheuristic, a counter K is adopted to fix the number of trains that are re-routed in each move. The initial value of K is set to 1, i.e. a single train is re-routed in *IncSol*. The metaheuristics iterate the search for better solutions starting from *IncSol* until a stopping criteria is reached.

The stopping criterion of all the metaheuristics is a maximum computation time T_{max} or the particular situation in which the maximum consecutive delay (i.e. the best objective function value *f*(*IncSol**)) is equal to 0. Additionally, VND stops the search when no improvement is obtained during a local search.

At each iteration, a neighbourhood of *IncSol* is generated and a new solution $IncSol'$ is selected. Then, the *Move Or Not* function is performed as follows. In case an improving move $IncSol'$ is obtained via the local search (i.e. $f(IncSol') < f(IncSol)$, a new iteration is performed by setting $IncSol'$ as the new incumbent solution and K is set to 1. Otherwise, the parameter K is set to $K+1$ and a new iteration is performed until $K \le K_{max}$. When $K = K_{max}$ the algorithm diversifies the search with a change of neighbourhood structure (if the algorithm works with a single neighbourhood structure this step is not performed).

The general pseudo-code of the variable neighbourhood search algorithms is reported in Fig. 3. Each metaheuristic returns the best CDR solution (*IncSol**) and the objective function value (*f* (*IncSol**)).

The iterative step of the metaheuristics studied in this paper differs in the choice of $IncSol'$:

VND: This is a deterministic version of variable neighbourhood search in which a local search is performed in a restricted neighbourhood of *IncSol*, via the neighbourhood search strategy (for a given value of parameters L and $\mathcal{N}_i$) and the best improvement strategy. The best neighbour is set as $IncSol'$.

General VNS: This is a generalization of the VND in which the neighbourhood of *IncSol* is generated and a solution $IncSol''$ is selected via the shaking procedure (for a given value of parameter K) that takes a neighbour of *IncSol* at random. The VND algorithm is applied starting from the solution $IncSol''$. The resulting solution is $IncSol'$.

Basic VNS: This version of VNS combines the deterministic and random changes of neighbourhoods. This algorithm first performs the shaking procedure (for a given value of parameter K) on the incumbent solution *IncSol*, obtaining a solution $IncSol''$. Then, a local search is performed in a restricted neighbourhood of $IncSol''$, via the neighbourhood search strategy (for a given value of parameter L and neighbourhood structure $\mathcal{N}_i$) and the best improvement strategy. The best neighbour is set as $IncSol'$.

Reduced VNS: This is a completely randomized version of the VNS metaheuristic. The shaking procedure (for a given value of parameter K) is performed starting from the incumbent solution *IncSol*. The resulting solution is set as $IncSol'$. This VNS can be viewed as a reduction of Basic VNS in which the local search is not performed. The rationale is to look at a larger number of neighbours compared to Basic VNS, even if these are randomly selected. However, we note that this metaheuristic scheme has a high risk of re-evaluating the same solutions several times, and thus being trapped into a local optimum.

6. Computational experiments

This section presents the experimental results on the TS, VND and VNS metaheuristics of Section 5.

Four practical railway test cases are investigated in a laboratory environment:

- an Italian single-track network (named “Italian (I) test case”);
- a Dutch double-track network between Utrecht and Den Bosch (named “First Dutch (FD) test case”);
- a Dutch busy and complex area around Utrecht central station (named “Second Dutch (SD) test case”);
- a British double-track network nearby the city of London (named “British (B) test case”).

All the studied test cases are modelled with a microscopic level of detail, which means that switches, signals, block sections, and track segments in complex station areas are considered (yielding several hundreds of resources per test case). Furthermore, train movements are described with a precision of seconds.

For each test case, we consider a set of 20 traffic disturbance instances, varying the initial delays of trains. The experiments are executed on a workstation Power Mac with processor Intel Xeon E5 quad-core (3.7 GHz), 12 GB of RAM. The algorithms are implemented in AGLIBRARY and use a total computation time T_{max} = 180 s. The MILP formulation of the CDR problem is solved by using the commercial solver: IBM LOG CPLEX MIP 12.0 that is executed with a time limit of 2 h.

Table 1 presents the parameters studied for the variable neighbourhood search algorithms. Regarding the TS algorithm, the parameters are set as described in Section 5.5 (according to the parameter tuning in [13]), expect for the BB computation time limit that is set as for the VND/VNS algorithms. In Table 1, the assessment of the new algorithms is based on a set of pilot CDR instances with T_{max} = 180 s.

Regarding the information provided in Table 1, Column 1 reports the maximum computation time (in seconds), Column 2 the computation time given to the branch-and-bound scheduling algorithm (BB Time, in seconds), Column 3 the number of

Table 1
Experimental setting of the algorithmic parameters.

T_{max}	BBTime (s)	K_{max}	L
180	2/**4**/8/12	3/4/**5**/7	5/**10**/15/20

Table 2
Best configurations of the CDR algorithms for each test case.

Test case	Best TS [13]	Best VND	Best VNS General	Best VNS Basic	Best VNS Reduced
Italian (I)	Complete	DJ	WOCP	WOCP	WOCP
First Dutch (FD)	Restart	WOCP+FNWJ	FNWJ	DJ+WOCP	DJ+FNWJ
Second Dutch (SD)	Complete	DJ	WOCP+DJ	DJ	FNWJ+DJ
British (B)	Restart	DJ	WOCP	WOCP	FNWJ

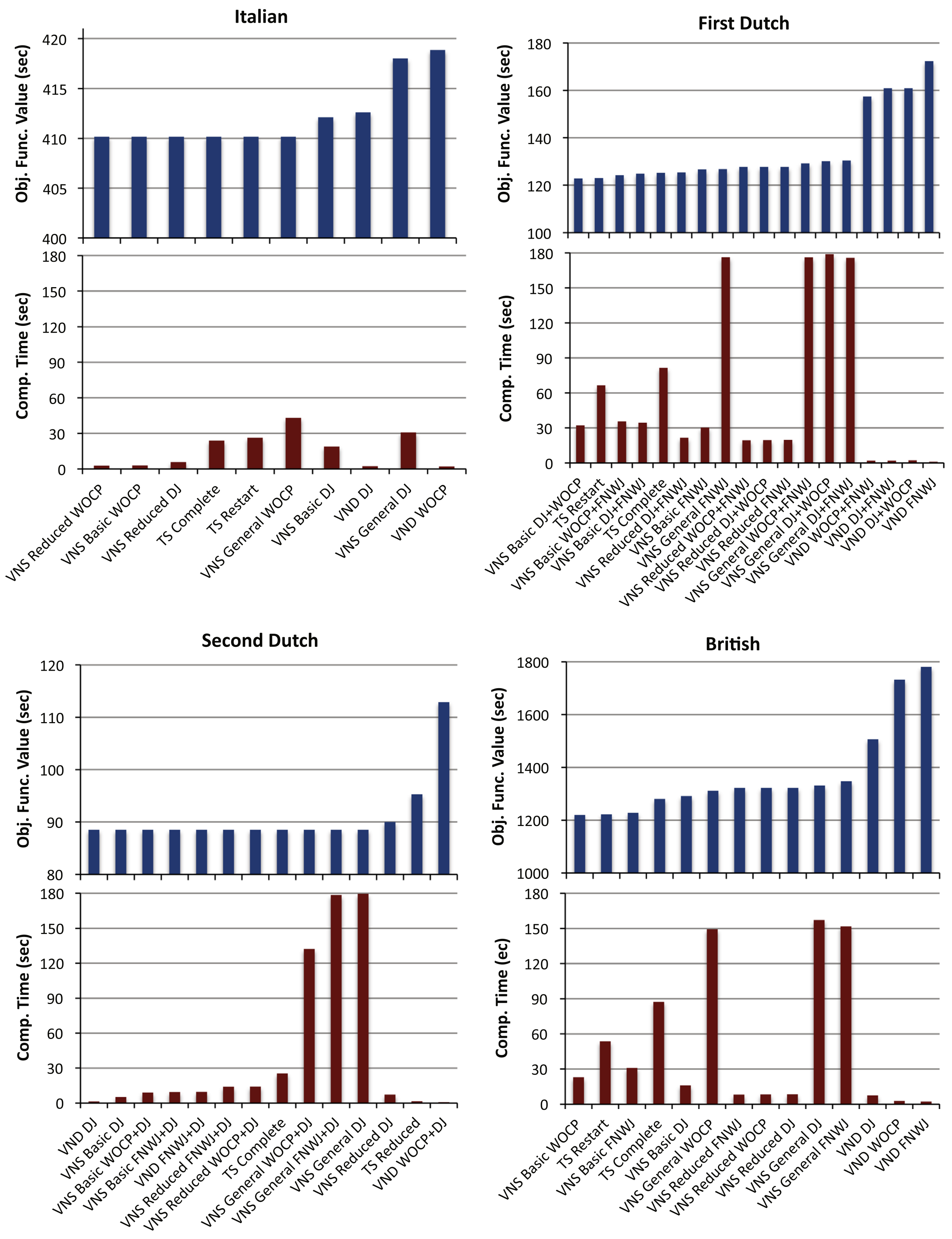

Fig. 4. Quantitative comparison between configurations of the CDR algorithms.

trains that are rerouted in the current neighbourhood (K_{max}), Column 4 the size of the restricted neighbourhood (L). The best value of each parameter is reported in bold.

Table 2 presents the best configuration of each CDR algorithm for each test case. For the TS algorithm we report the best search strategy, while for the VND and VNS algorithms we report the best

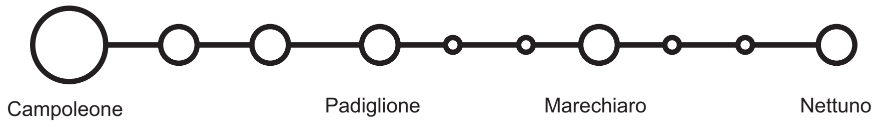

Fig. 5. Italian test case – the Campoleone Nettuno line.

Table 3
Characteristics of the Italian test case instances.

Time horizon (min)	Network length (km)	Number of trains	Number of routes per train	Number of resources per train	MILP variables		
					$\lvert N\rvert$	$\lvert A\rvert$	$\lvert C\rvert$
720	26	42	8	18	1094	752307	336

combination of the neighbourhood structures. In Table 2, the assessment of the TS, VND and VNS algorithms is performed with T_{max}=180 s, and is based on the set 20 traffic disturbance instances for each test case. We used the following two criteria in lexicographic order of importance to establish the best algorithmic configuration: (1) the average value of the objective function of the CDR problem; (2) the average time required to find the best CDR solution. Fig. 4 presents these two key performance indicators that are used to evaluate logically the performance of the algorithms. For each test case we show the two best search strategies for TS, the best combinations of the neighbourhood structures for VND and VNS. Specifically, the best neighbourhood structures are: WOCP, DJ for the I test case, DJ+WOCP, WOCP+FNWJ, DJ+FNWJ, FNWJ for the FD test case, DJ, WOCP+DJ, FNWJ+DJ for the SD test case, WOCP, FNWJ, DJ for the B test case.

In the next subsections, we use the best configuration of each algorithm as indicated in Tables 1 and 2. Specifically, we present a detailed assessment of the computational results obtained for each CDR instance by the best TS, VND and VNS algorithms and by the commercial MILP solver. The computational experiments are reported per test case, together with additional information on the tested CDR instances. We conclude the section with some general discussion and observation on the obtained results.

6.1. *Results on the Italian test case*

Fig. 5 shows a schematic view of the dispatching area of Campoleone–Nettuno, (i.e. the regional line FR8) with the amount of tracks per branch, and stations or minor stops. The railway network consists of 10 stations and 9 bidirectional single-track segments between stations. Most stations have two parallel platform tracks to allow re-routing and meet-pass operations. The railway infrastructure is around 26 km long. There are potential conflict points (where the dispatcher takes ordering and routing decisions) at the entrance and exit of each station. In Campoleone, there is a connection with the regional line FR7 and with the line towards Roma Termini station, the main station in Rome.

We consider a daily timetable which describes the movement of all trains running in the line during day hours, specifying, for each train, planned arrival/passing times at each station platform along its route. At stations, a train is not allowed to depart from a platform stop before its scheduled departure time and is considered late if arriving at the platform after its scheduled arrival time. The current daily timetable has 42 trains, with an average travel time for passengers of 42 min. Traffic disturbances are studied in which a set of trains is delayed at their entrance in the network.

Table 3 reports on the CDR instances of the Italian test case. Column 1 reports the time horizon of traffic prediction (in minutes), Column 2 the approximate length of the railway network (in kilometers), Column 3 the number of trains in the network during the entire horizon of traffic prediction, Column 4 the average number of routes assigned to each train (including the default route), Column 4 the average number of resources traversed by each train, Columns 5–7 the average size of the MILP formulation in terms of the number of timing variables (i.e. the set $|N|$), the number of train scheduling variables (i.e. the set $|A|$) and the number of train routing variables (i.e. the set $|C|$).

This single-track railway network presents several alternative routings, since each train can be routed in several stations, where two parallel platform tracks are available.

Table 4 gives the computational results for 20 CDR instances of the Italian test case. In Column 1, each CDR instance is identified by a three-field code [α_β_γ], in which α identifies the network, β is the maximum initial delay (in seconds), and γ is the average initial delay (in seconds). We recall that the initial delay is caused by disturbances and cannot be recovered by re-scheduling train movements, except by using the available time reserves in the timetable. In the other columns, the performance of each algorithm/solver is presented in terms of the objective function value (i.e. the maximum consecutive delay, in seconds) and the time to compute the best solution (in seconds). The best average objective function values are reported in bold. For each CDR algorithm, we only present the results obtained for the best configuration of Table 2.

From the results of Table 4, VNS Basic and VNS Reduced are the best algorithms in terms of both indicators, while the other algorithms either present a larger delay or a larger computation time. VND is often the fastest algorithm but it does not provide the best-known solutions, while VNS Reduced uses VND combined with the shaking procedure and is able to compute the best-known solutions. TS and VNS General are, on average, much slower than VNS Basic and VNS Reduced. CPLEX is always outperformed by the CDR algorithms in terms of both indicators, expect for instance I_10000_2618.3. Furthermore, the lower bound returned by CPLEX is very low and does not certify the optimality of any CDR solution. For this set of CDR instances the low quality of CPLEX is probably due to the fact that the single-track train scheduling problem presents several infeasible train timing and ordering solutions, resulting in deadlock situations.

6.2. *Results on the first Dutch test case*

Fig. 6 presents the Utrecht – Den Bosch dispatching area, that consists of 191 block sections and 21 platforms. The railway network is around 50 km long, connecting the cities of Utrecht and Den Bosch. There are two main tracks, a long corridor for each traffic direction, a dedicated stop for freight trains nearby Zaltbommel and 7 passenger stations.

The infrastructure offers some possibility of train re-ordering and re-routing. Each train has a default route and a set of local re-routing options. Re-routing options can be applied along corridors or within a station, in which a train may be allowed to stop at different nearby platforms. Only standard train routes are considered and some less important switches have been omitted. Considering all possible alternative re-routing options yields a set of 356 routes. Fig. 6 shows the dispatching area considered with the amount of tracks per area, and the indication of minor/major stations.

Table 4
Results obtained for the Italian (I) test case instances.

CDR instance	TS [13]		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)
I_4000_986.3	237	0.6	237	0.6	237	89.3	237	0.6	237	4.6	79,390	364.2
I_3864_965.6	262	0.6	262	0.6	262	0.6	262	0.6	262	0.6	458	7189.2
I_3883_983.3	270	0.6	270	0.6	270	0.6	270	0.6	270	0.6	83,848	1042.1
I_4000_981.4	249	0.6	249	0.6	249	0.7	249	0.6	249	0.6	56,463	6826.8
I_3976_1077.8	276	0.6	276	0.6	276	0.6	276	0.6	276	0.6	83,471	1644.9
I_3646_949.3	261	0.6	261	0.6	261	164.6	261	0.6	261	0.6	62,933	6023.6
I_4000_979.8	300	0.6	300	0.6	300	0.6	300	0.6	300	0.6	80,187	4833.1
I_4000_971.8	297	0.6	297	0.6	297	0.6	297	0.6	297	0.6	84,282	1325.7
I_3981_981.9	385	1.0	385	1.0	385	1.0	385	1.0	385	1.0	396	5537.1
I_10000_2618.3	436	7.2	436	13.9	436	129.1	436	3.8	436	5.9	64,808	6828.3
I_10000_2629.8	605	23.2	605	0.7	605	0.6	605	0.6	605	0.7	76,458	5547.3
I_9553_2579.1	539	67.6	539	7.7	539	9.0	539	4.6	539	8.0	83,781	1175.5
I_10000_2688	577	135.2	626	3.8	577	174.4	577	19.9	577	7.9	51,731	6457.6
I_10000_2600.1	467	0.6	467	0.7	467	7.6	467	0.6	467	0.7	66,631	6059.3
I_9739_2689.4	345	0.6	345	0.7	345	0.6	345	0.6	345	0.7	55,803	6552.6
I_9008_2504.5	787	33.8	787	0.6	787	0.6	787	0.6	787	0.7	59,847	5822.8
I_9489_2607.9	638	7.1	638	3.6	638	147.4	638	2.8	638	13.9	84,785	900.9
I_10000_2615.3	486	151.4	486	6.4	486	133.4	486	16.4	486	7.7	84,002	1343.1
I_10000_2618.3	448	1.3	448	1.3	448	1.2	448	1.2	448	1.3	448	6456.1
I_10000_2584.3	338	0.7	338	0.7	338	0.6	338	0.6	338	0.8	344	7190.1
Avg results	**410.2**	21.7	412.6	2.3	**410.2**	43.1	**410.2**	2.9	**410.2**	2.9	58,003.3	4456.0

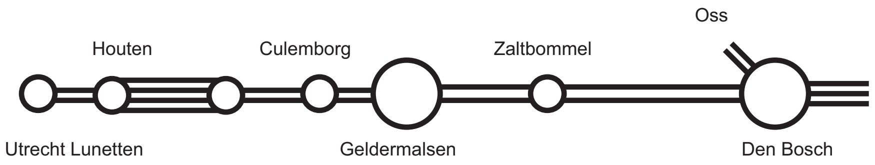

Fig. 6. First Dutch test case – the Utrecht – Den Bosch area.

We consider a provisional hourly timetable for 2007 extended to the entire railway area. During peak hours, 26 passenger and freight trains are scheduled, in both directions, for the area around Geldermalsen. A more complex situation occurs at Den Bosch station, where up to 40 trains are scheduled each hour.

Additional constraints are included on the minimum transfer time between connected train services. Connections of the rolling stock are provided in Zaltbommel and Den Bosch stations. Passenger connections are located at Den Bosch station for the traffic directions from Oss to Utrecht and vice versa. The minimum time for passenger transfer connections varies from two to five minutes, depending on the distance to be travelled between the arrival platforms.

Table 5 presents average information on the CDR instances of this test case, with trains subject to random initial delays. This network presents a huge number of $|A|$ variables, since these are defined for each pair of trains that have routes sharing resources of the 50-km-long railway network.

Table 6 reports on the computational results for 20 CDR instances of the Utrecht – Den Bosch test case. The instance code is as for Table 4. The best average results are obtained for VNS Basic in terms of the objective function value (as reported in bold). Specifically the best-known solution for instance FD_738_256.1 is only computed by VNS Basic and VNS General. The computation time of VNS Basic is decreased by more than half compared to TS. However, TS, VND and VNS Reduced are faster to compute the best-known solution for some instances. VNS General is the slowest algorithm to compute the best-known solution.

Regarding the results obtained by the CPLEX solver, the lower bound is used to certify the optimality of some CDR solutions, while the upper bound is sometime good, even if those bounds are computed in a very long computation time. In Table 6, four CDR instances are solved to proven-optimum by some algorithms. The optimal solutions are identified with an asterisk in the columns regarding the objective function value. Two optimal solutions are also computed by CPLEX, while the other two are only certified by CPLEX. Overall, the optimality of several CDR instances is not proved by CPLEX, since the overall problem has a huge number of train ordering ($|A|$) variables required when combining all possible routing alternatives.

Table 5
Characteristics of the first Dutch test case instances.

Time horizon (min)	Network length (km)	Number of trains	Number of routes per train	Number of resources per train	MILP variables		
					$\|N\|$	$\|A\|$	$\|C\|$
60	50	40	9	31	1615	1,092,557	356

6.3. Results on the second Dutch test case

This test case is based on the railway network around the central station of Utrecht, the busiest station in the Netherlands. Fig. 7 shows the overall network layout that has a diameter of around 20 km and 600 block sections, with the amount of tracks per branch, and the indication of minor/major stations considered. The main station area (known as Utrecht Central) provides 20 platform tracks, more than 100 switches and around 200 block sections. There are 5 main traffic directions that are delimited by the following minor stations: Utrecht Overvecht (on the line

Table 6
Results obtained for the First Dutch (FD) test case instances.

CDR instance	TS [13]		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)
FD_770_273.4	151	1.5	191	4.2	151	175.8	151	10.4	151	82.3	3900	6836.1
FD_738_256.1	68	59.0	78	0.4	64	176.2	64	45.3	71	11.9	91	6972.5
FD_991_401.1	154	1.2	154	0.3	154	177.6	154	0.7	154	2.7	307	6954.0
FD_1356_495.7	106*	86.9	187	0.0	117	177.9	106*	63.1	106*	57.9	3552	7195.4
FD_1249_534.0	119	117.5	301	0.7	130	180.0	119	81.9	130	48.0	3630	1903.9
FD_972_306.8	202	164.6	257	0.1	225	179.2	202	172.8	208	104.6	4349	7047.0
FD_1321_412.1	68	28.1	113	1.3	68	177.7	68	54.1	68	26.6	197	5872.0
FD_1372_379.6	291	6.4	278	25.9	291	167.5	291	4.0	291	4.0	4239	6292.5
FD_1776_607.0	135	9.5	181	0.1	135	180.2	135	1.9	135	4.5	410	1545.4
FD_659_133.1	94	144.1	142	0.5	98	179.7	94	30.7	94	5.8	111	6047.0
FD_816_158.2	114*	56.0	137	0.3	114*	179.1	114*	17.0	114*	32.6	114*	5272.0
FD_977_191.4	91*	27.4	176	0.4	91*	178.8	91*	92.4	91*	8.9	211	7064.4
FD_1017_201.3	85	42.4	97	0.4	85	179.5	85	0.5	85	2.0	4539	7018.0
FD_1240_293.5	103	143.9	103	1.2	103	178.5	103	2.0	103	14.5	4577	1969.3
FD_1312_294.6	123	116.6	123	2.4	123	178.3	123	12.3	123	10.1	3847	6780.6
FD_888_84.9	71	112.1	116	0.6	102	176.7	71	41.7	98	11.1	916	7132.6
FD_872_117.2	148	0.0	148	0.0	148	158.6	148	0.0	148	0.1	148	4418.1
FD_1371_182.8	122	60.5	150	0.5	122	177.2	122	4.2	122	0.6	3630	1898.1
FD_1769_216.2	99	55.2	99	0.3	99	172.6	99	0.3	99	3.3	99	6661.7
FD_1788_313.3	116*	96.1	116*	0.3	116*	172.9	116*	6.4	116*	0.4	116*	5909.3
Avg results	123.0	66.5	157.4	2.0	126.8	176.2	**122.8**	32.1	125.4	21.6	1949.2	5539.5

Table 7
Characteristics of the second Dutch test case instances.

Time horizon (min)	Network length (km)	Number of trains	Number of routes per train	Number of resources per train	MILP variables		
					$\|N\|$	$\|A\|$	$\|C\|$
75	20	79	3	22	2549	44,730	228

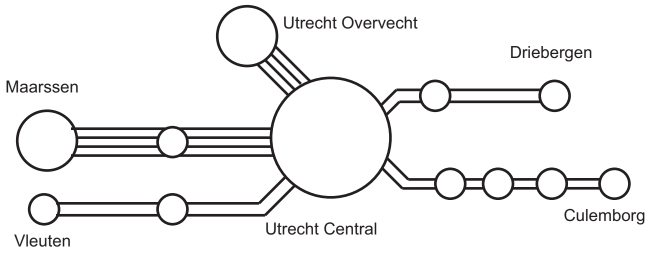

Fig. 7. Second Dutch test case – the Utrecht Central Station area.

towards Amersfoort), Driebergen-Zeist (on the line towards Arnhem and Germany), Culemborg (on the line towards Den Bosch), Maarssen (on the line towards Amsterdam), and Vleuten (on the line towards Rotterdam and The Hague).

The reference timetable is periodic with a cycle length of one hour. The timetable schedules 79 trains in a peak hour, with mixed passenger and freight traffic flows. The passenger trains are divided into International services going from the Netherlands to Germany and vice versa, Intercity services, Local trains and Sprinter services (faster local trains). The timetable provides connections between passenger services, coupling and splitting of rolling stock for intercity and local services coming from/going to Rotterdam, the Hague or Amersfoort, as well as re-use of rolling stock for commuter services towards Utrecht Overvecht and Culemborg. The alternative graph model of the traffic running on this complex station area is based on the aggregated formulation of the station routings at the interlocking areas described in [18].

Table 7 presents average information on the CDR instances tested for this network. For this network, the routing alternatives are platforming options within the main station area of Utrecht. We limit the number of stop platforms for all trains to a set of adjacent station platforms, thus limiting the connection time between two connected trains, i.e. the transfer time of passengers from one platform to another one.

The train delays are based on a statistical fitting procedure of the different train categories (similar to the one presented by Yuan [56]), based on the arrival and departure data recorded by ProRail at Utrecht Central in April 2008. Here we consider a set of 20 timetable perturbation instances. For each instance, all trains suffer an entrance deviation, and multiple trains have a positive delay at their entrance in the network.

Table 8 reports on the results for 20 CDR instances of the Utrecht Central test case. The instance code is as for Table 4. For this set of instances, VND is the best algorithm since the best-known solution is, on average, computed in a shorter computation time (the computation time is up to 4 s) compared to the other algorithms. VNS Basic is, on average, the second best algorithm. Overall, the improvement of VND and VNS algorithms versus both the TS algorithm and the commercial solver is a strongly reduced computation time. Specifically, CPLEX is always outperformed by VND, VNS Basic and VNS Reduced. CPLEX is useful to certify the optimality for 8 CDR instances (see the values with asterisk). However, the optimality for the other 12 CDR instances is still not certified by CPLEX after 2 h of computation.

6.4. Results on the British test case

The test bed is a mixed-traffic railway network nearby the city of London, approximately from King's Cross station to Huntingdon station, on the East Coast Main Line of The United Kingdom. Fig. 8

Table 8
Results obtained for the Second Dutch (SD) test case instances.

CDR instance	TS [13]		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)
SD_360_48.4	88	8.3	88	1.1	88	153.9	88	0.5	88	6.9	88	180.0
SD_510_48.8	95	25.1	95	0.9	95	175.7	95	26.9	95	3.7	95	168.0
SD_259_45.8	99	1.5	99	0.1	99	1.2	99	0.1	99	0.2	99	152.3
SD_253_42	90	135.4	90	2.0	90	174.7	90	2.1	90	51.4	90	255.9
SD_273_48.1	83	0.1	83	0.1	83	180	83	0.1	83	0.1	83	298.7
SD_382_48.6	51*	59.8	51*	1.2	51*	15.4	51*	0.5	51*	5.7	51*	263.5
SD_286_44.2	51*	0.1	51*	0.1	51*	2.1	51*	0.1	51*	0.1	51*	39.8
SD_502_37.2	51*	0.1	51*	0.1	51*	2.2	51*	0.1	51*	0.2	51*	155.3
SD_377_44.3	51*	158.2	51*	2.1	51*	174.3	51*	6.3	51*	27.9	51*	62.8
SD_310_29.7	51*	1.5	51*	1.0	51*	180	51*	0.6	51*	2.3	51*	148.2
SD_418_43.5	160	2.8	160	2.5	160	180	160	3.0	160	9.5	160	190.0
SD_528_34.3	232	4.0	232	4.0	232	180	232	4.0	232	4.0	232	402.5
SD_740_38.1	93	13.9	93	0.1	93	178.5	93	0.1	93	0.1	93	390.5
SD_493_32.3	66	0.2	66	0.2	66	1.2	66	0.2	66	0.2	66	37.9
SD_411_33.4	153	43.7	153	1.1	153	171.8	153	28.4	153	10.2	153	240.0
SD_1165_47.1	51*	3.1	51*	1.8	51*	179.5	51*	6.1	51*	3.2	51*	178.9
SD_493_23.4	51*	0.8	51*	0.7	51*	160.5	51*	0.4	51*	3.4	51*	43.4
SD_370_34.6	51*	30.4	51*	3.5	51*	180	51*	13.1	51*	30.3	51*	129.5
SD_675_28.6	70	12.1	70	2.1	70	175.2	70	6.0	70	113.1	70	619.5
SD_475_27.2	133	6.7	133	0.9	133	180	133	4.3	133	7.9	133	178.1
Avg results	**88.5**	25.4	**88.5**	1.3	**88.5**	132.3	**88.5**	5.1	**88.5**	14.0	**88.5**	206.7

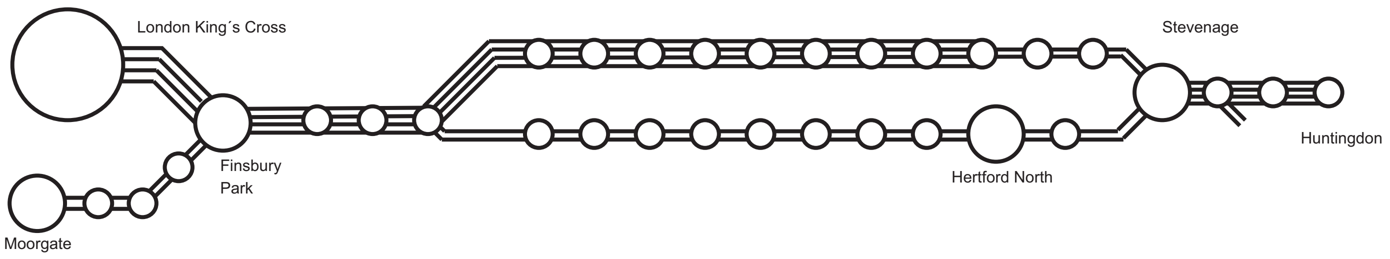

Fig. 8. British test case – the East Coast Main Line nearby London.

shows a layout of the studied network with the amount of tracks per branch, and the indication of minor/major stations. In this set of experiments the scheduler has to deal with strongly disrupted traffic situations in which some trains have speed restrictions and others are re-routed.

Table 9 presents information on the largest CDR instances tested for the British test case. For this set of CDR instances we only consider two routes per train, as provided by an industrial partner in [27].

Table 10 reports on the computational results for 20 CDR instances of the British railway network. The instance code is as for Table 4. For this set of instances, we know the optimal solution for all CDR instance, as certified by the lower bound of CPLEX. Regarding the performance of the various algorithms, VNS Basic is the best algorithm in terms of the average objective function value (as reported in bold), even if the other algorithms and the commercial solver compute a better solution for instance British_512_52.9. Furthermore, VNS Basic is often the fastest algorithm to compute the optimal solution. However, some CDR instances are solved to optimality in a shorter time by VND, VNS Reduced and TS.

6.5. Discussion on the obtained results

This section gives a brief overview of the average performance of the algorithms described in this paper.

Table 9
Characteristics of the British (B) test case instances.

Time horizon (min)	Network length (km)	Number of trains	Number of routes per train	Number of resources per train	MILP variables		
					|N|	|A|	|C|
60	80	90	2	69	5565	46,219	128

- Comparing TS and VNS Basic, the latter outperforms the former in terms of the average objective function value and also provides an average strong reduction of the time to compute the best-known solutions. A motivation is that TS mostly performs single routing changes in RCPO, while VNS Basic is based on multiple simultaneous routing changes in a combination of neighbourhood structures.
- VNS Basic is the best variable neighbourhood search algorithm in terms of the average objective function value, since it combines the local search and the shaking procedures.
- VNS Basic is better than VNS Reduced, since the former is also guided by the local search procedure, helping to intensify the search of better quality solutions in specific regions of the search space.
- Comparing VND and VNS General, the latter algorithm sometimes improves the performance of VND via the shaking procedure, that can be a profitable attempt to escape from a local optimum.

Table 10
Results obtained for the British test case instances.

CDR instance	TS [13]		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)	Value (s)	Time (s)
B_4558_113.0	2236*	61.9	2236*	0.2	2236*	176.4	2236*	0.1	2236*	0.1	2236*	11.8
B_474_48.1	108*	6.3	108*	3.2	108*	180	108*	16.2	108*	4.6	108*	75.0
B_452_49.6	379*	4.8	979	1.5	379*	179.6	379*	45.2	379*	4.4	379*	111.8
B_2577_152.4	532*	114.0	813	1.9	532*	176.4	532*	6.4	532*	2.3	532*	350.0
B_852_67.4	412*	3.1	412*	1.3	672	179.3	412*	1.8	672	0.7	412*	25.5
B_716_59.9	277*	0.1	277*	0.1	277*	180	277*	0.1	277*	0.1	277*	4.1
B_437_55.1	3151*	118.0	3151*	0.1	3151*	0.1	3151*	0.1	3151*	0.1	3151*	79.9
B_520_54.1	353*	12.4	353*	4.7	353*	180	353*	4.3	353*	1.4	353*	13.6
B_556_56.9	451*	23.3	451*	0.9	901	178.5	451*	0.6	901	1.6	451*	22.0
B_2374_77.4	3199*	68.3	3199*	1.7	3199*	180	3199*	2.4	3199*	0.5	3199*	2410.3
B_1218_77.1	491	157.0	2626	2.1	408*	174.2	408*	70.3	408*	43.6	408*	2434.1
B_561_46.7	3499*	0.1	3499*	0.1	3499*	176.2	3499*	0.1	3499*	0.1	3499*	131.9
B_2194_59.2	1426*	35.3	1426*	0.1	1426*	0.1	1426*	0.1	1426*	0.1	1426*	2489.9
B_2255_65.1	1992*	80.3	1992*	0.2	1992*	0.2	1992*	0.2	1992*	0.2	1992*	2750.8
B_487_49.8	421*	25.4	421*	12.9	421*	153.8	421*	142.0	421*	3.5	421*	1667.3
B_452_46.2	408*	34.6	3030	0.1	1579	159.9	408*	118.7	1579	2.1	408*	352.2
B_1349_98.3	597*	14.6	597*	7.9	597*	180	597*	6.1	597*	6.6	597*	1751.1
B_512_52.9	2788*	103.0	2788*	1.9	2788*	177.6	2838	0.1	2788*	0.8	2788*	14.7
B_540_50.1	1202*	110.6	1266	98.6	1202*	170.5	1202*	40.3	1202*	82.5	2244	2263.0
B_556_52.0	508*	97.8	508*	8.6	508*	179.9	508*	3.1	723	7.1	508*	493.6
Avg results	1221.5	53.6	1506.6	7.4	1311.4	149.4	**1219.9**	22.9	1322.2	8.1	1269.5	872.6

- DJ, WOCP and their combination are the best neighbourhood structures when incorporated in VNS Basic. These neighbourhood structures are new promising ingredients to solve the CDR problem.
- Different CDR instances result in rather unsimilar average trends regarding the quality of the solutions computed by the various algorithms. In general a small amount of train traffic, in terms of trains/hour, makes the CDR algorithms rapidly converge to almost the same solution. When the traffic is more dense and multiple re-routing options are available, the CDR algorithms compute more diverse solutions.
- The computational speed of AGLIBRARY is mostly depending on the algorithmic structure and configuration, but it also depends on the railway infrastructure and traffic flow characteristics.
- The complexity of the CDR instance depends on the type of the re-ordering and re-routing alternatives available for each train. For instance, a train can avoid a conflict with another train by changing the stop platform in a station area, while a train can only slightly anticipate or posticipate a conflict with another train on a corridor by performing a local re-routing. In the latter case, the train ordering is the key decision variable to solve the conflicting traffic situation. In general, the interdependence between train scheduling and routing variables plays a key role in the resolution of the CDR problem.
- The commercial solver is not able to compute a good quality solution for most of the CDR instances in a short computation time, and therefore cannot be part of a DSS for real-time train traffic control.

7. Conclusions and future research

This paper proposes fast scheduling and routing metaheuristics for real-time railway traffic management in busy networks, with particular focus on the efficient control of strong traffic disturbances (such as multiple train delays and temporarily unavailable block sections). The CDR problem is modelled via the alternative graph, that is a generalization of the disjunctive graph, and as a MILP formulation for simultaneous train scheduling and routing. To solve the CDR problem, several algorithmic innovations are considered, which relate to design of effective metaheuristics based on a problem decomposition into train scheduling and routing decisions. Variable neighbourhood search schemes (VND, Reduced VNS, Basic VNS and General VNS) are proposed based on systematic changes of a combination of neighbourhood structures.

The new metaheuristic algorithms are benchmarked against a state-of-the-art tabu search algorithm [13] and a commercial MILP solver. The evaluation is performed over multiple networks with varying traffic and infrastructure characteristics. The algorithms proposed in this paper, with various combinations of neighbourhood structures, improve the effectiveness of the previously developed CDR algorithms and outperform the commercial MILP solver. The main contributions of the variable neighbourhood search are a general significant reduction of the time required to compute good quality (sometimes proven optimal) solutions, and the computation of new best known solutions for some CDR instances.

Further research should be focused on a number of issues: the assessment of the proposed methodology to solve the CDR problem for different railway networks, traffic flows and types of demand or disturbance that could be described by some metric, in order to find an approximate relation between the instance characteristics and the expected algorithmic performance; the development and evaluation of alternative CDR heuristics, metaheuristics and exact approaches; the customization and application of the models and algorithms proposed in this paper to other transportation and logistics problems.

Acknowledgements

We acknowledge the support from Ing. Giacomo Zaninotto and Ing. Alessandro Toli. We also would like to thank the anonymous reviewers for their helpful and constructive remarks. The CDR instances are available upon request for research purposes by sending an email to the corresponding author of this paper.

Appendix

A small illustrative example is proposed to explain the basic characteristics of the model, the neighbourhoods, the neighbours and their evaluations. We consider 4 trains (A, B, C, D) that are running on the railway network of Fig. 9; those can be identified by different colors. Trains A and B cross the network from left to right, while trains C and D run in the opposite direction. Moreover, A and D are local services which stop at the stations R and Q, while the other trains have no planned stop in the network.

The network is composed by 8 resources (block sections), labelled from 1 to 8 in the top part of Fig. 9. Signals delimit the block sections. Moreover, the two stations R and Q (respectively resources 2, 7, 8; and 4, 6) are associated with additional platform resources, reported as 2R and 8R (station R); and as 4Q and 6Q (station Q). Those resources model the dwell process, for the trains that have a scheduled stop at the stations (i.e. trains A and D).

Each train has to be routed in the network and there are multiple alternative routes. Trains B and C can use any of the resources 2, 7, 8 in the area of station R, and any of the resources 4 and 6 in the area of station Q. As any route in the first station area can be combined with any route in the second station area, a total of 6 routing options are available for trains B and C. Due to the need to stop at station R, trains A and D can use only resources 2 and 8. They cannot use resource 7, since there is no platform available in that resource. So trains A and D can only be routed between resources 2 and 8, and between resources 4 and 6, for a total of 4 routing options. The full set of train routes, including the default ones, is reported schematically on the bottom side of Fig. 9. The default routes, reported in dotted lines, are as follows. Between brackets is the running time (dwell time for the station stop): **A1:** 1 (1), 8(2), 8R(1), 3(3), 4(2), 4Q(1), 5(2); **B1:** 1(1), 2(2), 3(3), 4(2), 5(2); **C1:** 5(2), 6(3), 3(5), 7(3), 1(2); **D1:** 5(2), 6(3), 6Q(1), 3(5), 8(3), 8R (1), 1(2).

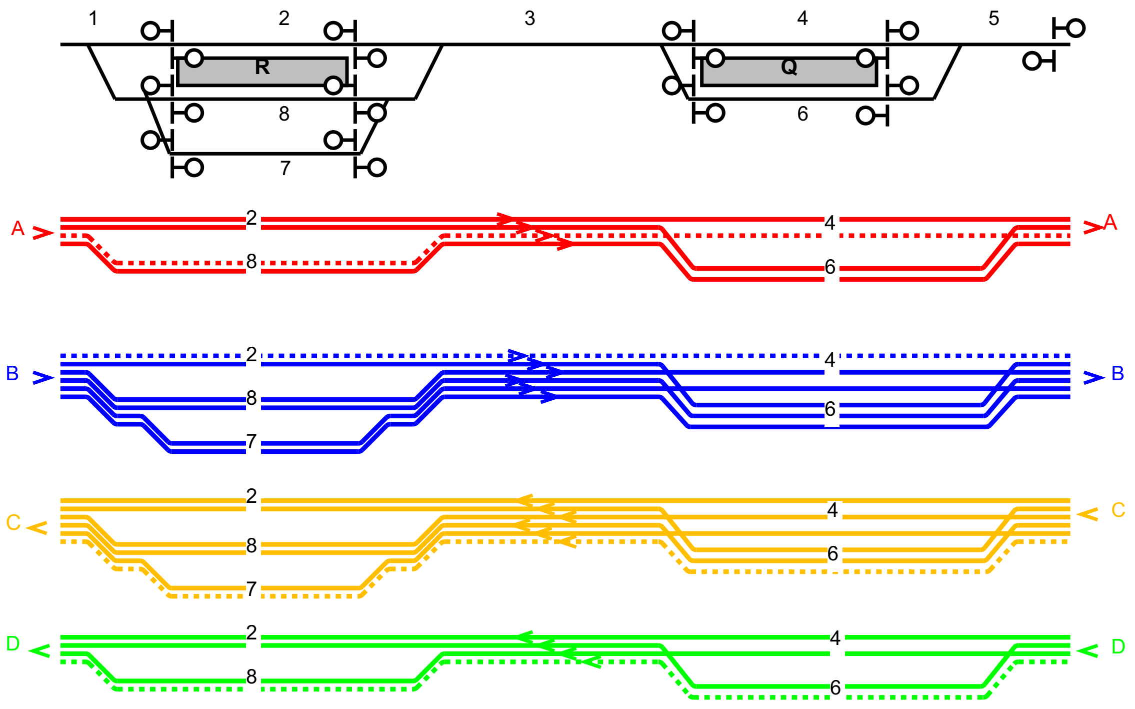

Fig. 9. Railway network and the routing alternatives for the four trains.

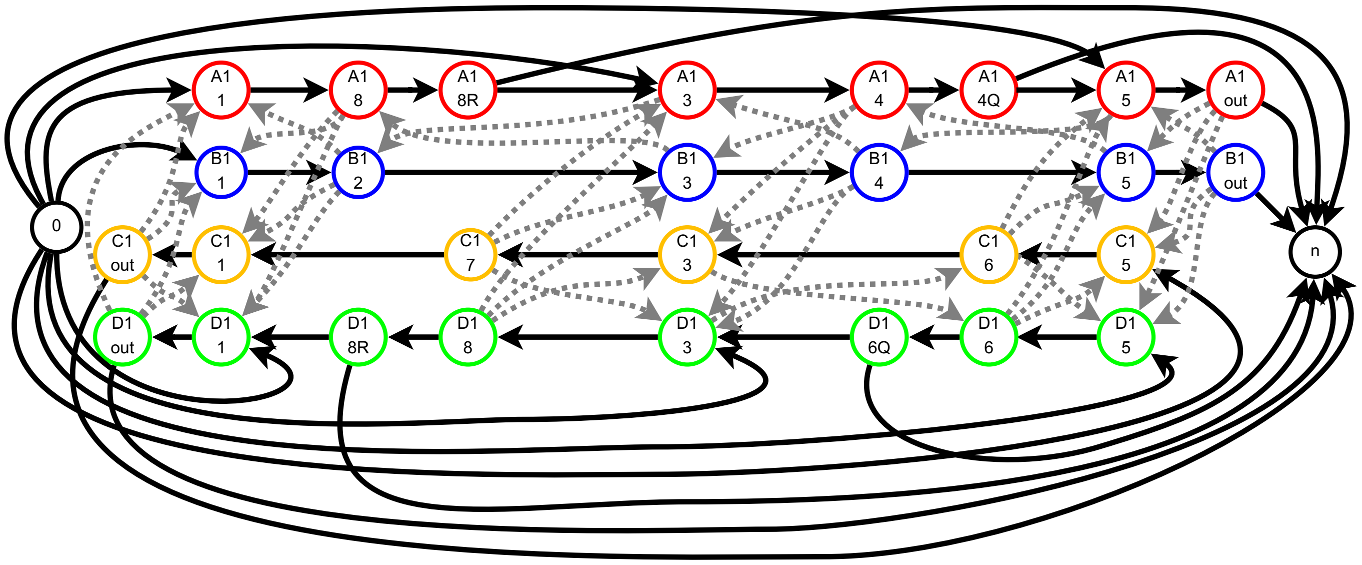

Fig. 10. Alternative graph of the example with default train routes.

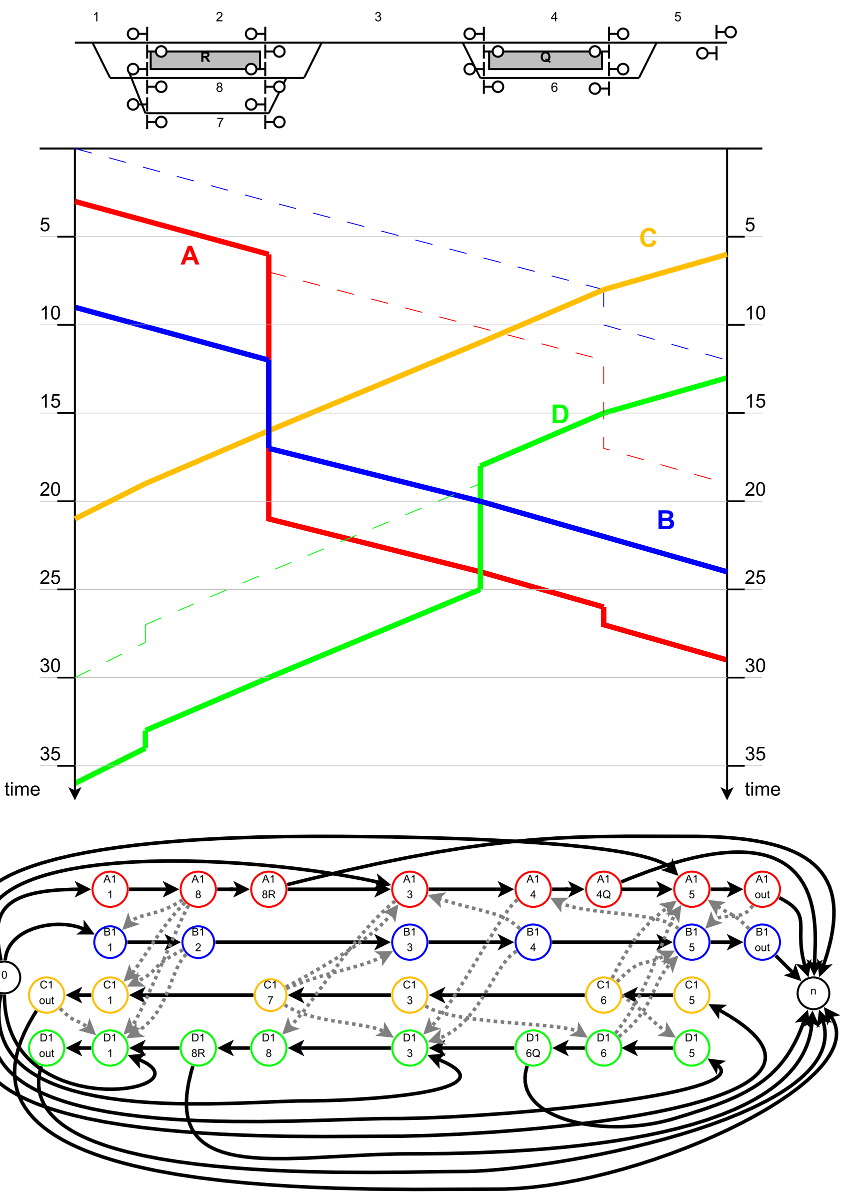

Fig. 11. A CDR solution shown as a time–distance plot, and the associated alternative graph.

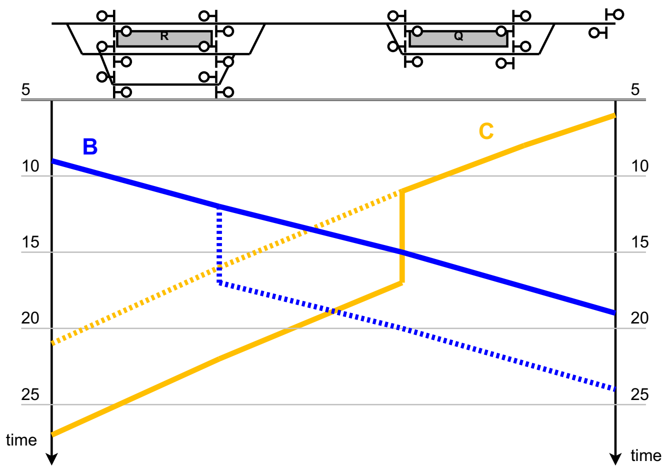

Fig. 12. Free-net waiting operation – ranking computation.

Fig. 10 shows the alternative graph model of the CDR problem with the default routes. Each train results in a sequence of nodes (operations) reported horizontally. Each node (e.g. A1-8) is identified by the name of the train and routing (e.g. A1) plus the resource over which the operation is performed (e.g. 8).

In the alternative graph model, the fixed arcs are reported in solid color, while the alternative arcs are reported in dotted grey. The latter arcs are given for each shared resource, namely resources 1, 3, 5 for all trains; resource 8 for trains A, D; resource 4 for trains A, B; resource 6 for trains C, D. Moreover, a number of fixed arcs exit node 0, in relation with the entrance time (including the initial delay) in the network and the minimum departure time from the station platforms. Analogously, a number of fixed arcs enter node *n*, in relation with the exit time from the network, and the arrival time at the station platforms. For this illustrative example, the headway time between consecutive trains is always equal to 1 time unit.

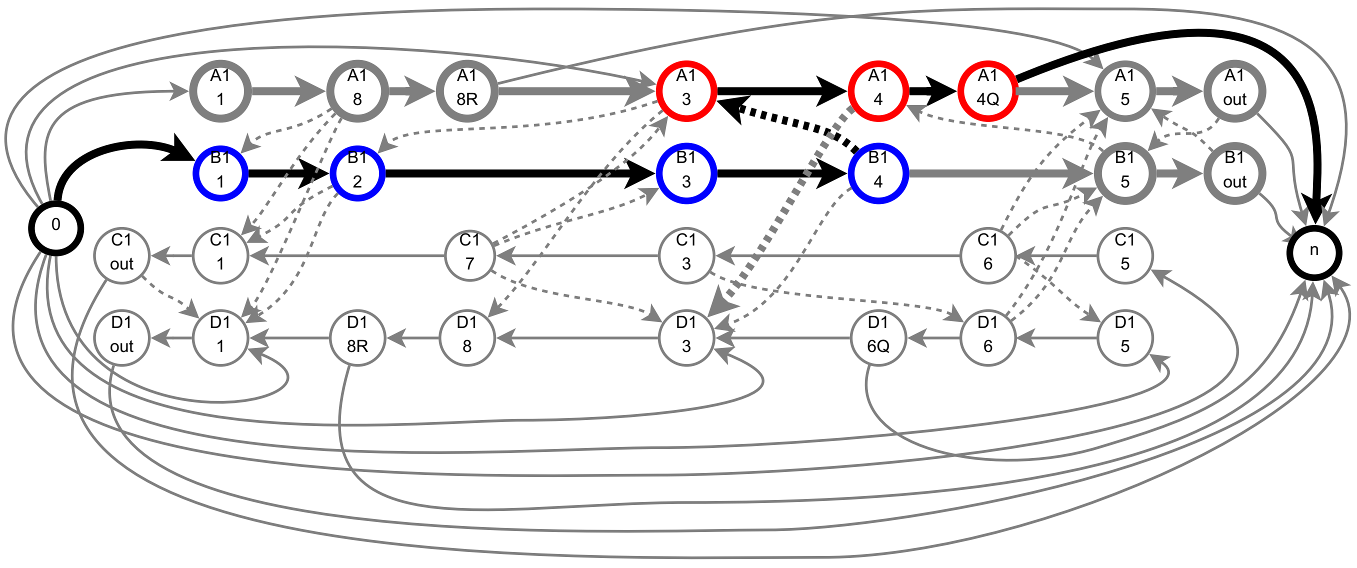

Fig. 13. Ramified critical path.

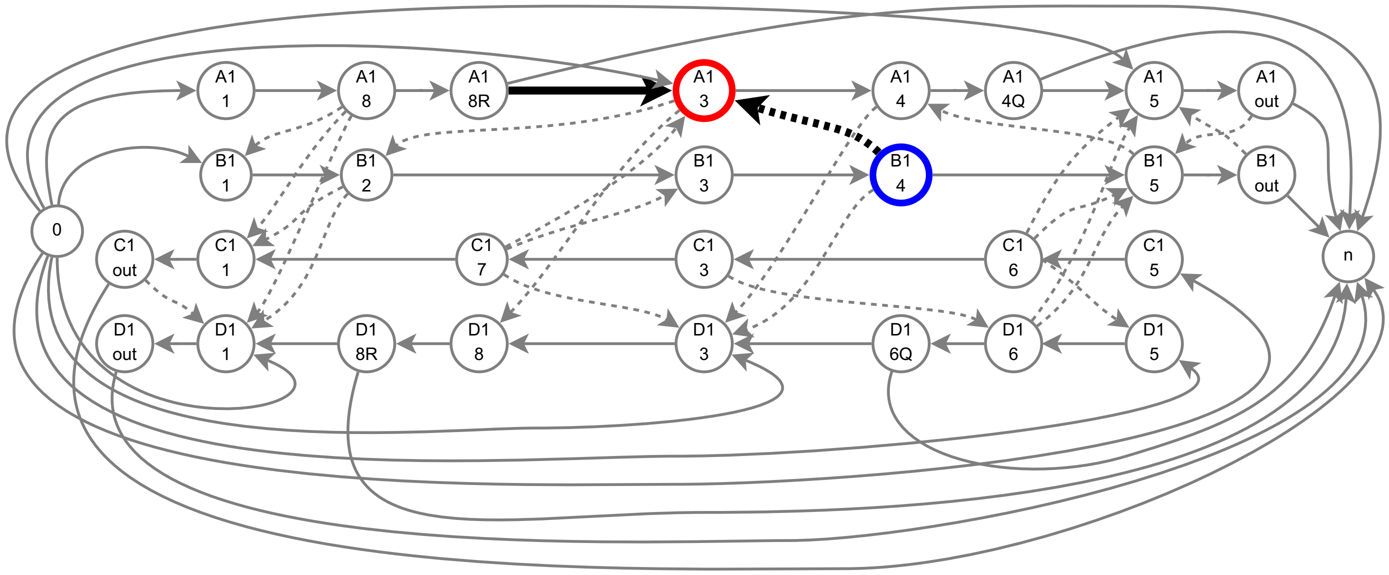

Fig. 14. Waiting operations on the critical path.

A solution to the CDR problem is reported in Fig. 11 as a time distance path (top) and alternative graph (bottom). For the time distance graph, space is along the x-axis, time (increasing downwards) is along the y-axis. In this solution, train C goes first over resource 3, followed by trains B, A and D. The minimum headway time (one time unit) over the sections constrains the departure of train A from station R, as well as the departure time (in the opposite direction) of train D. Those trains are assumed to be waiting at station R until the resource 3 becomes available. In the dotted lines of Fig. 11, the original plan is reported. The train B suffers an initial delay of 9 time units, while the other trains have no initial delays. In Fig. 11, the scheduled arrival times of the trains are: **A1:** 6 at station R (no delay); 12 at station Q (the delay is 14, as the realised arrival time is 26); 19 at exit of the network (the delay is 10, as the realised exit time is 29). **B1:** 12 at the exit of the network (the realised exit time is 24; the consecutive delay is 3). **C1:** 21 at the exit of the network (no delay); **D1:** 18 at station Q (no delay); 27 at station R (the delay is 6, as the realised arrival time is 33); 30 at the exit of network (the delay is 6 as the realised exit time is 36).

In the alternative graph model of the CDR solution reported at the bottom of Fig. 11, exactly one arc from each alternative pair has been selected, i.e. a train ordering decision has been taken for each potential conflict. For instance, the chosen train order over resource 3 (i.e. C–B–A–D) results in the following arc selection: (C1-7, A1-3); (C1-7, B1-3); (C1-7, D1-3); (B1-4, A1-3); (B1-4, D1-3); (A1-4, D1-3). This is a scheduling solution with fixed routes, with a maximum consecutive delay of 14.

We next discuss the four neighbourhoods explained in Section 5.3 in terms of the train and route rankings.

Free-Net Waiting Operations Jobs neighbourhood $\mathcal{N}_{FNWJ}$: The $\mathcal{N}_{FNWJ}$ ranking is based on the sum of the consecutive delays collected at each waiting node in the graph of the incumbent routing solution in which all alternative arcs are unselected (i.e. free-net traffic situation) but the one generating the waiting node. For example, let's consider the pair ((B1-4, C1-3), (C1-7, B1-3)) that concerns the order of trains B and C over resource 3. We next refer to the time distance graph of Fig. 12.

If the arc (B1-4, C1-3) is selected, the order reported in solid style in Fig. 12 lines is implemented. Train C arrives at the end of

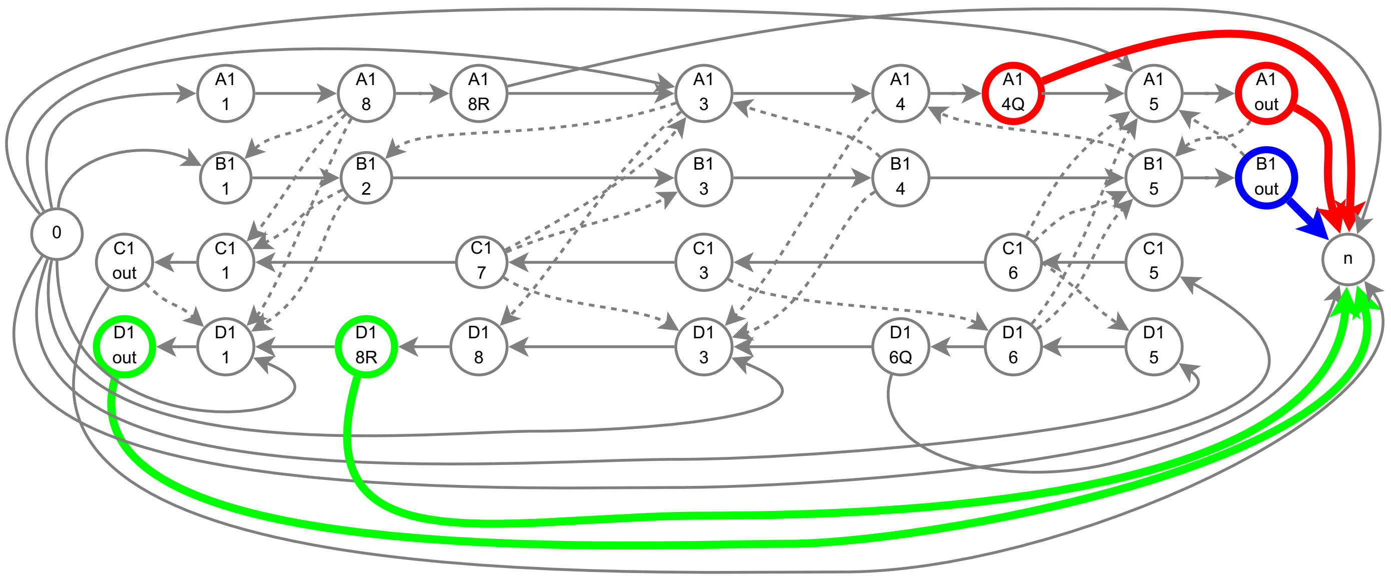

Fig. 15. Delayed jobs.

resource 3 at time 11, and has to wait for the exit of train B from resource 3, that happens at time 16, plus 1 time unit of headway time making it 17. This results in a delay to train C of 6 time units. Train B suffers no additional delay. The max consecutive delay is thus 6.

If the other arc of the pair (C1-7, B1-3) is selected, the other train order is considered, as reported via the dotted lines in Fig. 12. Train B has to wait from time 12, for the exit of train C from resource 3, that happens at time 16, plus 1 time unit of headway time. Train B then exits the network at time 24, with a consecutive delay of 3 time units. Differently, train C can enter resource 3 at time 11 without additional delay, and exit the network without any delay. The maximum consecutive delay is thus 3.

The $\mathcal{N}_{FNWJ}$ ranking computes the score as the minimum consecutive delay generated on the two waiting nodes related to each alternative pair, i.e. min(3, 6)=3 is assigned to B1 and C1 due to the alternative pair ((B1-4, C1-3), (C1-7, B1-3)). The complete score of each train is computed by summing up the minimum consecutive delay generated by each alternative pair. In this example, no other alternative pair generates any additional score, i.e. if only two trains at a time are considered in the network (free-net situation) no other conflict arises.

The $\mathcal{N}_{FNWJ}$ ranking values are thus 3 for B1 and C1, 0 for the other trains.

Ramified Critical Path Operations neighbourhood $\mathcal{N}_{RCPO}$: For the solution of Fig. 11, the critical path is highlighted in Fig. 13 in color and black. The operations on the critical path are as follows: B1-1, B1-2, B1-3, B1-4, A1-3, A1-4, A1-4Q. The ramified critical path is an extension of the critical path including also the waiting operations preceding or following the ones on the critical path. In this case the ramified critical path corresponds to all operations of trains A1 and B1, highlighted in the thicker lines (grey or black). In $\mathcal{N}_{RCPO}$, the ranking values for each train are computed as the maximum value $l^{S'(F^R)}(0,krp)+l^{S'(F^R)}(krp,n)\ \forall(krp)$ in the ramified critical path of the graph of the incumbent solution. This value corresponds exactly to the makespan, i.e. the maximum consecutive delay of 14, for all nodes in the critical path. For each train route, the maximum of its ranking values is the ranking score. The score of A1 and B1 is 14, the score for C1 and D1 is 0.

Waiting Operations Critical Path neighbourhood $\mathcal{N}_{WOCP}$: The $\mathcal{N}_{WOCP}$ ranking is based on the sum of the consecutive delays collected on the waiting operations on the critical path of the incumbent solution. We refer to Fig. 14 for a graphical illustration. In this example, there is a single waiting operation (A1-3) in which a consecutive delay of value 14 time units is collected for A1. The consecutive delay (14) of A1-3 is computed as follows: the weight of the path $l^{S'(F^R)}(0,B1-4)=20$, corresponding to the time needed by train B1 to exit B1-3 (and to enter B1-4); plus the weight $w^F_{(B1-4),(A1-3)}=1$, corresponding to the headway time; minus the weight of $l^{S'(F^R)}(0,B1-3)=7$, corresponding to the time at which train A1 would be able to enter A1-3, without any other potential conflict. This yields a score of 14. There is no other waiting operation on the critical path. The ranking of $\mathcal{N}_{WOCP}$ is thus 14 for train A1, 0 for the others.

Delayed Jobs neighbourhood $\mathcal{N}_{DJ}$: The $\mathcal{N}_{DJ}$ ranking is based on the maximum consecutive delays at some relevant locations for each train. Fig. 15 highlights the fixed arcs going into node n, in which some consecutive delays are experienced. Train C1 is not delayed and its score is thus 0. The other trains have the following consecutive delays: A1-8R: 0 A1-4Q: 14; A1-out: 10; B1-out: 3; D1-4Q: 0; D1-8R:6; D1-out: 6. For each job, the largest delay is taken, which corresponds to a rank of 14 for train A1, 3 for B1, 0 for C1, 6 for D1. Fig. 15 highlights the arcs going into node n in which consecutive delays are collected.

References

[1] Ahuja R, Cunha C, Şahin G. Network models in railroad planning and scheduling. In: Greenberg H, Smith J, editors. Tutorials in operations research, vol. 1; 2005. p. 54–101.
[2] Almodóvar M, García-Ródenas R. On-line reschedule optimization for passenger railways in case of emergencies. Comput Oper Res 2013;40(3):725–36.
[3] Cadarso L, Marín Á. Improving robustness of rolling stock circulations in rapid transit networks. Comput Oper Res 2014;51:146–59.
[4] Cadarso L, Marín Á, Maróti G. Recovery of disruptions in rapid transit networks. Transp Res Part E 2013;53:15–33.
[5] Cacchiani V, Huisman D, Kidd M, Kroon L, Toth P, Veelenturf L, et al. An overview of recovery models and algorithms for real-time railway rescheduling. Transp Res Part B 2014;63:15–37.
[6] Cai X, Goh CJ. A fast heuristic for the train scheduling problem. Comput Oper Res 1994;21(5):499–510.
[7] Caimi G, Fuchsberger M, Laumanns M, Lüthi M. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. Comput Oper Res 2012;39(11):2578–93.
[8] Cheng Y. Hybrid simulation for resolving resource conflicts in train traffic rescheduling. Comput Ind 1998;35(3):233–46.
[9] Chiu C, Chou C, Lee J, Leung H, Leung Y. A constraint-based interactive train rescheduling tool. Constraints 2002;7:167–98.

[10] Cordeau JF, Toth P, Vigo D. A survey of optimization models for train routing and scheduling. Transp Sci 1998;32(4):380–404.
[11] Corman F, D'Ariano A, Hansen IA, Pacciarelli D. Optimal multi-class rescheduling of railway traffic. J Rail Transp Plan Manag 2011;1(1):14–24.
[12] Corman F, D'Ariano A, Pacciarelli D, Pranzo M. Evaluation of green wave policy in real-time railway traffic management. Transp Res Part C 2009;17:607–16.
[13] Corman F, D'Ariano A, Pacciarelli D, Pranzo M. A tabu search algorithm for rerouting trains during rail operations. Transp Res Part B 2010;44(1):175–92.
[14] Corman F, D'Ariano A, Pacciarelli D, Pranzo M. Optimal inter-area coordination of train rescheduling decisions. Transp Res Part E 2012;48(1):71–88.
[15] Corman F, D'Ariano A, Pacciarelli D, Pranzo M. Bi-objective conflict detection and resolution in railway traffic management. Transp Res Part C 2012;20(1):79–94.
[16] Corman F, D'Ariano A, Pacciarelli D, Pranzo M. Dispatching and coordination in multi-area railway traffic management. Comput Oper Res 2014;44:146–60.
[17] Corman F, D'Ariano A, Pranzo M, Hansen IA. Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. Transp Plan Technol 2011;34(4):341–62.
[18] Corman F, Goverde RMP, D'Ariano A. Rescheduling dense train traffic over complex station interlocking areas. In: Ahuja RK, Moehring R, Zaroliagis C, editors, Robust and online large-scale optimization. Lecture notes in computer science, vol. 5868. Springer; 2009. p. 369–86.
[19] Corman F, Pacciarelli D, D'Ariano A, Samà M. Railway traffic rescheduling taking into account minimization of passengers discomfort. In: Corman F, Voss S, Negenborn RR, editors, Computational logistics. Lecture notes in computer science, vol. 9335. Springer; 2015. p. 602–16.
[20] Corman F, Quaglietta E. Closing the loop in railway traffic control: framework design and impacts on operations. Transp Res C 2015;54(1):15–39.
[21] Corman F, Xin J, Toli A, Negenborn RR, D'Ariano A, Samà M, et al. Optimizing hybrid operations at large-scale automated container terminals. In: Proceedings of the 4th international conference on models and technology for intelligent transportation systems. Budapest, Hungary; 2015.
[22] D'Ariano A. Innovative decision support system for railway traffic control. IEEE Intell Transp Syst Mag 2009;1(4):8–16.
[23] D'Ariano A, Corman F, Pacciarelli D, Pranzo M. Reordering and local rerouting strategies to manage train traffic in real-time. Transp Sci 2008;42(4):405–19.
[24] D'Ariano A, Pacciarelli D, Pistelli M, Pranzo M. Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area. Networks 2015;65(3):212–27.
[25] D'Ariano A, Pacciarelli D, Pranzo M. A branch and bound algorithm for scheduling trains in a railway network. Eur J Oper Res 2007;183(2):643–57.
[26] D'Ariano A, Pistelli M, Pacciarelli D. Aircraft retiming and rerouting in vicinity of airports. IET Intell Transp Syst J 2012;6(4):433–43.
[27] D'Ariano A, Samà M, D'Ariano P, Pacciarelli D. Evaluating the applicability of advanced techniques for practical real-time train scheduling. Transp Res Proc 2014;3:279–88.
[28] Dollevoet T, Corman F, D'Ariano A, Huisman D. An iterative optimization framework for delay management and train scheduling. Flex Serv Manuf J 2014;26(4):490–515.
[29] Fang W, Yang S, Yao X. A survey on problem models and solution approaches to rescheduling in railway networks. IEEE Trans Intell Transp Syst 2015;16(6):2997–3016.
[30] Hansen P, Mladenović N, Moreno Pérez JA. Variable neighbourhood search: methods and applications. 4OR – Q J Oper Res 2008;6(4):319–60.
[31] Hansen IA, Pachl J. Railway timetabling & operations. Hamburg, Germany: Eurailpress; 2014.
[32] Kecman P, Corman F, D'Ariano A, Goverde RMP. Rescheduling models for railway traffic management in large-scale networks. Public Trans: Plan Oper 2013;5(1–2):95–123.
[33] Kroon LG, Maróti G, Nielsen LK. Rescheduling of railway rolling stock with dynamic passenger flows. Transp Sci 2014;49(2):165–84.
[34] Lamorgese L, Mannino C. The track formulation for the train dispatching problem. Electron Notes Discret Math 2013;41:559–66.
[35] Liu SQ, Kozan E. Scheduling trains as a blocking parallel-machine job shop scheduling problem. Comput Oper Res 2009;36:2840–52.
[36] Lusby RM, Larsen J, Ehrgott M, Ryan DM. A set packing inspired method for real-time junction train routing. Comput Oper Res 2013;40(3):713–24.
[37] Mannino C, Mascis A. Optimal real-time traffic control in metro stations. Oper Res 2009;57(4):1026–39.
[38] Mascis A, Pacciarelli D. Job shop scheduling with blocking and no-wait constraints. Eur J Oper Res 2002;143(3):498–517.
[39] Mazzarello M, Ottaviani E. A traffic management system for real-time traffic optimization in railways. Transp Res Part B 2007;41(2):246–74.
[40] Meng L, Zhou X. Robust single-track train dispatching model under a dynamic and stochastic environment: a scenario-based rolling horizon solution approach. Transp Res Part B 2011;45(7):1080–102.
[41] Moreno Pérez JA, Moreno-Vega JM, Rodríguez Martín I. Variable neighbourhood tabu search and its application to the median cycle problem. Eur J Oper Res 2003;151(2):365–78.
[42] Pacciarelli D, Pranzo M. Production scheduling in a steelmaking-continuous casting plant. Comput Chem Eng 2004;28(12):2823–35.
[43] Pellegrini P, Marlière G, Rodriguez J. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. Transp Res Part B 2014;59:58–80.
[44] Pellegrini P, Rodriguez J. Single European sky and single European railway area: a system level analysis of air and rail transportation. Transp Res Part A 2013;57(1):64–86.
[45] Pranzo M, Meloni C, Pacciarelli D. A new class of greedy heuristics for job shop scheduling problems. Lect Notes Comput Sci 2003;2647:223–36.
[46] Rodriguez J. A constraint programming model for real-time train scheduling at junctions. Transp Res Part B 2007;41(2):231–45.
[47] Şahin İ. Railway traffic control and train scheduling based on inter-train conflict management. Transp Res Part B 1999;33(7):511–34.
[48] Samà M, D'Ariano A, D'Ariano P, Pacciarelli D. Optimal aircraft scheduling and routing at a terminal control area during disturbances. Transp Res Part C 2014;47(1):61–85.
[49] Samà M, D'Ariano A, Pacciarelli D. Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. Transp Res Part E 2013;60(1):140–55.
[50] Szpigel B. Optimal train scheduling on a single track railway. In: Ross M, editor. Operational Research, vol. 72. The Netherlands: Amsterdam; 1973. p. 343–52.
[51] Törnquist J, Persson JA. N-tracked railway traffic re-scheduling during disturbances. Transp Res Part B 2007;41(3):342–62.
[52] Törnquist Krasemann J. Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. Transp Res Part C 2012;20(1):62–78.
[53] Wegele S, Schnieder E. Automated dispatching of train operations using genetic algorithms. Computers in railways IX. Southampton, UK: WIT Press; 2004. p. 775–84.
[54] Wegele S, Slovák R, Schnieder E. Real-time decision support for optimal dispatching of train operation. In: Hansen IA, Radtke A, Pachl J, Wendler E, editors. Proceedings of the 2nd international conference on railway operations modelling and analysis, Hannover, Germany; 2007.
[55] Xin J, Negenborn RR, Corman F, Lodewijks G. Control of interacting machines in automated container terminals using a sequential planning approach for collision avoidance. Transp Res Part C 2015;60(1):377–96.
[56] Yuan J. Stochastic modelling of train delays and delay propagation in stations [Ph.D. thesis]. TRAIL thesis series T2006/6. Delft University of Technology, Delft, The Netherlands; 2006.