Public Transp (2010) 2: 219–247
DOI 10.1007/s12469-010-0032-7

ORIGINAL PAPER

Centralized versus distributed systems to reschedule trains in two dispatching areas

Francesco Corman · Andrea D'Ariano · Dario Pacciarelli · Marco Pranzo

Published online: 10 November 2010

Abstract Railway dispatchers are in charge of rescheduling trains during operations in order to limit propagation of disturbances occurring in real-time. To help the dispatchers in such task, an advanced decision support system, ROMA (Railway traffic Optimization by Means of Alternative graphs), has been recently implemented to optimize railway traffic within a single dispatching area. This paper presents a novel distributed optimization system to control trains running in a Dutch railway network that is divided into two complex dispatching areas with dense traffic, each one controlled by a single dispatcher with the support of a local ROMA. A coordination level is introduced in order to manage the interaction among the two local ROMAs. An extensive computational assessment of the centralized and distributed systems is performed by using simple and advanced train scheduling algorithms, including dispatching rules adopted during operations. The effectiveness of the distributed system

F. Corman (✉) · A. D'Ariano
Department of Transport and Planning, Delft University of Technology, Stevinweg 1, 2600 GA Delft, The Netherlands
e-mail: f.corman@tudelft.nl

A. D'Ariano
e-mail: a.dariano@tudelft.nl

A. D'Ariano · D. Pacciarelli
Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, via della Vasca Navale 79, 00146 Rome, Italy

A. D'Ariano
e-mail: a.dariano@dia.uniroma3.it

D. Pacciarelli
e-mail: pacciarelli@dia.uniroma3.it

M. Pranzo
Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Siena, via Roma 56, 53100 Siena, Italy
e-mail: pranzo@dii.unisi.it

is shown in terms of computation time and delay minimization for practical statistical entrance delay distributions and in presence of an increasing number of blocked platforms in the main station area.

Keywords Railway rescheduling · Traffic optimization · Schedule coordination

1 Introduction

The traffic control of nation-wide railway networks is usually managed by a set of regional traffic control centers. For instance, the control of the Dutch railway network is subdivided in one main center in Utrecht, four regional centers (Amsterdam, Eindhoven, Rotterdam and Zwolle) and thirteen traffic control offices, each controlling several dispatching areas.

In each dispatching area there is a dispatcher who receives real-time information on the route, current location and speed of each train, and the status of each track in the railway system. The dispatcher analyzes the data (checking if the timetable is coherent with the current trains' positions and speeds), calculates whether and where conflicts are going to occur and solves them on the basis of experience and rules. Possible control actions include changing dwell times at scheduled stops or train orders at junctions, stations and passing points. Other control actions involve major modifications such as route change or cancellation. The main goal of dispatchers is the minimization of train delays. Note that dispatchers also take into account effects on the passengers, especially when those do not conflict with the main goal.

The traffic control is often hierarchically organized in at least two decision levels. At the lower level there are line dispatchers, who control dispatching areas with a local view of the traffic flow. At the higher level the network dispatchers are responsible for the coordination of the rescheduling decisions taken by several line dispatchers with a global overview of the traffic flow.

This paper deals with the development of Decision Support Systems (DSSs) for real-time railway traffic management of railway networks composed by multiple dispatching areas. The problem of coordinating the decisions taken by dispatchers is quite underinvestigated in the literature on railway traffic management and, to the best of our knowledge, this paper is one of first attempts to deal with this problem. Therefore, we limit our investigation to the case of two dispatching areas coordinated by a network dispatcher.

DSSs are being developed by railway companies and research institutions to support the rescheduling process. Such automated systems are mostly based on centralized procedures that must deliver viable solutions in a short computation time. However, real-time train rescheduling is a difficult NP-hard problem (Corman et al. 2010) and the best solvers available are able to solve to optimality instances at the level of a single dispatching area. Effectively managing train traffic in a network with multiple areas is still a challenging problem, since the computation time of existing exact rescheduling algorithms may increase dramatically with the size of the network.

The computational complexity of the train scheduling problem can be limited to an acceptable level by decomposition, since local decisions are taken in a parallel fashion

and independently from each other. However, little research addressed the problem of automatically coordinating independent DSSs, each controlling a single dispatching area. The main issue of a distributed approach is therefore the coordination between the solutions of the sub-problems, since the aggregated solution must be globally feasible (see, e.g., Pacciarelli 2003; Tsuruta et al. 1999).

Global feasibility requires that a number of constraints at the borders between dispatching areas is satisfied. The exit time of a train from an area must be compatible to the entrance time in the subsequent area. Train orders and routes at the border should be consistent and locally feasible solutions should not cause deadlock situations from a global perspective. Such constraints cannot be satisfied when solving the sub-problems individually, since each sub-problem has a myopic view of the entire process. Coordination action is therefore necessary and consists either in the modification of local solutions or in the addition of specific constraints to each sub-problem, so that the local solution are forced to be globally feasible.

This paper proposes a distributed framework for railway traffic control in a network of two areas. The centralized decision support system, ROMA (Railway traffic Optimization by Means of Alternative graphs) (D'Ariano 2008; D'Ariano et al. 2008; D'Ariano and Pranzo 2009) is integrated into the distributed framework. Each dispatching area is managed by a local ROMA while the problem of coordinating adjacent areas is solved with a distributed approach. Specifically, the coordination procedure alternates a solution step, in which each local ROMA produces a solution for its local area, with a coordination step. In the latter step, each local ROMA receives information on the traffic flow at the border between areas and uses it to produce a solution compliant with that of the other area. In order to exploit the potential of distributed optimization, concurrent and parallel execution is supported by a standard communication protocol (Message Passing Interface Forum 1994).

Extensive computational experiments are carried out in order to compare the performance of centralized and distributed approaches on a practical test case composed by two dispatching areas in The Netherlands. The network is a complicated and densely occupied network around Utrecht Central station, with an hourly timetable of about 80 trains per hour. Different algorithms are compared to solve the line dispatcher problem, including an exact optimization algorithm and two scheduling rules frequently used in the railway practice. Instances include multiple delayed trains and disruptions with an increasing number of blocked platforms in order to assess the performance of the centralized and distributed systems when the complexity of the instances increases. The solution quality is assessed in terms of maximum and average train delays. The computation time of both approaches is also investigated.

The main contributions of this paper are fourfold:

- A methodology is given to check the global feasibility of local schedules based on aggregated information;
- A schedule coordination procedure is introduced and tested;
- The two-level coordinated traffic control is assessed for the first time on a real network composed of two dispatching areas with dense traffic;
- A systematic study of the distributed and centralized approaches is provided for increasing traffic disturbances.

The paper outline is the following. Section 2 defines the railway terminology used throughout the paper. Section 3 provides an overview of the existing literature on train rescheduling. Section 4 briefly describes the centralized system and the mathematical model and procedures for train scheduling. Section 5 introduces the distributed system, including models and algorithms for train schedule coordination. Section 6 reports on the computational results on a real-world test case from the Dutch railway network, and deals with a variety of disturbed traffic situations. Section 7 concludes the paper and gives directions for further research.

2 Problem definitions

In its basic form a railway network is composed of stations, links and block sections separated by signals. Signals control the train traffic on the routes and are located before every junction as well as along the lines and inside the stations. A block section is a track segment between two main signals and, for safety reasons, may host at most one train at a time. Signals, interlocking and Automatic Train Protection systems (ATP) control the train traffic by imposing a minimum safety separation between trains, setting up conflict-free routes and enforcing speed restrictions on running trains.

The main characteristics of most railway signaling systems is the fixed block signaling system. A train may have movement authority to enter a block section only after the train ahead has completely left it and the ATP system releases the block section. In this paper we consider the Dutch three-aspect fixed block signaling system, in which a signal aspect may be red, yellow or green. However, the discussion holds for most fixed block signaling systems. A red signal aspect means that the subsequent block section is either out of service or occupied by another train, a yellow signal aspect means that the subsequent block section is empty, but the following block section is still occupied by another train, and a green signal aspect indicates that the next two block sections are empty. A train is allowed to enter the next block section if the signal aspect is either green or yellow, but the latter requires deceleration and stop before the next signal if this remains red. Descriptions of railway signaling systems and traffic control regulations can be found, e.g., in Goddard (2006).

The passage of a train through a particular block section is called an *operation*. A *route* of a train is a sequence of operations to be processed during a *service* (train run). At any time a train route is feasible if all its block sections are available for traffic. The *timing* of a route specifies the starting time t_i of each operation in the route. Each operation requires a traveling time, called *running time*. The running time is known in advance since all trains travel at their scheduled speed, which usually contains some margins to recover from small delays.

The running time of a train on a block section starts when its head (the first axle) enters the block section. Safety regulations impose a minimum distance separation among the trains, which translates into a minimum *setup time* (time headway) between the exit of a train from a block section and the entrance of the subsequent train into the same block section. This time includes the interval 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

cover the sighting distance (see, e.g., Hansen and Pachl 2008; Nie and Hansen 2005; Pachl 2002).

The timetable describes the movement of all trains circulating in the network by specifying, for each train, the planned arrival/passing times at a set of relevant points along its route (e.g. stations, junctions, and the exit point of the network). 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*. Additional practical constraints related to passenger satisfaction can be included, such as minimum transfer times between connected train services.

Unexpected events occurring during operations may cause delays which make the timetable infeasible. The delay may propagate causing a domino effect of increasing disturbances. We define an *entrance perturbation* as a set of delayed trains at the entrance in a dispatching area, due to the propagation of delays from previous dispatching areas. An *infrastructure disruption* is the unavailability of one or more block sections, which causes alterations in the train travel times and routes. Running time prolongation may occur because of headway conflicts between consecutive trains or technical failures. Route changes are due to some block section being unavailable for a certain amount of time and dwell time perturbations are due to traffic delays at stations.

Real-time railway traffic management copes with the temporary infeasibility by adjusting the timetable of each train, in terms of routing and timing, and/or by re-sequencing the trains at the entrance of each merging/crossing point. The railway traffic is predicted over a given time horizon. The task of dispatchers is to regulate traffic with the main objective of minimizing train delays in such a way that the new schedule is compliant with railway rules and with the actual position of each train. The latter information enables the computation of the *release time* of each train with respect to the starting time t_0 of traffic prediction, which is the expected time at which each train enters its first block section in the area under study.

The *total delay* is the positive difference between the estimated train arrival time and the scheduled time at a relevant point in the network, and can be divided into two parts. The *initial delay* is caused by original failures and disturbances and can only be recovered by exploiting available running time reserves, i.e., with trains traveling at maximum speed. The *consecutive delays* are the additional delays generated by the dispatching measures taken in response to initial delays.

A potential *conflict* occurs when two or more trains claim the same block section simultaneously, and a decision on the train ordering has to be taken. A set of trains causes a *deadlock* when each train in the set claims a block section ahead which is not available, due either to a disruption or to the occupation/reservation for another train in the set.

3 Overview of the related literature

This section gives an overview on the literature on train rescheduling. In general, automated rescheduling systems are designed to support traffic controllers in the management of railway traffic in a given railway area and within a short time limit of

Table 1 Recent contributions on centralized train rescheduling

Paper	Approach	Size	Solution method
Adenso-Díaz et al. (1999)	MA	DA	Local heuristic
D'Ariano et al. (2007a)	MI	DA	Heuristics + branch & bound
Hirai et al. (2006)	MA	DL	Local heuristic
Jacobs (2004)	MI	N	Simulation + priority rules
Ping et al. (2001)	MA	DL	Genetic algorithms
Rodriguez (2007)	MI	J	Constraint programming + branch & bound
Şahin (1999)	MI	SL	Look-ahead heuristic
Takagi et al. (2006)	MI	J	Simulation + genetic algorithms
Törnquist (2006)	MA	N	Mixed integer linear programming
Wegele et al. (2007)	MI	N	Simulation + genetic algorithms

computation. An important objective is the minimization of train delays at stations as well as at the exit from the area. Other research directions focus on the delay minimization for passengers and goods (for an example of this area of research, see e.g. Schöbel 2009). Several factors influence the solution quality, including the level of detail considered to formulate the problem, the size and complexity of the studied area, the density of traffic, the type of disturbance and the length of the time horizon of traffic prediction. These factors play an important role and need to be carefully chosen in accordance with the chosen solution approach.

We classify the variety of approaches presented in the literature discussion in two groups: centralized and distributed systems. The first group has a single core in charge of deciding the rescheduling actions while the second group is organized in entities (agents, modules or subsystems) that take decisions and need to be coordinated.

Table 1 reports a list of recent contributions on centralized train rescheduling. We analyze three important factors: the level of detail (MA = macroscopic or MI = microscopic), the characteristics of the studied area (SL = Single-track Line, DL = Double-track Line, J = Junction, DA = Dispatching Area or N = Network with multiple dispatching areas) and the solution methodology. All the studied approaches consider realistic railway networks.

In general, macroscopic approaches model train movements and headway times without considering explicitly all details of the safety system. A train run on the track between two stations is described by a running time on the overall track and a minimum headway between successive trains. Differently, microscopic approaches control trains at the level of block signals and offer more detailed information about train movements and actual state of the signaling system along the tracks. Detailed information is used regarding track layout, switches and signals in order to compute accurate running and headway times. Clearly, the latter approaches require detailed models and a larger computation time to compute train schedules compared to the former approach.

A main drawback of most centralized approaches is that they are heavily affected by the size of instances. A trade-off must be found between the size of the studied area and the time horizon of traffic prediction. In fact, the problem complexity increases

Table 2 Recent contributions on distributed train rescheduling

Paper	Approach	Decomposition	Structure	Solution method
Chou et al. (2007)	MA	Junctions	DEC	Local heuristic
Iyer and Gosh (1995)	MA	Trains, Stations	DEC	Resource allocation
Jia and Zhang (1994)	MA	Areas, Network	HIER	Local heuristic
Lamma et al. (1997)	MA	Stations, Areas	DEC	Local heuristic + priority
Lee and Gosh (2001)	MA	Trains, Stations	DEC	Resource allocation
Mazzarello and Ottaviani (2007)	MI	Areas, Network	HIER	Local heuristic
Missikoff (1997)	MA	Trains	DEC	Local heuristic + priority
Parodi et al. (1996)	MA	Trains	DEC	Resource allocation + priority
Salido et al. (2007)	MA	Trains, Stations	DEC	Local heuristic
Strotmann (2007)	MI	Areas, Network	HIER	Local heuristic

with the number of trains, tracks and stations. If a small test case is considered, only few trains and few conflicts can be detected and solved. Thus, solutions computed with a limited time horizon of traffic prediction are myopic, since the rescheduling process does not consider conflicting trains outside the studied area and time horizon (D'Ariano and Pranzo 2009).

Concerning the solution method and quality, many approaches combine a simulation model with simple decision rules. The performance of such DSSs can be quite erratic. Searching for optimal solutions may result in exceedingly large computation time, so when increasing the size of the problems some speed-up is crucial in order to come up with at least a good quality solution within a reasonable time.

A time-effective approach consists of handling large problems by decomposing them into several smaller, well-structured, weakly interrelated sub-problems. The sub-problems are then solved via effective algorithms. Finally, the coordination issue among the different subsystems is addressed and solved, delivering good quality dispatching solutions in a timely manner.

Table 2 reports a list of contributions on distributed train rescheduling. We analyze four important factors: the level of detail (MA or MI), the entities involved in the decomposition (junctions, trains, stations, areas and/or network), the structure of the decomposition (DEC = Decentralized or HIER = Hierarchical) and the solution methodology.

With a distributed approach, different physical or logical entities, like trains, block sections or infrastructure managers, may have a certain decisional capacity and negotiate with other entities for the access to shared resources. Decisional entities may be organized at the same or at different hierarchical levels. In the latter case, entities at the higher level guide the behavior of lower level entities and the final solution can be reached after a bottom-up or a top-down negotiation. With a bottom-up negotiation, the low level entities take the main decisions and a high level entity checks the decision and, if necessary, asks for modifications to the low level entities. With a top-down negotiation, the high level entity takes the main decisions on the basis of some aggregated model and leaves detailed decisions to the low level entities. The latter

may ask for modifications, for example if no feasible solution can be found given the decisions of the high level entity.

The timetabling problem is typically solved through a top-down decomposition in which tentative timetables are first produced at the higher level and then checked for feasibility on the basis of detailed models. A well-known example of this approach is given by the Dutch system DONS (van den Berg and Odijk 1994), in which CADANS (Schrijver and Steenbeek 1994) produces cyclic timetables at nation-wide level based on macroscopic models and then STATIONS (Zwaneveld et al. 2001) assigns platforms and routes to trains in a railway station based on a more detailed model.

A recent top-down approach to timetabling is proposed by Caimi et al. (2009). With this approach, train service intentions are first generated at the higher level and then a detailed network-wide timetable is built at the lower level by decomposing the network in condensation and compensation zones. Condensation zones are main station areas while compensation zones are open tracks between stations. Detailed timetables are produced in each zone almost independently from each other. The notion of portals is then introduced in order to coordinate the passing times of trains at the border between condensation and compensation zones.

Differently from timetabling, distributed train rescheduling is typically approached with a bottom-up hierarchical approach (Jia and Zhang 1994; Mazzarello and Ottaviani 2007; Strotmann 2007) or with a single level decentralized approach (see Table 2). In fact, the rescheduling problem is heavily constrained locally by the current train positions and by the existing timetable. Therefore, the main decisions have to be taken locally, while higher level decisional entities address the global feasibility issue of local solutions.

On the whole, in the literature on distributed approaches to train rescheduling still few decision support systems are able to quickly and effectively reschedule train movements in practical networks. Despite the efforts devoted to developing sophisticated dispatching procedures, most of the existing approaches suffer from a lack of a general methodology or are designed to solve fictitious instances of traffic flow perturbation. Specifically, from Table 2, among the single level decentralized approaches only Lee and Gosh (2001) study a practical network, while among hierarchical approaches only Mazzarello and Ottaviani (2007) and Jia and Zhang (1994) report on practical experiences.

At least five critical requirements for a good system operability can be envisaged: (i) DSSs should be able to compute dispatching solutions within limited time; (ii) DSSs for line dispatchers should be able to recover local feasibility in presence of multiple train delays or disruptions, i.e., the adjusted timetable must be compliant with the actual train positions and infrastructure status; (iii) DSSs for network dispatchers should be able to check rapidly the local solutions for global feasibility; (iv) DSSs should provide good quality solutions also when dealing with complicated and dense railway networks. (v) DSSs should operate with accurate input data. If the quality of input data is poor, it is unlikely that the solutions provided by the DSSs will be effective in practice. A critical prerequisite of good DSSs is therefore the use of reliable methods for traffic prediction and data estimation.

This work contributes to filling the gap between theory and practice of train rescheduling. Specifically, we refer to the first four critical requirements described

above. The basic methodology for railway traffic regulation and coordination is based on the ideas recently developed within the European project COMBINE 2 (Pacciarelli 2003) and then exploited by Mazzarello and Ottaviani (2007), Strotmann (2007). In this paper, a systematic study of the hierarchically distributed framework is provided and a comparison is carried out with the centralized approach. Differently from previous papers, an exact algorithm is used for rescheduling trains and difficult practical instances are utilized for the assessment.

4 Centralized rescheduling system

This section describes the decisional kernel of the dispatching support system ROMA, that we use as a DSS for the line dispatcher. ROMA detects and solves train conflicts while minimizing the maximum and average consecutive delay in lexicographic order (D'Ariano et al. 2008). In this study, we use the rerouting function of ROMA only for the purpose of disruption recovery and not in the optimization module. This choice limits the need for coordination between dispatching areas to scheduling decisions only. Rerouting decisions are taken by the network dispatcher as a preliminary decision before the scheduling phase. The decisional kernel works as follows. For a given set of real-time disturbances, a conflict resolution procedure computes a new feasible schedule (i.e., deadlock-free and conflict-free) compatible with the status of the network, by defining orders and times for all trains at each block section.

The underlying model used by ROMA to formulate the conflict resolution problem is a job shop with no-store and no-swap constraints. This problem is formulated with an alternative graph (Mascis and Pacciarelli 2002) and by using the blocking time theory to compute arc weights (see, e.g., Pachl 2002). The main value of the alternative graph is the microscopic and flexible modeling of the network topology at the level of railway signal aspects and operational rules.

We next summarize the main aspects of the alternative graph formulation used in this paper. A more detailed description can be found, e.g., in Corman et al. (2010), D'Ariano et al. (2008). In case of fixed block signaling, each pair [train, block section] corresponds to a node in the alternative graph and the arcs between nodes represent precedence constraints. Other constraints relevant to the railway practice can be included into the model (see, e.g., D'Ariano 2008; D'Ariano et al. 2007a, 2007b; D'Ariano and Pranzo 2009). A train route is viewed as a job in the alternative graph. A job is a sequence of operations, i.e., block sections that must be traversed sequentially by the train. A fixed running time for each operation is known in advance, that is the minimum time occurring between the starting of consecutive operations. A possible additional waiting time can be necessary in order to solve train conflicts. A train schedule defines the starting time of each operation. Since a block section cannot host two trains at the same time, a potential conflict occurs whenever two or more trains require the same block section. In this case, a passing order must be defined between the trains that is modeled in the graph by introducing a suitable pair of alternative arcs for each pair of trains traversing the same block section. A deadlock-free and conflict-free schedule is obtained by selecting one of the two alternative arcs from

each pair, in such a way that there are no positive length cycles in the graph (i.e., deadlock). In order to evaluate a train schedule, we use the maximum consecutive delay as the main performance indicator of a solution. The maximum consecutive delay can be computed as the length of the longest path in the graph when exactly one arc from each alternative pair is selected.

4.1 Illustrative example

Let us illustrate the construction of the alternative graph for a small railway network with 9 block sections and three trains (A, B and C). The route of train A is given by the sequence of operations $A1$, $A4$, $A5$, $A6$, $A7$, $A9$. The route of train B is $B2$, $B4$, $B5$, $B6$, $B7$, $B8$ while the route of train C is $C9$, $C7$, $C6$, $C5$, $C4$, $C3$. The three trains share the same path from block section 4 to 7. Additionally, trains A and C share the block section 9. Trains B and C are slow trains and enter their first block section at release time 0 and 100, respectively. Their minimum running time on each block section is 20 time units. Train A is a fast train that enters its first block section at release time 210 and runs on each block section of its route with a minimum running time of 10 time units. All the setup times are fixed to 10 time units. There are only three points that are relevant for the delay computation, namely the block sections 3, 8 and 9. Trains A, B and C are scheduled to leave the network at exit time 270, 120 and 220. There is therefore no extra time added to the travel time of each train in the network.

Figure 1 (top) shows a space-time diagram of the timetable solution. For clarity, we only show the running and setup time of the relevant block sections. Figure 1 (bottom) then presents the corresponding alternative graph formulation. We denote a node with the pair (train, block section) of the associated operation or with the pair (train, exit point), except for dummy nodes. The selected alternative arcs are represented by dashed arcs. For clarity, the running and setup times are not depicted in the alternative graph. The three fixed arcs departing from node 0 model the release time of each train, whereas the arcs entering node n model the objective function. Since the initial delay is zero, the weight of the latter arcs is equal to the exit time of the associated trains from the network as specified in the timetable (but with negative weight). Since there are no delays, the longest path in the graph has length equal to 0.

Figure 2 (top) shows the space-time diagram of the optimal solution to the real-time railway traffic management problem for a disturbed situation in which the entrance of train B is delayed by 185 time units. The dotted line for train B represents its free running path, computed by considering its new entrance time and disregarding the presence of the other trains. The new exit times are 270, 220 and 340 for trains A, B and C, respectively.

Figure 2 (bottom) presents the optimal solution in terms of alternative graphs. Since train B is delayed, its release time and its minimum exit time are updated in the graph. Furthermore, there is an additional waiting time for train B to enter block section 4 that causes a maximum consecutive delay of 35 time units. The other trains are on time.

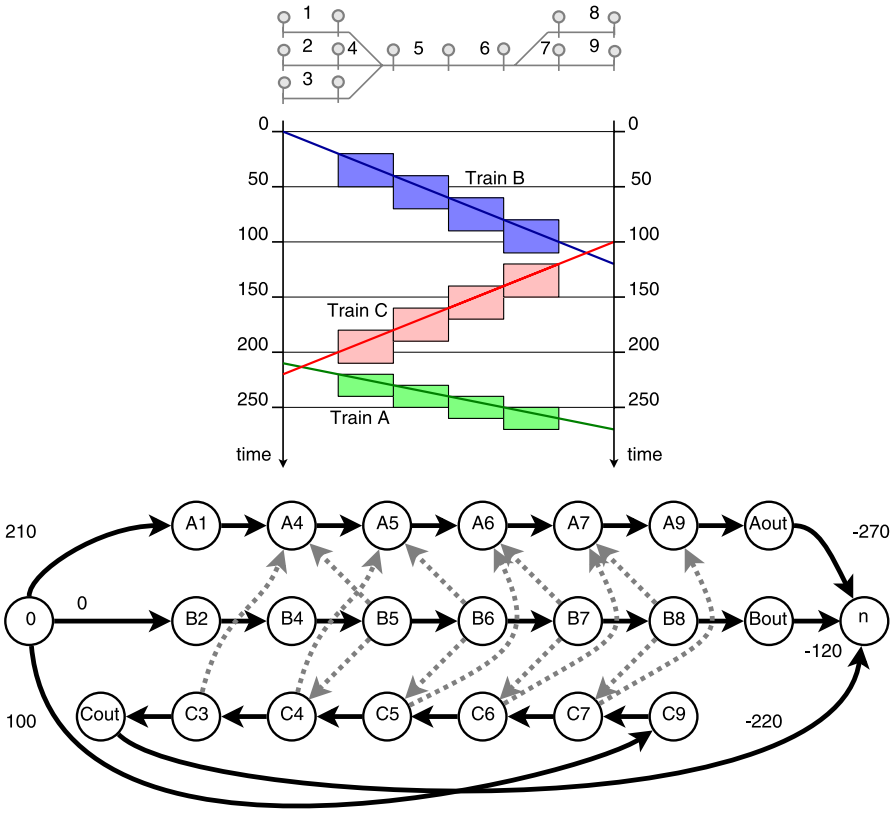


Fig. 1 Timetable solution: space-time diagram (*top*) and alternative graph (*bottom*)

4.2 Train scheduling procedures

Given a timetable, the entrance position of each train in the network and the current status of the infrastructure, each train scheduling procedure implemented in ROMA is in charge of computing quickly a feasible schedule for each train, i.e., defines a feasible entrance time of all trains on each block section. The following three scheduling procedures are tested in this paper:

- First Come First Served (FCFS): This is a well-known dispatching rule which gives precedence to the train arriving first at a block section. This rule requires no dispatching action since trains pass at merging or crossing points on the basis of their actual order of arrival and not necessarily as in the timetable.
- ARI (Automatische Rijweg Instelling): This rule is a fully automated version of the route setting procedure used in the Netherlands (Berends and Ouburg 2005). Decisions on train orders are taken every time a conflict is detected. If both conflicting trains have an entrance delay below 3 minutes the train order is assigned according to the timetable for conflicting trains requiring the same track and according to the First Come First Served rule for conflicting trains requiring different incompatible

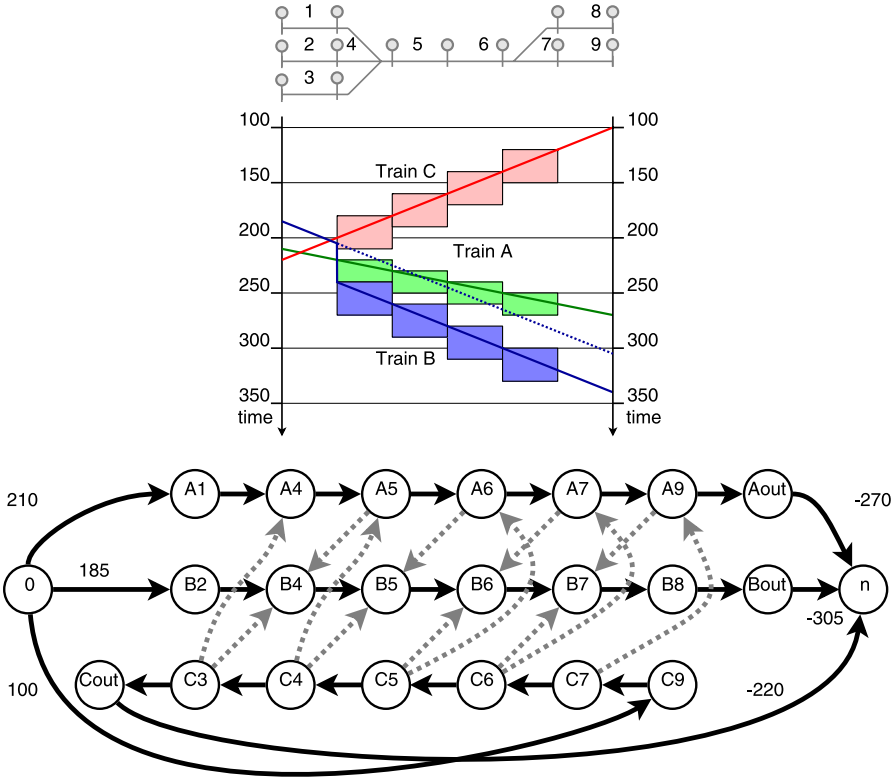


Fig. 2 Optimal train scheduling solution to the disturbed situation

tracks. If at least one of the conflicting trains is delayed by more than 3 minutes, conflicting trains are scheduled on the basis of train priorities: first intercity trains, then local trains and finally freight trains. For trains with the same priority, precedence is given to the train with the smallest number of scheduled stops after the conflicting point.

– Branch and Bound (BB): This is an exact scheduling algorithm that explores all the reordering alternatives and chooses the one minimizing the maximum consecutive delay. Here, we consider a truncated branch and bound (D'Ariano et al. 2007a) that returns near-optimal schedules for practical train scheduling problems within a computation time compatible with operational needs.

The resolution of the train scheduling problem formulated by alternative graphs requires a computation time that is rapidly increasing with the dimensions of the graph. Precisely, the number of disjunctions considered (alternative arcs) increases quadratically with the number of trains and linearly with the number of shared resources (block sections or stopping platforms). Furthermore, the increase in computation time is different for the various algorithms considered. The computation times of FCFS and ARI increase polynomially in the number of variables, while the one of the exhaustive search increases exponentially in the worst case.

5 Distributed rescheduling system

This section addresses the problem of controlling the railway traffic in large networks. In this paper, large network means that a single centralized ROMA system is unable to control the traffic in the whole network within the short time available for computation. In what follows, a large railway network is partitioned by traffic controllers on a geographical basis, since the decomposition of the network into smaller areas should be performed statically. The ideal size of each area clearly depends on the expected traffic pattern in the controlled area, and this parameter has effects on the solution quality.

The following subsections describe the architecture of the distributed system, introduce the aggregate formulation adopted to detect global infeasibilities, and present a novel procedure to coordinate the local scheduling solutions.

5.1 System architecture

Figure 3 shows the distributed system architecture in terms of actors involved and information flow. The system is designed to support hierarchically the work of two line dispatchers and a network dispatcher.

Each area is controlled by a local ROMA, and two local ROMAs exchange information about their solutions, coordinated at a higher level. If a local ROMA is not able to compute a feasible schedule, a *local infeasibility* is found, i.e., conflicts and/or

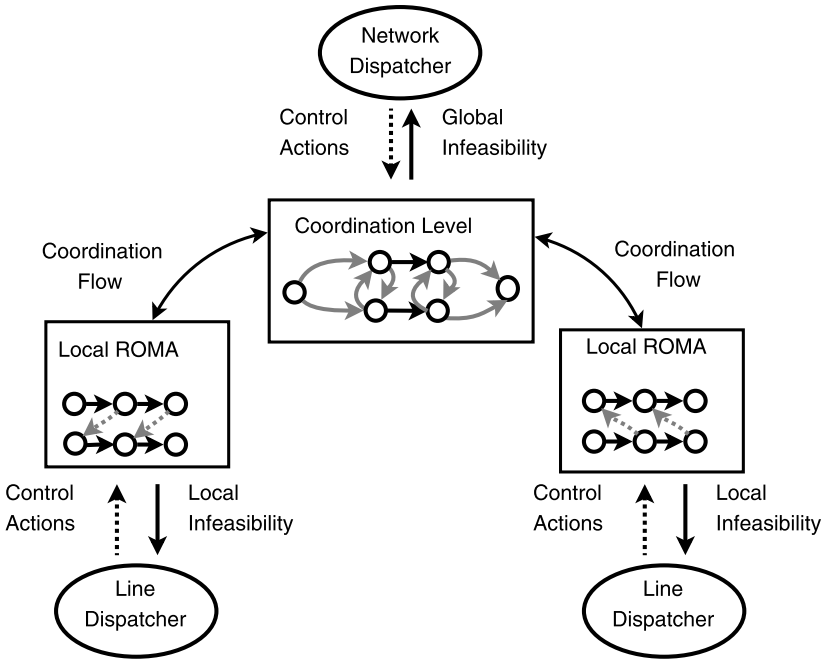


Fig. 3 Framework of the distributed rescheduling system

deadlocks that involve several trains over the local area. In case of local infeasibility, the line dispatcher is asked to take control actions. On the other hand, the coordination procedure may fail, resulting in a *global infeasibility*, i.e., conflicts and/or deadlocks that involve several trains over several local areas. In this other case of infeasibility, the network dispatcher is asked to resolve the situation. This is achieved by imposing additional constraints at global level to the coordination level of the dispatching system or by communicating new control actions to line dispatchers.

The coordination level is in charge of coordinating the solution processes by exchanging relevant local information while detecting and solving global infeasibilities. Only aggregate data describing the interactions between trains running in adjacent areas are sent to the coordinator, that has to solve global infeasibilities (e.g. by changing train orders at the borders) and to transmit quickly the border constraints to the local ROMAs. If no global infeasibility is found, the coordination results in exchanging the relevant border information between the areas. A compact representation of variables and constraints of each area limits the size of the set of data to be managed, avoiding the coordination level to become the bottleneck of the procedure.

A communication protocol has been implemented between the local ROMAs and the coordination level. Reliable and fast exchange of messages between the involved actors is necessary to achieve a synchronized and coordinated behavior. To this end, a standard protocol for high performance computing, Message Passing Interface (Message Passing Interface Forum 1994), is used to effectively support concurrent and parallel execution of processes, as well as inter-process communication.

5.2 Formulation with alternative and border graphs

We now describe the mathematical models for solving and coordinating the scheduling solutions of local solvers (i.e., local ROMAs). A limited amount of local information is exchanged in order to ensure that a set of locally feasible schedules is globally feasible. The information sent by the local ROMAs to the coordinator is extracted from the alternative graph representation of the traffic flow in each local area. For each train passing from an area to another one we consider *border nodes* in the alternative graph. These nodes are associated to the operations representing trains crossing the borders between areas. Note that each block section at the border between two areas is included in both areas. Similarly, the associated border nodes are duplicated.

Figure 4 (top) shows the space-time diagram of a solution to the real-time railway traffic management problem for the disturbed situation of the illustrative example in which train B is delayed by 185 time units. The proposed solution is obtained by dividing the railway network in two local areas, computing the optimal solution for each local area, checking the global feasibility of the local solutions. The three trains (A, B and C) traverse both local areas, called x and y. Trains A and B run from area x to area y while train C runs from area y to area x.

Figure 4 (bottom) reports the local alternative graphs of the proposed solution. The border nodes in the two graphs are $A5$, $A6$, $B5$, $B6$, $C5$ and $C4$. In the left graph, the release time of train C is the time operation $C5$ starts in the other graph, while the minimum exit time of train A and train B is computed as their free running till

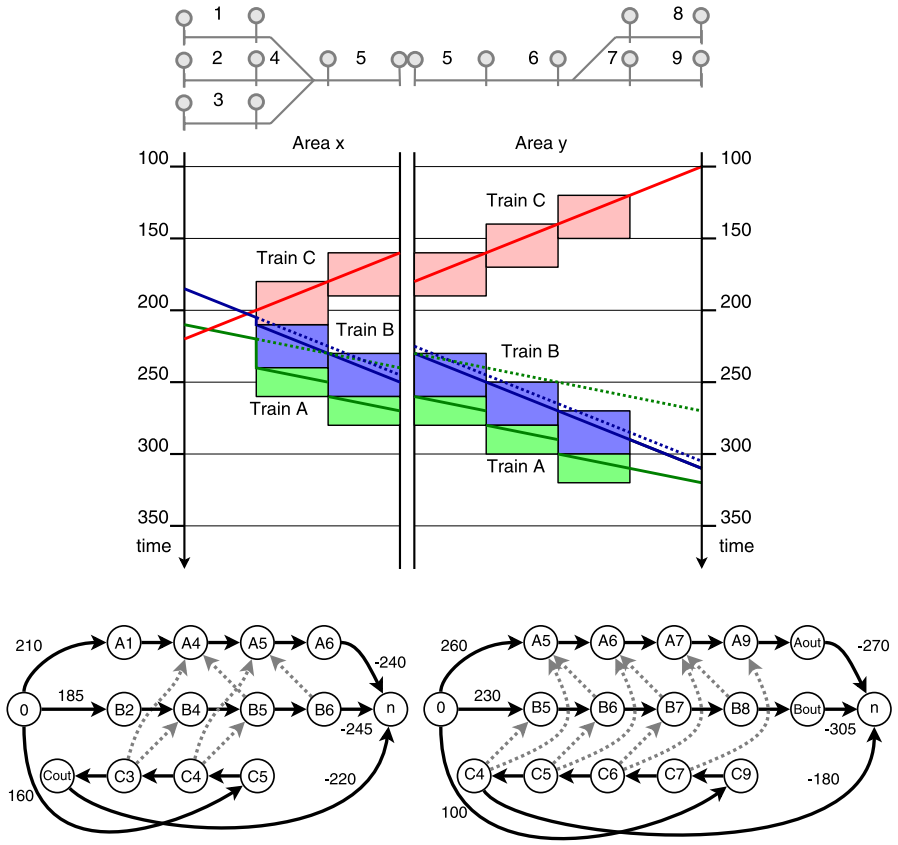


Fig. 4 Optimal local solutions but sub-optimal global solution

the end of block section 5. In the right graph, the release time of train A (B) is the starting time of operation $A5$ ($B5$) in the left graph, while the minimum exit time of train C is computed as its free running till the end of block section 5. In both the local areas, the trains are scheduled at block section 5 in such a way that C precedes B and B precedes A. The local solutions are locally optimal schedules and also result in a globally feasible solution. However, the global solution is sub-optimal. Train B has a conflict with train C on block section 4, with an additional waiting time of 5 time units, while train A has several conflicts with train B on block sections 4, 5, 6 and 7. The most delayed train is train A that has a maximum consecutive delay of 50 time units. This simple example shows that there is an interesting gap between the optimal solution of the global area and the solution obtained by coordinating the optimal solutions of the local areas.

Another disturbed situation is next studied for the timetable of the illustrative example. Trains A and B are delayed at their entrance in the network by 20 and 120 time units, respectively. Figure 5 (top) shows the space-time diagram of the optimal local solutions for the two areas. Specifically, the maximum delay in the left graph is equal to 30 time units since train C has an additional waiting time on block section 5,

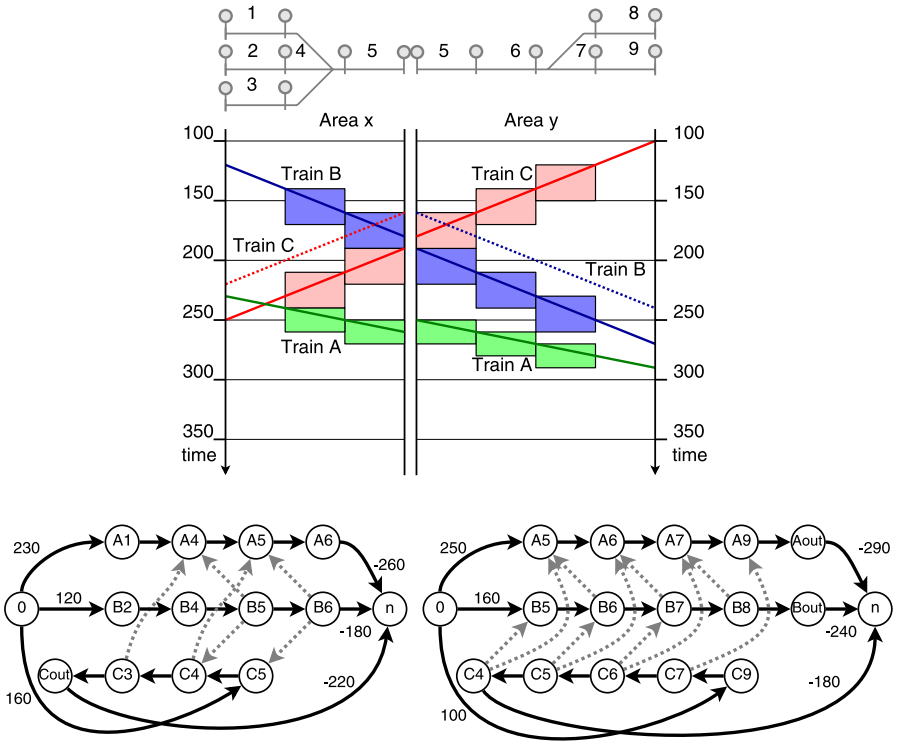


Fig. 5 Optimal local solutions but infeasible global solution

caused by train B. Similarly, the maximum delay in the left graph is equal to 30 time units.

In the new disturbed situation, the optimal solutions of the two areas do not match and the coordinator will detect a global infeasibility as follows. The local ROMA controlling area x decides to give precedence to train B over train C for block section 5. Hence, there is a positive length path in area x from node $B5$ to node $C5$ (see Fig. 5 (bottom-left)). Conversely, the local ROMA controlling area y decides to give precedence to train C over B on block section 5. In this case, there is a positive length path in area y from $C5$ to $B5$. The global situation is not known by the two ROMAs, which consider perfectly feasible their respective plans. On the other hand, at higher level the coordinator detects a positive length cycle involving nodes $B5$ and $C5$, that corresponds to the deadlock situation depicted in Fig. 5 (top). The coordinator must ask one of the two solvers to change the order between trains B and C in order to get a globally feasible solution (i.e., in order to avoid the deadlock situation).

In the remaining part of this subsection, we describe how global infeasibilities between areas can be easily detected by an aggregate formulation. Based on the proposed decomposition of the network and the modeling of border situations, we define the *border graph* $G_B = (V_B, A_B)$, which consists of the set V_B of all border nodes and the set A_B of all border arcs. Note that nodes 0 and n are also nodes of the border graph, since these nodes represent a common reference for all areas. Let b_i, b_j be two

border nodes. The set of border arcs is defined as follows. If a directed path from b_i to b_j exists in one of the solutions for the local areas, the arc (b_i, b_j) belongs to A_B. The weight of (b_i, b_j) is the length of a corresponding longest directed path in this local solution. The border graph (its arcs and the arc weights) is therefore a function of the current local solutions. The following theorem shows that the local solutions produced by the local ROMAs are globally feasible if and only if there are no positive length cycles in the resulting border graph.

Theorem 1 *Consider a global area composed by k local areas. Given a locally feasible solution for each local area, the union of the k solutions is globally feasible if and only if the border graph has no positive length cycles.*

Proof Consider the global alternative graph associated to the whole region. This is obtained by sewing together the k local alternative graphs. By definition, a solution of the global alternative graph is feasible if and only if it contains no positive length cycles. A positive length cycle in the global graph can either belong entirely to a local area, or can involve nodes belonging to several areas. The former case cannot occur since all local solutions are feasible. The latter case occurs if and only if there is a positive length cycle involving at least two border nodes. This means that in each local area there is a directed path connecting a pair of border nodes, the total length of these paths being positive. Thus, there is a cycle in the global graph if and only if the border graph contains a positive length cycle, which concludes the proof. □

Each local ROMA transmits to the coordinator the list of all border nodes, together with the information concerning the route chosen for each train. The coordinator then matches the border nodes of different areas in order to build an aggregate representation of the interactions among the areas. Additional information computed by each local ROMA is the length of the longest path between every pair of border nodes. The coordinator exchanges with the local ROMAs only information related to the border nodes. This information is communicated by each ROMA to the coordinator that has to detect situations of global infeasibility. From Theorem 1, the coordinator simply has to build the border graph and check for the existence of positive length cycles. This can be computed in a fast way by means of existing graph search algorithms. For example, the algorithm of Floyd and Warshall requires a computing time $O(n^3)$, where n is the number of nodes of the border graph.

The left-hand side (right-hand side) of Fig. 6 shows the border graph for the example of Fig. 4 (Fig. 5). In the border graph on the left-hand side of Fig. 6 there is no positive length cycle. For reasons of clarity, we did not depict the (positive) weight of each border arc in the graph. On the other hand, in the border graph on the right-hand side of Fig. 6 there are several positive length cycles involving two or more of the following border nodes: $B5$, $B6$, $C4$ and $C5$. The border arcs causing at least a cycle are emphasized in black in Fig. 6.

When a global infeasibility is detected, the coordinator may either impose precedence constraints among some trains at border nodes or impose the same value for the exit time from an area and the entrance time in the subsequent area for some train (see Sect. 5.3). Each local ROMA will then look for a new schedule in which these

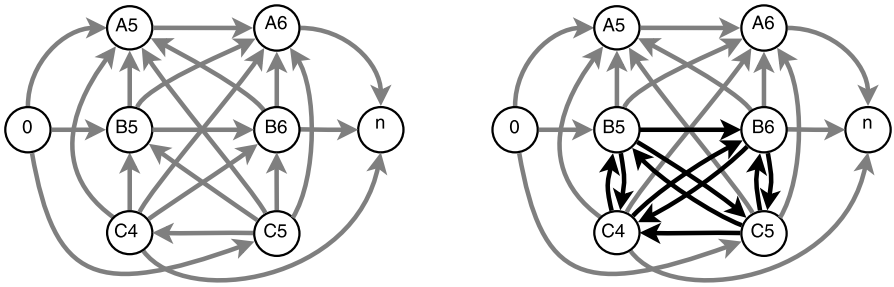


Fig. 6 Border graphs of the feasible (*left*) and infeasible (*right*) global solutions

Fig. 7 General sketch of the schedule coordination procedure

Input: set $\{x_1, x_2\}$ of local areas, border variables $B(i)$
$i = 1$
while time limit of computation is not reached **do**
  for each local area x, concurrently
    impose $B(i)$ to the scheduling problem of the local area x
    solve the scheduling problem of the local area x
    if no solution to the local scheduling problem is found
      return local infeasibility to line dispatcher of area x
    else compute $B_x(i+1)$ from the local solution
  end for
  build border graph from the two local solutions
  if border graph has positive length cycles
    global_feasibility = false
  else global_feasibility = true
  if $B_{x_1}(i+1)$ and $B_{x_2}(i+1)$ are consistent and globally feasible
    return consistent and globally feasible solutions
  use update strategy to compute $B(i{+}1)$ from $B_{x_1}(i{+}1)$ and $B_{x_2}(i{+}1)$
  $i = i + 1$
end
return global infeasibility to network dispatcher

constraints are satisfied. After possible iterations for defining a feasible schedule, the resulting solution will be globally feasible if and only if a locally feasible solution can be found by all local ROMAs and the border graph has no positive length cycles.

5.3 Schedule coordination procedure

Figure 7 shows a new schedule coordination procedure that is based on an iterative exchange of information between two local solvers and the coordinator. At each step, the train scheduling problem in each local area is solved by the corresponding local solver. The coordinator checks whether the local solutions are consistent and result in a globally feasible solution, if not it communicates some constraints at the border between areas to the local solvers. The two steps are repeated until a globally feasible solution is found or a termination criterion is met. For the two areas, the local solutions are *consistent* when the local solvers compute the same times (and orders) for all operations associated to the border nodes between the two areas.

We introduce the border variables $B(i) = (T(i), O(i))$ for each scheduling iteration i so that a consistency check can be done between the local solutions. Variables $B(i)$ are used for bidirectional communication between the local ROMAs and the coordinator. A local ROMA sends a complete solution to the coordinator (times and orders). In response, the coordinator sends a partial solution (release times and partial

orders) to the local ROMAs, regarding constraints at the borders that must be satisfied by the local solver at iteration $(i + 1)$. In the communication from the coordinator to the local solvers at iteration i, the border variables $T(i)$ contain the minimum entrance times at which the trains can enter the block sections at the borders. $O(i)$ is the set of partial orders for the trains at the borders, and clearly depends on $T(i)$.

For each local area, we define the border variables $B_x(i) = (T_x(i), O_x(i))$, representing times and orders of all trains running in the area x at iteration i. For two areas x_1 and x_2, the local solutions of the different areas are consistent if the values of border variables $B_{x_1}(i)$ and $B_{x_2}(i)$ are the same for all the border nodes contained in both areas.

The initial values of the border variables, $B(1)$, are computed as follows. Each local ROMA first computes the minimum starting time of each operation in its local alternative graph by running trains at maximum speed without considering the interactions among them, and communicates the exit time of every train to the coordinator. The coordinator then propagates these exit times to the neighboring area, as the minimum entrance times $T(1)$. The set $O(1)$ is left empty. Consequently, the local areas adjust the values of the border variables. This procedure converges after a finite number of iterative adjustments since the values of the border variables are updated progressively.

Given a set X of local areas and the border variables $B(1)$, the coordination procedure gets a solution from each local area, computed by one of the scheduling algorithms described in Sect. 5.3. The scheduling problem of each local area is decoupled from the one of other areas, so that the local ROMAs can compute their solutions by concurrent and parallel execution. If a local ROMA is not able to find a feasible schedule, a local infeasibility is reported to the line dispatcher that will have to implement other dispatching actions in its local area such as the modification of some train routes.

After computing a feasible schedule for both local areas, each local ROMA reports the new values for the border variables and the longest paths between each pair of border nodes. The coordinator collects this information, builds the border graph and analyzes the local solutions at network level. If the local solutions are globally feasible and consistent, the coordination procedure ends by returning the local solutions. If the local solutions are not globally feasible and consistent, the coordinator calls an update strategy to compute additional coordination constraints, and the overall procedure is iterated with new values for the border variables. If a given time limit of computation is reached before a globally feasible solution is found, the procedure reports a global infeasibility. In this case, the network dispatcher is asked to take scheduling decisions different from the ones computed by the local solvers, such as specify a new processing order between some operations at the area borders.

The simple update strategy implemented in this paper to compute $B(i + 1)$ is as follows. An inconsistency is detected when there is at least one train crossing a border between the two areas with an exit time from an area different from the entrance time in the subsequent area. In this situation, the coordinator imposes a new minimum entrance time for the train in the subsequent area. If a train exits area x_1 and enters area x_2 and its entrance time in area x_2 is smaller than its exit time from area x_1, the coordinator propagates the exit time from the previous area to the next one. In other words, the train is forced to enter area x_2 not before the exit time from area x_1.

When the inconsistency involves train ordering, the coordinator also imposes the same order among the trains at the border in both areas, as follow. If the infeasibility involves two trains running in the same direction, the train order of the previous area is imposed to be the same in the following area. On the other hand, if the infeasibility involves two trains running in opposite directions, the train order is obtained by giving precedence to the first train reaching the border in the schedules of both local areas. In our computational experience, this simple procedure quickly converges to globally feasible solutions since only a few local decisions propagate to the other area.

6 Test case

This section presents the experiments performed to evaluate the centralized and distributed systems over a large sample of real-life instances. The aim of the study is to assess how much train delays could be minimized by centralized and distributed approaches. We compare the solutions obtained by simple and advanced scheduling algorithms, further compared to the dispatching rule used in practice in the Netherlands. The procedures of the two systems are implemented on a PC equipped with two processors at 1.6 GHz, 2 GB Ram and Windows XP operating system. For both systems the time limit of execution is 60 seconds. In the distributed system, each run of the scheduling algorithm is truncated after 10 seconds of computation.

6.1 Description of the instances

The topology of the railway network around the main station of Utrecht is similar to a star with 5 main directions crossing each other (see Fig. 8). The main lines relaying the North and South regions of the Netherlands are connected to the lines to the West and the East. The network is delimited by the following stations: Utrecht Overvecht on the line to Amersfoort, Driebergen-Zeist on the line to Arnhem, Culemborg on

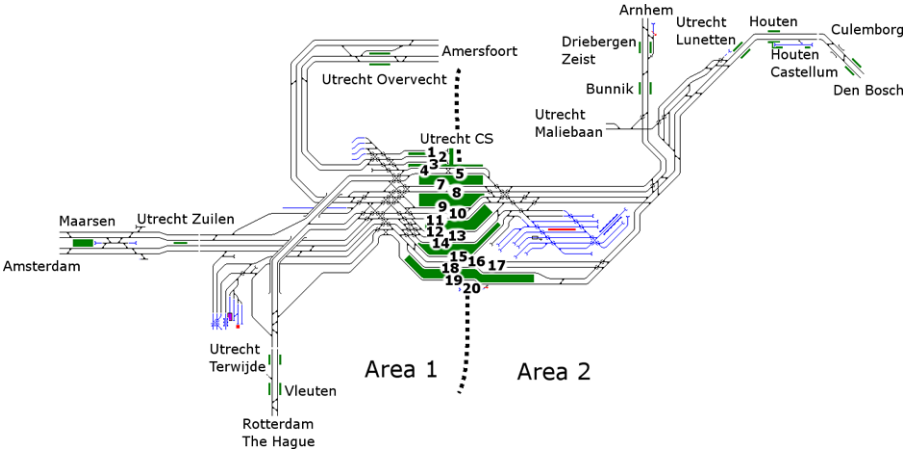


Fig. 8 Railway network around Utrecht Central station

the line towards Den Bosch, Vleuten on the line to Rotterdam and The Hague plus Maarssen on the line towards Amsterdam. In total, the diameter of the area is around 20 km long.

This is one of the most complex railway areas in the Netherlands. In total, the network includes more than 600 block sections. The interlocking area of Utrecht Central station has, alone, around 200 block sections and more than 100 switches, leading to a large amount of inbound and outbound routes. Utrecht Central station provides 20 platform tracks, most of which might be reserved to two trains at a time. Most of the platform tracks are used by through traffic, i.e., trains running in the opposite direction, even if some trains change direction after their stop at a platform. There are also three dead-end platforms. We use one peak hour of the 2008 timetable that schedules up to 80 trains. The trains are mostly for passenger services, operated by NS (Nederlandse Spoorwegen), except for a few freight trains.

In the centralized system the network is managed as a whole, while in the distributed system the network is divided into two adjacent areas, as shown in Fig. 8. At the borders between the two areas there are 20 platform tracks of Utrecht Central station. The operations at the area borders represent the 23 trains traversing the two areas. The other 57 trains only run on a single area. The same division of the dispatching task is also adopted in actual operations.

Practical statistical entrance delay distributions are studied that cause initial delays in the network. For the given timetable, we consider more than 33000 train events (arrivals, departures, dwell processes and passing times) that have been recorded at Utrecht Central station in April 2008 by ProRail. A statistical fitting procedure (Yuan 2006) is adopted to set the parameters of the Weibull distributions, which are used to characterize the delays of different trains and the variation in the dwell time process.

From the calculated statistical distributions, we investigate 90 random timetable perturbation instances with delayed entrance times and dwell time extensions. Each instance considers one hour of time horizon of traffic prediction, with 80 trains in the network. The average entrance delay is around 90 seconds per train while the maximum entrance delay is 390 seconds. These instances represent an average level of perturbed operations over one month of observation. We combine each delay instance with a set of six infrastructure scenarios that result in increasingly reduced availability of infrastructure at Utrecht Central station.

Table 3 Description of the disruption scenarios considered

Infrastructure scenario	Blocked platforms	% Unavailable platforms	% Rerouted trains
i	–	0	0
ii	7 8	10	10
iii	7 8 12 13 14	25	21
iv	2 3 7 8 12 13 14	35	35
v	2 3 7 8 12 13 14 18	40	45
vi	2 3 7 8 10 12 13 14 16 17 18	55	50

Table 3 describes the six infrastructure scenarios (see Column 1). Column 2 reports the specific platforms that are blocked and completely unavailable for traffic, Column 3 the percentage of unavailable platform tracks at Utrecht Central station, and Column 4 the percentage of trains on the hourly timetable that need to be rerouted to another station platform. The train rerouting is limited to the trains that cannot perform their scheduled trip in the station interlocking area with their original route since the assigned platform is blocked. It is worth being noted that scenarios (v) and (vi) require trains running on a single track in different traffic directions, thus resulting in an additional difficulty for the dispatching process due to possible deadlocks.

In total, we consider a set of 540 disturbance instances (i.e., all the combinations of the 90 timetable perturbations and of the 6 infrastructure scenarios). In the next subsection, we will compare the centralized and distributed approaches over all the 540 disturbance instances.

6.2 Computational results

This subsection compares the global solutions obtained by the centralized and distributed systems. We consider the three scheduling procedures described in Sect. 4. For both the centralized and distributed systems, Table 4 reports the average behavior for the ARI procedure, Table 5 for the FCFS procedure and Table 6 for the BB

Table 4 Average results on the two approaches by scheduling trains via the ARI procedure

Tests	Centralized system			Distributed system					
Infra case	Comp time	Max delay	Avg delay	Comp time	Max delay	Avg delay	Coord constr	Sched iter	Glob inf
i	10.5	271	10.9	4.8	276	11.9	14	1.1	0%
ii	11.9	437	22.7	7.2	437	23.3	15	1.2	4%
iii	13.7	584	32.6	7.7	561	31.5	27	2.0	17%
iv	13.9	584	47.0	8.5	561	46.8	27	2.0	17%
v	13.2	1515	66.2	8.6	1520	63.3	27	2.0	16%
vi	15.4	1515	65.2	7.7	1520	64.6	28	1.8	22%

Table 5 Average results on the two approaches by scheduling trains via the FCFS procedure

Tests	Centralized system			Distributed system					
Infra case	Comp time	Max delay	Avg delay	Comp time	Max delay	Avg delay	Coord constr	Sched iter	Glob inf
i	0.1	134	2.8	0.2	141	2.9	13	1.0	0%
ii	0.1	390	14.4	0.3	390	14.2	14	1.1	5%
iii	0.1	495	23.4	0.3	493	23.0	17	1.2	8%
iv	0.1	503	33.8	0.3	503	34.0	17	1.2	8%
v	0.1	1515	56.8	0.3	1520	55.3	17	1.2	8%
vi	0.1	1515	64.1	0.5	1520	62.1	28	1.8	22%

Table 6 Average results on the two approaches by scheduling trains via the BB algorithm

Tests	Centralized system			Distributed system					
Infra case	Comp time	Max delay	Avg delay	Comp time	Max delay	Avg delay	Coord constr	Sched iter	Glob inf
i	0.6	77	1.6	0.5	77	1.6	13	1.0	0%
ii	2.3	221	7.5	0.6	221	7.3	16	1.2	9%
iii	6.6	235	13.4	1.9	235	13.7	18	1.3	12%
iv	6.8	248	17.3	1.8	248	17.7	17	1.2	14%
v	60.0	1515	56.8	22.6	1520	42.0	17	1.2	11%
vi	60.0	1515	64.0	25.8	1520	43.2	24	1.5	29%

algorithm. Each row of the tables reports the average results aggregated over the 90 timetable perturbation instances and for a given infrastructure scenario (see Column 1). Columns 2–4 describe the performance of the centralized system in terms of computation time (in seconds), maximum and average consecutive delays (in seconds). Similarly, Columns 5–7 give the same performance indicators for the distributed system. In both systems, the train delays are computed at all stations and at the exit from the overall network. The last three columns report further information on the distributed system. Column 8 presents the average number of border constraints imposed by the coordinator in order to recover the global feasibility, Column 9 the average number of scheduling iterations in the schedule coordination procedure, Column 10 the percentage of iterations that result in a global infeasibility. For each scheduling iteration, a new border graph is computed in order to check the global feasibility of the current local solutions. On average, the border graph has around 600 border nodes and 1100 border arcs. For all tested instances, there is no infeasibility and inconsistency in each final solution returned by the centralized and distributed systems. This interesting result is probably due to the choice of dividing the proposed two-area railway network at the platform tracks of Utrecht Central station.

Table 4 reports the average results for the implemented ARI dispatching rule. The centralized and distributed systems present global solutions that do not differ much in terms of maximum and average consecutive delays, with the distributed system performing slightly better. This is due to the fact that ARI considers a combination of local scheduling rules and train priorities, which is only relatively sensitive to the problem decomposition and to the actual traffic situation. Both systems are able to deliver a feasible solution within 15 seconds, and the distributed system is always faster than the centralized one. Despite the solution process at the local level is heuristic, the coordination does not result in a major workload, being always able to resolve all the infeasibilities within up to 2 iterations of coordination.

Table 5 gives the average results of the FCFS procedure. Also in this case, the two systems present similar global solutions in terms of consecutive delays. This is due to the fact that FCFS does also not take decisions by evaluating the propagation of train delays. Both systems are again very fast to solve the scheduling instances, even if for this heuristic the distributed system is slightly slower than the centralized one. This is in part due to the overload introduced by the coordination layer and in part to

technical reasons related to the computation of the local schedules, that in practice is not completely performed in parallel due to the iterative exchange of information.

Table 6 presents the average results of the BB algorithm. The centralized system is more time consuming than the distributed one, the difference being more evident in case of strong disruptions. This is due to the direct impact of the instance size and complexity on the computation time of BB (the global alternative graphs have more than 5000 alternative pairs while each of the two local alternative graphs has less than 3000 alternative pairs). However, the global solutions reported for the two systems are very similar in terms of maximum consecutive delays. The distributed system reports smaller average consecutive delays, specially for the infrastructure scenarios (v) and (vi). This is probably due to the iterative use of BB in the distributed solving procedure.

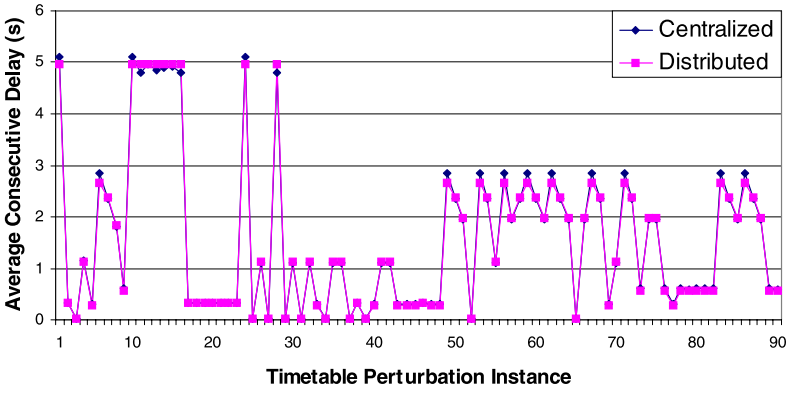
Comparing the solutions computed by the BB algorithm and by the other scheduling procedures, BB outperforms both ARI and FCFS, for both the centralized and distributed systems. This is due to the better performance of BB in minimizing the consecutive delays compared to the local rules.

The last three columns of Tables 4, 5 and 6 give other relevant information about the distributed framework. The number of coordination constraints and of scheduling iterations are two indicators on the information exchanged between the two local ROMAs and the coordinator. Comparing the results for the different infrastructure scenarios, when the level of disturbances increases we conclude that more coordination constraints have to be imposed and more scheduling iterations are required to obtain globally feasible solutions.

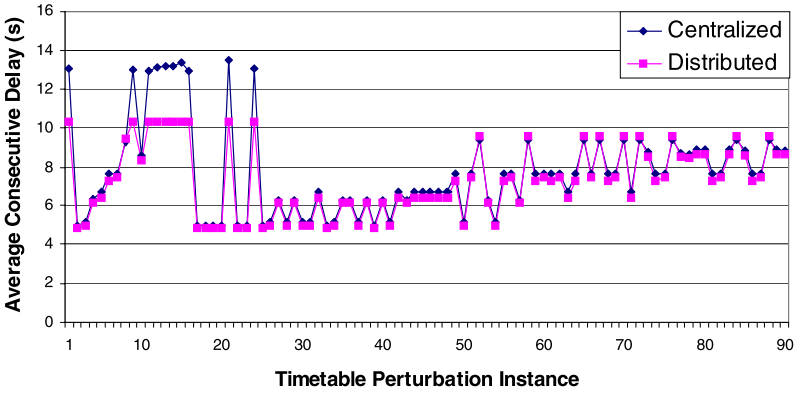
During the iterative scheduling procedure, the percentage of global infeasibilities at intermediate iterations (see the last column of the tables) is also increasing when dealing with more disrupted scenarios. However, the final iteration of the schedule coordination procedure is always able to return a globally feasible solution. It is interesting to note that this indicator does not depend on the scheduling procedure used by the local ROMAs.

Figures 9 and 10 give a detailed view on the performance of the centralized and distributed systems when using the BB algorithm. Each plot is done for the 90 timetable perturbation instances of Sect. 6.1 and one infrastructure scenario. The x-axis shows the instances ordered by increasing average entrance delay, while the y-axis reports the average consecutive delay for each instance. The plots give an idea of the low correlation between entrance and consecutive delays. We found similar features when comparing other input/output factors such as the number of delayed trains or other punctuality measures. Differently, the average consecutive delay increases appreciably with the variation of the infrastructure scenario, since when reducing the available infrastructure more traffic is scheduled on the same platforms and there are more potential conflicts causing a larger delay propagation.

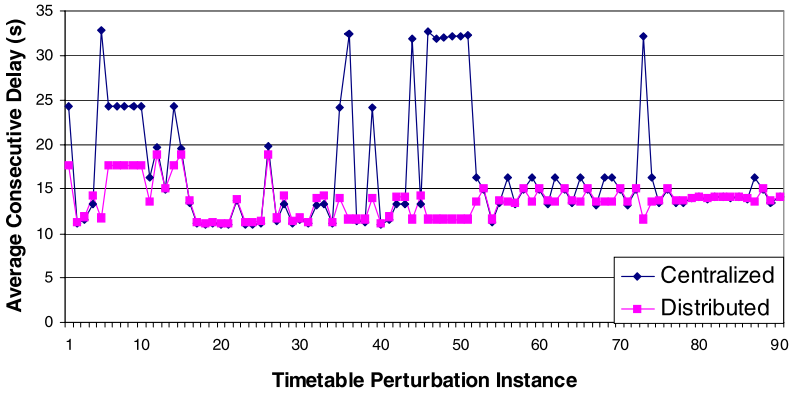
In the scenarios from (iii) to (vi), the distributed system outperforms significantly the centralized system. The distributed system is more robust to changes in the entrance delays in the scenarios (iii) and (iv), and it is systematically better in the scenarios (v) and (vi). The main reasons for this behavior are seemingly the following. On the one hand, it is difficult for the centralized system to compute optimal solutions in a short computation time when trains interact heavily and there is a limited



Infrastructure Scenario (i)

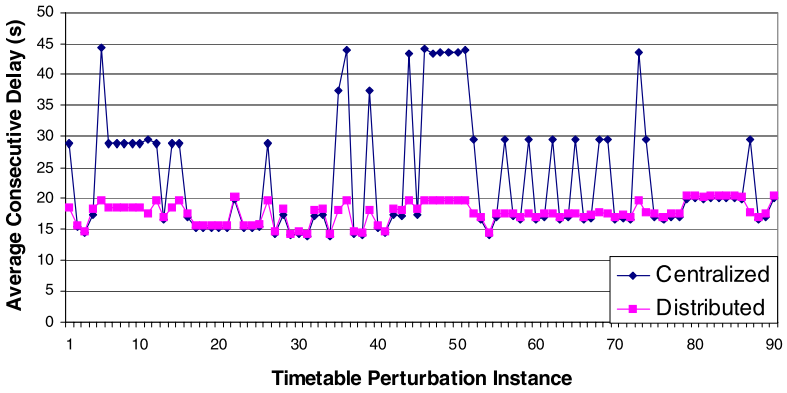


Infrastructure Scenario (ii)

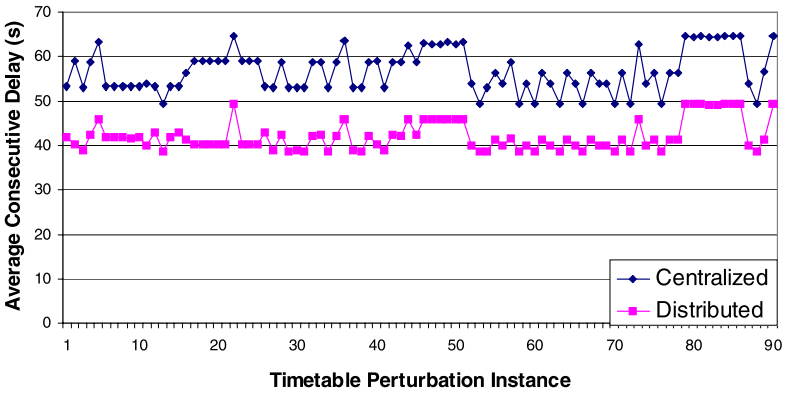


Infrastructure Scenario (iii)

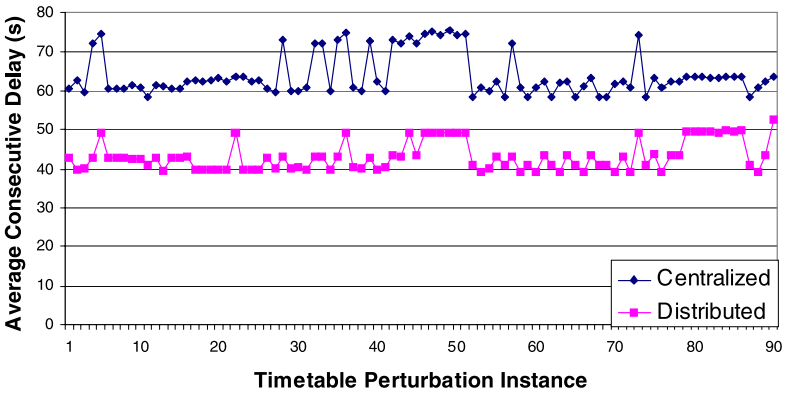
Fig. 9 Comparative diagrams of average consecutive (output) delays against timetable perturbation instances sorted by average entrance delay, scenarios i–iii, BB algorithm



Infrastructure Scenario (iv)



Infrastructure Scenario (v)



Infrastructure Scenario (vi)

Fig. 10 Comparative diagrams of average consecutive (output) delays against timetable perturbation instances sorted by average entrance delay, scenarios iv–vi, BB algorithm

capacity available in the station interlocking areas. On the other hand, the network decomposition into sub-problems keeps the complexity within an acceptable level for the local ROMAs, and the coordination procedure is still able to guide the local solvers to find globally feasible solutions.

7 Conclusions and future research

This paper presents models, procedures and experiments on centralized and distributed rescheduling systems for the management of a complex and densely occupied railway area of the Dutch network. Heuristic and exact train scheduling algorithms plus novel coordination procedures have been investigated on several timetable perturbation instances and infrastructure scenarios. The distributed system with the BB algorithm for the local ROMAs is faster than the centralized ROMA solved by BB, and shows significantly better performance in terms of the consecutive delays, specially for serious traffic disturbances. Differently, when using the FCFS procedure the centralized system is faster than the distributed one, but the average consecutive delay may increase up to 75% compared to the results of BB, and an even larger gap is obtained when comparing BB with the ARI procedure.

Future research should be dedicated to develop more sophisticated techniques for traffic control of large and busy networks. Following this research line, a number of issues should be addressed. Since Theorem 1 holds for any number of areas, effective coordination techniques could be developed for managing larger networks with more than two dispatching areas. To this aim, the coordinator should have more decisional capacity in order to solve possible cycles in the border graph of multiple areas (i.e., deadlocks involving trains running in different areas). The impact of coordination actions on the quality of the global solutions is also a problem that requires further research. The impact of various subdivisions of the network into local areas should be investigated more extensively. Finally, other features, such as rerouting and speed regulation, should be integrated in the decision support systems in order to face practical needs of railway operators.

Acknowledgements We thank the Dutch infrastructure manager ProRail (specially D. Middelkoop and L. Lodder) for providing the instances. This work is partially supported by the Dutch programs "Towards Reliable Mobility" of the Transport Research Centre Delft and by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation".

References

Adenso-Díaz B, González MO, González-Torre P (1999) On-line timetable re-scheduling in regional train services. Transp Res, Part B 33(6):378–398

Berends N, Ouburg N (2005) Beschrijving ARI-functionaliteit. Technical report 20, ProRail Internal Specification (in Dutch), Utrecht, the Netherlands

Caimi G, Burkolter D, Herrmann T, Chudak F, Laumanns M (2009) Design of a railway scheduling model for dense services. Netw Spat Econ 9(1):25–46

Chou YH, Weston PF, Roberts C (2007) Dynamic distributed control for real-time rescheduling of railway networks. In: Hansen IA, Radtke A, Pachl J, Wendler E (eds) Proceedings of the 2nd international seminar on railway operations modelling and analysis, Hannover, Germany

Corman F, D'Ariano A, Pacciarelli D, Pranzo M (2010) A tabu search algorithm for rerouting trains during rail operations. Transp Res, Part B 44(1):175–192

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

D'Ariano A, Pranzo M (2009) An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. Netw Spat Econ 9(1):63–84

D'Ariano A, Pacciarelli D, Pranzo M (2007a) A branch and bound algorithm for scheduling trains in a railway network. Eur J Oper Res 183(2):643–657

D'Ariano A, Pranzo M, Hansen IA (2007b) Conflict resolution and train speed co-ordination for solving real-time timetable perturbations. IEEE Trans Intell Transp Syst 8(2):208–222

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

Goddard E (2006) Overview of signalling and train control systems. In: The 9th institution of engineering and technology professional development course on electric traction systems, Manchester, UK. pp 336–350

Hansen IA, Pachl J (eds) (2008) Railway timetable and traffic: analysis, modelling and simulation. Eurailpress, Hamburg

Hirai C, Tomii N, Tashiro Y, Kondou S, Fujimori A (2006) An algorithm for train rescheduling using rescheduling pattern description language R. In: Allan J, Brebbia CA, Rumsey AF, Sciutto G, Sone S, Goodman CJ (eds) Computers in railways X. WIT Press, Southampton, pp 551–561

Iyer RV, Gosh S (1995) DARYN—A distributed decision-making algorithm for railway networks: Modeling and simulation. IEEE Trans Veh Technol 44(1):180–191

Jacobs J (2004) Reducing delays by means of computer-aided 'on-the-spot' rescheduling. In: Allan J, Brebbia CA, Hill RJ, Sciutto G, Sone S (eds) Computers in railways IX. WIT Press, Southampton, pp 603–612

Jia LM, Zhang XD (1994) Distributed intelligent railway traffic control: a fuzzy-decision making-based approach. Eng Appl Artif Intell 7(3):311–319

Lamma E, Mello P, Milano M (1997) A distributed constraint-based scheduler. Artif Intell Eng 11(2):91–105

Lee TS, Gosh S (2001) Stability of RYNSORD—a decentralized algorithm for railway networks under perturbations. IEEE Trans Veh Technol 50(1):287–301

Mascis A, Pacciarelli D (2002) Job shop scheduling with blocking and no-wait constraints. Eur J Oper Res 143(3):498–517

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

Message Passing Interface Forum (1994) MPI: a message passing interface standard. Int J Supercomput Appl High Perform Comput 8(3)

Missikoff M (1997) An object-oriented approach to an information and decision support system for railway traffic. In: Proceedings of the 1st international conference on knowledge-based intelligent electronic systems, pp 633–641

Nie L, Hansen IA (2005) System analysis of train operations and track occupancy at railway stations. Eur J Transp Infrastruct Res 5(1):31–54

Pacciarelli D (2003) Deliverable D3: Traffic regulation and co-operation methodologies—Code wp4urdv7001d. In: Project COMBINE 2 "enhanced COntrol centres for fixed and moving block sIgNalling systEms—2"—Number: IST-2001-34705

Pachl J (2002) Railway operation and control. VTD Rail Publishing, Mountlake Terrace

Parodi G, Vernazza G, Zunino F (1996) Stability and deadlock avoidance in distributed system for traffic control. IEEE Trans Veh Technol 45(4):732–743

Ping L, Axin N, Limin J, Fuzhang W (2001) Study on intelligent train dispatching. In: Proceedings of the 4th IEEE international conference on intelligent transportation systems, pp 949–953

Rodriguez J (2007) A constraint programming model for real-time train scheduling at junctions. Transp Res, Part B 41(2):231–245

Şahin İ (1999) Railway traffic control and train scheduling based on inter-train conflict management. Transp Res, Part B 33(7):511–534

Salido MA, Abril M, Barber F, Ingolotti L, Tormos P, Lova A (2007) Domain-dependent distributed models for railway scheduling. Knowl-Based Syst 20(2):186–194

Schöbel A (2009) Capacity constraints in delay management. Public Transp Plann Oper 1(2):135–154

Schrijver A, Steenbeek A (1994) Dienstregelingontwikkeling voor Railned: Rapport CADANS 1.0. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, the Netherlands, In Dutch

Strotmann C (2007) Railway scheduling problems and their decomposition. PhD thesis, Universität Osnabrück

Takagi R, Weston PF, Goodman CJ, Bouch C, Armstrong J, Preston J, Sone S (2006) Optimal train control at a junction in the main line rail network using a new object-oriented signalling system model. In: Allan J, Brebbia CA, Rumsey AF, Sciutto G, Sone S, Goodman CJ (eds) Computers in railways X. WIT Press, Southampton, pp 479–488

Törnquist J (2006) Railway traffic disturbance management. PhD thesis, Blekinge Institute of Technology

Tsuruta S, Eguchi T, Yanai S, Ooshima T (1999) A coordination technique in a highly automated train rescheduling system. In: IEEE international conference on systems, man, and cybernetics

van den Berg JHA, Odijk MA (1994) DONS: Computer aided design of regular service time-tables. In: Murthy TKS, Brebbia CA, Mellitt B, Sciutto G, Sone S (eds) Computers in railways IV. WIT Press, Southampton, pp 109–116

Wegele S, Slovák R, Schnieder E (2007) Real-time decision support for optimal dispatching of train operation. In: Hansen IA, Radtke A, Pachl J, Wendler E (eds) Proceedings of the 2nd international seminar on railway operations modelling and analysis, Hannover, Germany

Yuan J (2006) Stochastic modelling of train delays and delay propagation in stations. PhD thesis, TRAIL Thesis Series T2006/6, The Netherlands, 2006

Zwaneveld PJ, Kroon LG, Van Hoesel SPM (2001) Routing trains through a railway station based on a node packing model. Eur J Oper Res 128(1):14–33