Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Dispatching and coordination in multi-area railway traffic management

Francesco Corman[1], Andrea D'Ariano[2], Dario Pacciarelli[2], Marco Pranzo[3]

RT-DIA-190-2011 **Agosto 2011**

(1) Catholic University of Leuven,
Celestijnenlaan 300A - 3001 Heverlee, Belgium.

(2) Università degli Studi Roma Tre,
via della vasca navale, 79
00146 Roma, Italy.

(3) Università degli Studi Siena,
via Roma, 56
53100 Siena, Italy.

This work was partially supported by the Dutch program "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". We thank ProRail managers for providing the Dutch railway data.

ABSTRACT

This paper deals with the development of decision support systems for traffic management of large and busy railway networks. Railway operators typically structure the control of complicated networks into the coordinated control of several local dispatching areas. A dispatcher takes rescheduling decisions on the trains running on its local area while a coordinator addresses global issues that may arise between areas. While several advanced train dispatching models and algorithms have been proposed to support the dispatchers' task, the coordination problem did not receive much attention in the literature on train scheduling. As far as decision support systems are concerned, the control of a railway network can be achieved either by a centralized approach in which the whole scheduling problem is solved by a single scheduler, or by a distributed approach in which a coordinator sets constraints between areas and delegates scheduling decisions to local schedulers. This paper compares centralized and distributed procedures to support the task of dispatchers and coordinators. We adopt dispatching procedures driven by optimization algorithms and based on local or global information and decisions. Computational experiments on a Dutch railway network, actually controlled by ten dispatchers, assess the performance of the centralized and distributed procedures. Different types of traffic disturbances, including entrance delays and blocked tracks, are analyzed on various time horizons of traffic prediction. Results show that the centralized procedure faces increasing difficulty in finding feasible solutions in a short computation time. For seriously disturbed instances with a permanent track blockage, the distributed approach is more suitable than the centralized one in order to manage traffic in the overall area.

Keywords: Train Rescheduling Decisions; Disturbance Handling; Large-Scale Problems; Centralized and Distributed Optimization.

1 Introduction

Railway traffic management of large and busy networks is typically based on detailed train timetables, which are usually defined off-line even in presence of blocked tracks. Careful scheduling is necessary since safety rules between trains impose that at most one train at a time can occupy a *block section*, i.e., a portion of railway delimited by signals. In accordance to such rules, the sequence of trains traversing each block section of the railway network has to be accurately defined so that feasible timetables can be developed, without any deadlock [4]. A set of trains causes a *deadlock* when each train in the set claims a block section which is not available, as it is reserved or occupied by another train in the set, causing a circular wait condition. This leads to the impossibility of further movement by any train in the set.

During operations, a disturbance handling phase is needed in order to limit propagation of train delays and to avoid any deadlock in the network. In the railway practice this real-time task is often hierarchically organized into two decision levels. At the lower level, dispatchers control local areas with a visibility of the traffic flow limited to their respective areas. At the higher level, coordinators are responsible for the traffic management over a railway network of n areas with a global overview of the traffic flow and based on the rescheduling decisions taken by dispatchers. Typically, coordinators are mainly interested in controlling the trains traversing multiple areas and in taking decisions at the border between areas, while the traffic control in the local areas is left to the dispatchers. Figure 1(a) gives an example situation for a small single track line with 10 block sections and 6 trains, divided into 2 dispatching areas, named X and Y. In the left-right traffic direction, trains A and B pass through block sections 1, 2, 4, 5, 7 and 8; train D through block sections 5, 7 and 8. In the other direction, train C passes through block sections 4, 3 and 1; train E through block sections 6, 4, 3 and 1; F through block sections 9, 7, 6, 4 and 10. Figures 1(b) and 1(c) show the view of the local dispatchers on their respective area at a specific time instant of the traffic prediction.

Decision support tools are needed to help dispatchers and coordinators to detect and avoid deadlocks under the two-level hierarchy. On the one hand, tools have been recently developed by academic and professional railway researchers to support the dispatching process in a single dispatching area. On the other hand, the coordination problem did not receive much attention in the literature on multi-area train scheduling, although poor coordination between areas may result in poor overall performance. We will review the related literature in the next section.

Distributed optimization tools offer an effective way to deliver viable solutions in a short computation time. However, such tools evaluate the solution quality and feasibility only locally. The solutions obtained through distributed control may be of bad quality from a global perspective and even the global feasibility cannot be always guaranteed. Coordination is therefore needed in order to provide effective solutions for the overall network. Clearly, dividing the network in a large number of dispatching areas would increase the workload of the coordinator and simplify that of the local dispatchers. So, a compromise has to be found between the size of the dispatching areas, the computation time left to the local dispatchers to produce local solutions, and the type of algorithms used to reschedule trains in each area. Centralized scheduling algorithms can be used in small dispatching areas while distributed approaches with coordination are more suitable in large areas in order to obtain feasible solutions in a short time.

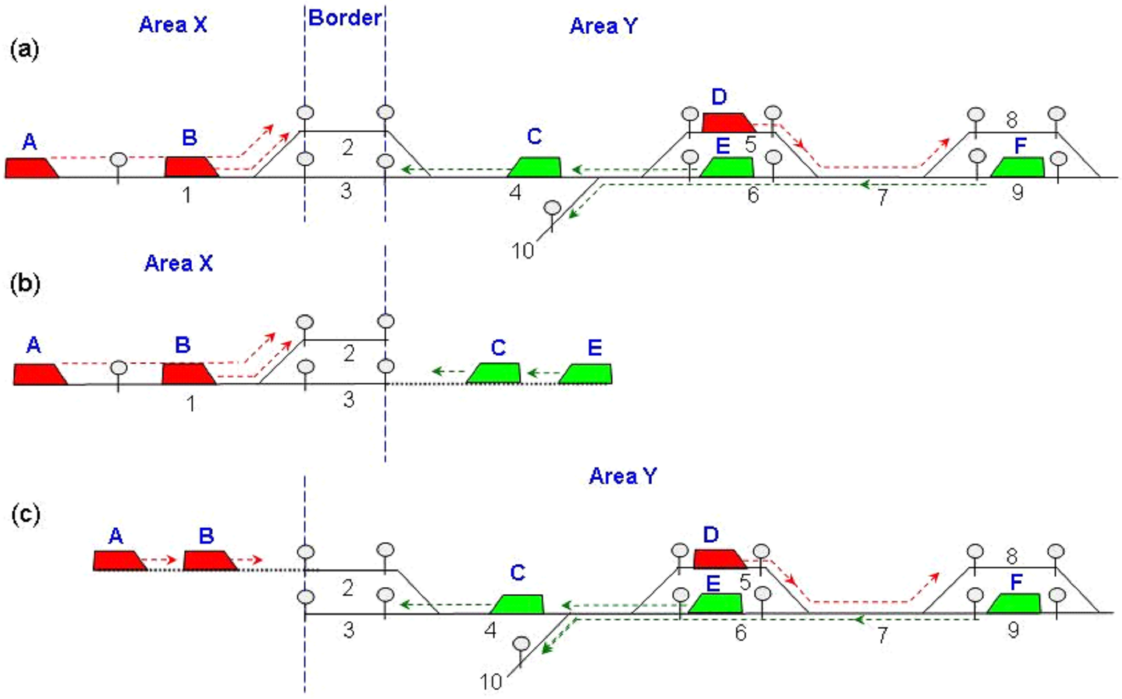

Figure 1: Example of a centralized view plus local views for area X and area Y.

This paper investigates the effects of decomposing a multi-area network in local areas, each controlled by a local dispatcher. In our model, a network coordinator sets constraints at the border between areas and delegates scheduling decisions to local schedulers, driving the search of the local dispatchers toward globally feasible schedules. In this context, the disturbance handling phase can be particularly difficult, specially when managing railway networks with dense traffic. In the example situation of Figure 1 there is a high risk of deadlocks. Specifically, two potential deadlocks are identified by the following circular wait conditions, linking each operation to the next one: [B5 $\rightarrow$ E4 $\rightarrow$ F6 $\rightarrow$ D7 $\rightarrow$ B5] and [E3 $\rightarrow$ B4 $\rightarrow$ A2 $\rightarrow$ C1 $\rightarrow$ E3]. The first deadlock appears if train F precedes train D on block section 7. If the dispatcher of area Y gives precedence to train D over train F on block section 7 then no deadlock arises. The second deadlock can be avoided by the dispatcher of area X by giving precedence to trains B and A over train E on block section 4. This sequencing decision implies that train D precedes train F on block section 7.

With the centralized view of Figure 1(a) is relatively simple to detect this risk of deadlock. In the two-level hierarchy deadlock avoidance can be a harder problem, since this requires to control two trains that do not cross the XY border (i.e., trains F and D), and with our model the coordinator does not take decisions on these trains. Furthermore, the local dispatcher of each area has a limited view of the network and may not recognize the risk of deadlock. In Figure 1(c), if train F is heavily delayed there is a high chance that the dispatcher of area Y gives precedence to train F over D on block section 7, with the (wrong) idea that all the trains moving from right to left can have priority over the others without causing any deadlock. In the next sections, we will show that the coordinator forces the dispatcher of area Y to avoid the deadlock by setting suitable constraints among the trains crossing the border (i.e., trains A, B, C and E).

The main contribution of this paper is the assessment of advanced rescheduling ap-

proaches for managing the rail traffic in a large network with dense traffic. The underlying model used to reschedule trains in each local dispatching area is a microscopic formulation based on the alternative graph of [19] and on the blocking time theory [14]. The coordination model is based on aggregated information coming from each local area. The coordination procedure generalizes to n dispatching areas the one studied in [6, 7] for two dispatching areas of the Dutch railway network. The new coordination procedure is tested on a practical railway network for various network decompositions, including the case in which the control of the whole area is centralized under the control of a single scheduler. The tested approaches range from simple and fast dispatching rules to more sophisticated scheduling heuristics [6, 7, 20, 23, 27] and a state-of-the-art exact algorithm [11]. The advanced train scheduling algorithms take decisions based on global information about the traffic flow and the impact of disturbances. Such algorithms provide better performance of the rail system, with respect to local dispatching rules, but suffer from a rapid increase of execution time and memory requirements as the system grows in size with more trains, track segments and stations. In fact, even the problem of deciding whether a feasible schedule exists or not is NP-hard [19] and thus the complexity of known exact solution algorithms is exponentially growing with the problem size.

An extensive campaign of experiments compares the centralized and distributed procedures on real-world instances. Specifically, four heuristics and two variants of the exact algorithm are evaluated for the dispatching problem. The best algorithms are then incorporated within the distributed framework and tested for network divisions. The assessment is based on different types of traffic disturbance and increasing time horizons of traffic prediction. We consider delayed trains at their entrance in the network and a serious and permanent track blockage situation. The proposed algorithms are compared in terms of number of feasible solutions, computation time and delay minimization.

The paper is organized as follows. Section 2 discusses the literature on railway traffic management on networks spanning over several dispatching areas. Section 3 presents the centralized and distributed approaches, and describes the solution procedures for dispatching and coordination. Section 4 reports on the computational experiments on a Dutch railway network and various disturbed traffic situations. Section 5 concludes the paper and outlines directions for further research.

2 Review of the related literature

This section reviews mathematical models and algorithms to support the task of dispatchers and coordinators. We limit the analysis to the real-time control of railway traffic and do not review papers on timetable development, which address a significantly different problem. A major difference between timetable planning and rescheduling is that in the latter case models and algorithms must be compliant with the actual state of the network and with the current position of each train. Moreover, rescheduling algorithms must deliver good solutions within short computation time. We classify the problems addressed in the literature in distributed rescheduling, centralized rescheduling and coordinated rescheduling. The complexity of the railway network considered ranges from a simple junction to a set of dispatching areas.

Among distributed approaches based on negotiations at the level of junctions, Vernazza and Zunino [29] propose an approach to solve train conflicts locally by enabling

a negotiation between the trains and the local infrastructure administrator. The control problem is modeled in terms of resource allocation tasks and priority rules are adopted for each local controller. The system simulates a realistic network and train conflicts at simple railway junctions are solved by local decision rules depending on the traffic intensity. Parodi et al. [22] study an advanced resource-allocation task for train scheduling based on local decision rules, and study how to detect and solve in advance possible deadlocks for different network configurations.

In Iyer and Gosh [15] every train is equipped with an on-board processor that claims the setup of train routes, dynamically and progressively, through explicit processor to processor communication primitives. Each train negotiates to get access to block sections while minimizing its total travel time. The decision process of each station is executed by a dedicated processor that, in addition, maintains absolute control over a given set of track segments and participates in the negotiation with the trains. Their experiments, carried on an artificial test case with up to 12 stations, 17 track segments and 48 trains, show that the computation time increases rapidly.

Lee and Gosh [18] also report on a decentralized train scheduling algorithm to control large networks within a short computation time. They study the system performance on a simulated railroad network with 50 stations, 84 track segments and several freight trains. Their results indicate that the proposed approach is stable compared to input traffic rate perturbations of finite durations, depending on the size and the degree of capacity use, and unstable under permanent track blockage and communications link failures.

Other distributed approaches use Petri Nets to model the railway traffic flow. Fay [13] describes an expert system and suggests a fuzzy rule-base, Fuzzy Petri Net, for train traffic control during disturbances. Experiments are performed on fictional data with some trains and one station. Zhu [31] introduces a simulation model based on stochastic Petri nets in order to assess the impact of incidents on the quality of operations. The latter approach focuses on how to determine train traffic delays caused by primary stochastic disturbances, especially technical failures. Cheng and Yang [1] include further factors, such as train connections and passenger trip types, in a similar fuzzy Petri Net approach for managing the dispatching process. The dispatching local decision rules are collected via interviews to experts. However, Petri Net approaches still seem to be far from producing near-optimal solutions to practical problem instances.

Several distributed approaches to railway scheduling problems are introduced by Salido et al. [26]. Semi-independent subproblems are generated by graph partitioning and by grouping together subsets of trains or contiguous stations. The subproblems are solved by constraint programming techniques. Experiments based on small fictitious instances with increasing trains and stations show promising results for the distributed approaches.

Among the centralized approaches for managing a dispatching area, Şahin [25] formulates a meet and pass problem as a job shop scheduling problem. Conflicts between up to 20 trains are solved in the order they appear for 19 meet points. An algorithm based on look-ahead measures detects potential delays and takes ordering decisions at merging or crossing points in order to minimize the average delays.

Wegele et al. [30] use genetic algorithms to reschedule trains with the objective of minimizing passenger annoyance, e.g. delays, change of platform stops and missed connections. The adopted dispatching strategies are dwell time modifications, adaptation of train speeds on corridors and local rerouting in-side stations. Examples of application on a large part of the German railway network are reported for a single delayed train.

Rodriguez [24] focuses on a real-time train conflict resolution problem and proposes a train routing and scheduling system based on constraint programming. The experiments show that a truncated branch and bound algorithm can find satisfactory solutions within a short time for a railway junction of a few kilometers traversed by up to 24 trains.

Törnquist and Persson [28] introduce a model for dispatching trains in a railway network with several merging and crossing points. A mixed integer linear programming problem is formulated and solved with commercial software packages. Heuristic scheduling strategies are proposed to reduce the search space by restrictions of reordering and local rerouting actions. Experiments are presented for various disturbance settings on a Swedish railway network with 253 track segments traversed by up to 80 trains for a 90-minute horizon of traffic prediction.

An exact method has been proposed by D'Ariano et al. [11] for a train scheduling problem with fixed routing. Their computational experiments, carried on the Dutch railway bottleneck around Schiphol International airport and for multiple delayed trains, show that optimal or near-optimal solutions can be found within a short computation time. In two follow-up papers [5, 10], this algorithm is incorporated in more sophisticated metaheuristic frameworks to manage multiple train rerouting options.

The problem of coordinating the tasks of different dispatchers is still a very under-researched topic. Among the few papers on this subject, Jia and Zhang [16] present a first approach based on fuzzy decision-making for distributed railway traffic control. A multi-level decisional process is described that consists of several regional decision centers to be coordinated. The test case is a main Chinese network with 12 stations and 12 trains.

Lamma et al. [17] propose a distributed advisory system which helps traffic controllers in traffic management and control within railway stations and along railway branches. The scheduling of trains along a railway line with up to 31 trains is performed by traffic control modules, each one controlling a limited number of block sections and solving train conflicts based on local decision rules. These modules need to frequently exchange information about the local traffic and to verify the solution feasibility.

Chou et al. [2] propose a distributed control system and study a number of railway areas that are mutually influenced. A novel time-shift coordination strategy between neighboring traffic control areas is proposed for collaborative train rescheduling. Distributed control techniques in neighboring regions are applied in a fictitious network and evaluated in terms of delay cost. Recently, Chou et al. [3] demonstrate a significantly lower delay cost compared to a first come first served rule for a realistic railway junction with around 12 trains per peak hour traveling in a single direction.

We next discuss recent approaches based on the alternative graph formulation introduced in [19]. Methodologies for railway traffic regulation and coordination of local areas were developed within the European project COMBINE 2 [21]. Mazzarello and Ottaviani [20] report on the implementation of these methodologies for two test cases of the Dutch railway network. They also report on a practical pilot carried out for one of the two test cases. Based on the COMBINE 2 architecture, Strotmann [27] presents a two-level approach for rescheduling trains between multiple areas. A first level solver computes traffic control measures in each area, while a second level solver is adopted to check whether neighboring areas have consistent solutions. A coordinator graph is used to find out infeasibilities between neighboring areas which will result in additional constraints to the first level solver. An iterative approach of imposing train ordering constraints is adopted until a feasible solution to the global problem is found or it is proven that no globally

feasible solution exists. Fictitious examples are studied with up to 16 trains and 73 block sections. Corman et al. [6, 7] develop ideas from the COMBINE 2 project to solve the coordination problem for a complex and busy railway station divided into two dispatching areas, and compare distributed and centralized systems.

As a general remark about the existing literature, we observe that most of the existing approaches lack of a thorough computational assessment and limit the analysis to simple networks or simple perturbation patterns. In fact, the analyzed delay patterns are often quite specific, e.g. only one train is delayed or the problem is limited to a single junction or to a straight line. Moreover, the models used in the literature for the assessment are often simplified and do not capture entirely the consequences of delays and other disturbances. In order to solve practical problems, detailed models are necessary that capture the real problem complexity. For instance, when dealing with large networks and dense traffic, possibly with disruption, the risk of deadlock is relevant and should not be ignored by real-time models. Safety rules must be modeled accurately in order to avoid deadlock situations in the network, specially for what concerns the reservation and occupation of each block section in busy railway corridors and in complicated station areas. The alternative graph model is among the few models that incorporate this microscopic level of detail within an optimization framework.

In practice, the real-time railway traffic management is based on the decomposition of large and busy networks in coordinated local areas. However, there is a lack of research on this topic, besides few seminal papers with limited empirical assessment. Published papers do not often analyze in depth the trade-off between solution quality, time horizon of traffic prediction and computational effort of the proposed algorithms.

This paper explores the limits for practical applicability of known versus new scheduling and coordination algorithms based on the alternative graph to support railway operators in the management of disturbed traffic situations on an increasing number of trains and levels of disturbance. To this aim, we analyze a Dutch railway network, spanning over ten dispatching areas, with various time horizons of traffic predictions and network decompositions. The disturbances include multiple delayed trains and a serious and permanent disruption in the network, which requires the rerouting of several trains and the management of complex traffic situations with the risk of deadlocks.

3 Solution methods and procedures

This section presents the disturbance handling process addressed in this paper. Centralized and distributed traffic management architectures are first introduced and then described in terms of scheduling and coordination models and algorithms.

3.1 Disturbance handling process

Disturbance handling is the management of railway traffic during operations in order to react to severe disturbances that make the timetable infeasible. The real-time timetable adjustment consists of rescheduling the trains running in the overall railway network, in terms of their routes, orders and times. The railway traffic is predicted over a given time horizon and the delay propagation is studied over multiple dispatching areas.

We consider a *microscopic timetable* which describes the movement of all trains running

in the network during a given hour, specifying, for each train, 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 after its scheduled arrival time. A *timetable perturbation* is given by a set of train delays at the entrance of the network or at scheduled stops and an *infrastructure disruption* is the presence of blocked tracks, requiring modifications of scheduled speeds and/or routes.

The microscopic formulation of the disturbance handling process requires modeling of railway networks at the level of block sections and signals. A block section is a track segment between two main signals and may host at most one train at a time. 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 performed in a dispatching area during a *service*. The *release time* of a train in an area is the minimum time at which the train can enter its first block section of the area, i.e., this is the earliest starting time of the first operation in the area associated to the train route. Each operation requires a given *running time* which depends on the actual speed profile followed by the train while traversing the block section. The minimum time separations among the running trains translate into a minimum *setup time* between the exit of a train from a block section and the entrance of the subsequent train into the same block section.

In our terminology, a *conflict* occurs when two or more trains claim the same block section simultaneously. In this situation a decision on the train ordering has to be taken and at least one of the trains involved has to increase its running time in order to include an additional waiting time to solve the conflict situation. We compute the resulting train delays as follows. The *total delay* is the difference between the calculated train arrival time and the scheduled time at a relevant point in the network, and is divided into two parts. The *initial delay* is caused by disturbances (e.g. failures, entrance delays, blocked tracks) and cannot be recovered by rescheduling train movements, except by running trains at their maximum speed. The *consecutive delay* is caused by the interaction between trains running in the network during a given time horizon of traffic prediction.

3.2 Traffic management architectures

Figure 2 shows the system architectures in terms of actors involved and information flow. We consider a network divided into n local areas with a dispatcher for each area.

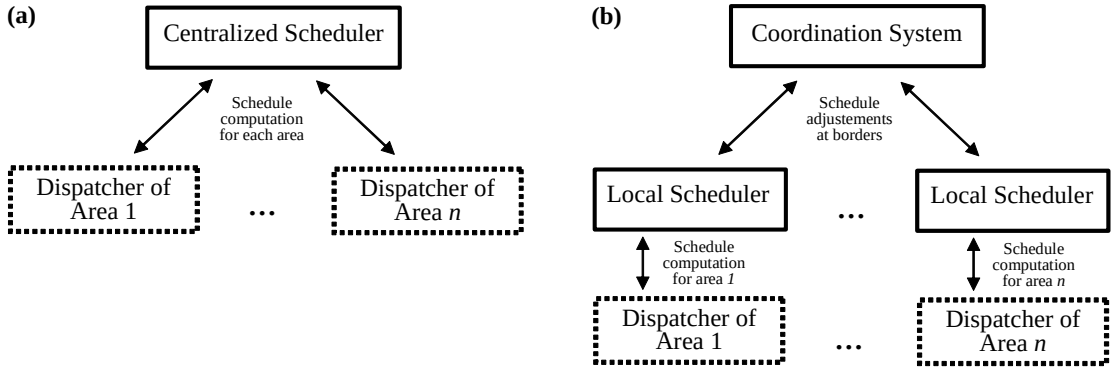

Figure 2: Centralized and distributed architectures.

With the centralized architecture of Figure 2(a), there is no need of coordination since the disturbance handling process is modeled and solved over the overall railway area as a single scheduling problem solved by a single centralized scheduler. When a globally feasible solution is found, each dispatcher receives the solution of its local area by the centralized scheduler. If the centralized scheduler does not find a feasible solution within a given time limit an infeasibility is reported and the human dispatchers are asked to take control actions in their respective areas.

With the distributed architecture of Figure 2(b), each area is controlled by a local scheduler while the overall area is supervised by a coordination system at a higher level. In this case there are two levels of infeasibility. If a local scheduler does not compute a feasible solution within the given time limit a *local infeasibility* is reported and the human dispatcher of that area is asked to take control actions. In this case there is not a globally feasible solution. If all local schedulers find a locally feasible solution in their areas but these solutions are not globally feasible, then a *global infeasibility* is found. In this case, the coordinator is asked to repair the situation by setting constraints at the borders between areas and the local schedulers are asked to find a new local solution compliant with coordination constraints. The procedure continues until a globally feasible solution is found or some local scheduler does not find a locally feasible solution compliant with the coordinator constraints. In the latter case, there is a global infeasibility.

3.3 Dispatching model and procedures

The disturbance handling process is formulated as a job shop scheduling problem with additional constraints and formulated as an alternative graph [19] using the blocking time theory [14] to compute time separations between operations.

The alternative graph is a triple $\mathcal{G} = (N, F, A)$, where N is the set of nodes, F is the set of fixed arcs and A is the set of pairs of alternative arcs. A *selection* S of arcs from A is obtained by choosing at most one arc from each pair in the corresponding set. The selection is *complete* if exactly one arc is chosen from each alternative pair. Given an unselected pair $((i,j),(h,k)) \in A$, a partial selection S *implies* the alternative arc (i,j) if the selection $S' = S \cup \{(h,k)\}$ is infeasible, i.e., if the graph $(N, F \cup S')$ contains a positive length cycle. A problem *solution* is represented by an alternative graph solution $(N, F \cup S)$ in which S is a complete selection. The solution is therefore feasible if the alternative graph contains no positive length cycles.

The alternative graph has been used to model and solve train scheduling problems in several papers [5, 6, 7, 8, 9, 10, 11, 12, 20, 21, 27]. The main value of this formulation is the detailed and flexible representation of network topology and signaling system. In case of fixed block signaling, each block signal corresponds to a node in the alternative graph and the arcs between nodes are used to model the blocking times. The alternative graph represents the routes of all trains in a given control area along with their precedence constraints (minimum headways) and release times. Since a train must traverse the block sections in its route sequentially, a train route is modeled in the alternative graph with a job that is a chain of operations (modeled by nodes in the set N) and associated precedence constraints (modeled by fixed arcs in the set F). This formulation requires that a feasible route for each train is given and a fixed traversing time for each block section is known in advance, except for possible additional waiting times between operations to solve train conflicts. A train schedule thus corresponds to the set of the starting time t_i of each

operation i. 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. At each block section, a passing order between trains must be thus defined. This is modeled in the alternative graph by introducing a suitable pair of alternative arcs (from the set A) for each pair of trains traversing the block section. A deadlock-free and conflict-free schedule is next obtained by selecting one of the two alternative arcs from each pair (i.e., a complete selection S) in such a way that there is no positive length cycle in the graph. In this paper, we use the maximum consecutive delay [11] as performance indicator of a solution. The consecutive delay is the delay introduced when solving conflicts in the dispatching area under study, and is caused by the propagation of the initial delays to the other trains.

Figure 3 shows the alternative graph for the area X of the example in Figure 1. The dummy node 0 and all arcs linked to this node model the release times, while the dummy node $*$ and all arcs linked to this node model the objective function. The other nodes of the set N indicate pairs <train, block section>, e.g. node $B1$ represents train B running on block section 1. The sequence of nodes depends on the train routes, as follows. Train A travels on block sections 1, 2 (its job operations are $A1$, $A2$ and $Aout$), train B travels on block sections 1, 2 (its job operations are $B2$ and $Bout$, since train B is already in block section 1 at time t_0), train C travels on block sections 3, 1 (its job operations are $C3$, $C1$ and $Cout$), train E travels on block sections 3, 1 (its job operations are $E3$, $E1$ and $Eout$). The arcs in the set F are arcs between nodes of the same job or arcs connected to nodes 0 or $*$.

In Figure 3, all arcs linking operations of different jobs are the alternative arcs of the initial selection S, implied by the initial position of the trains. The 2 pairs of unselected alternative arcs are not depicted in Figure 3, i.e., the alternative pairs: $((A2, C1), (Cout, A1))$ and $((A2, E1), (Eout, A1))$.

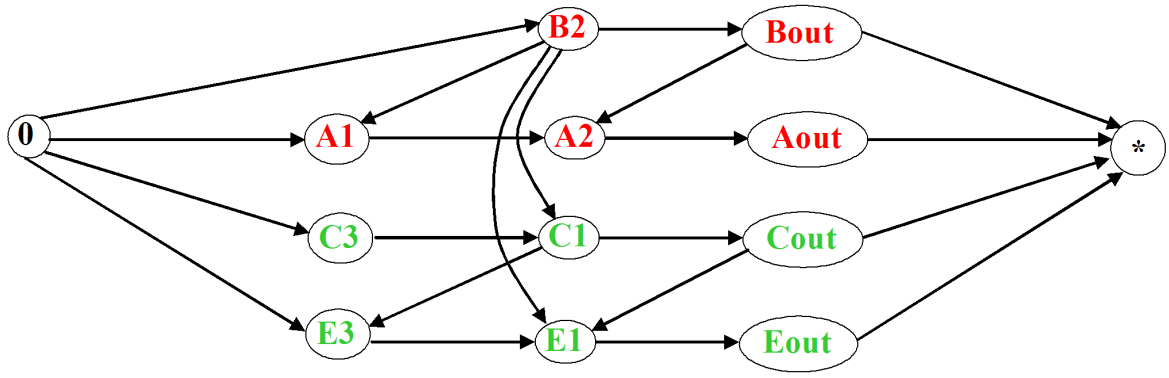

Figure 3: Initial selection for the alternative graph of area X.

We next describe the 15 scheduling procedures we use to solve the dispatching problem. Specifically, we consider the following published procedures: 12 heuristics and an exact algorithm. We then have a new heuristic used as a stand alone procedure or incorporated in the branch and bound algorithm described in [11]. In total, we evaluate 13 heuristics and 2 exact algorithms.

The First Come First Served (FCFS) dispatching rule, often used in the railway practice, solves train conflicts by assigning each block section to the first train that requires it. In other words, FCFS gives precedence to the train arriving first at each 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 scheduled in the timetable.

Another dispatching rule, called First Leave First Served (FLFS), is as follows. When two trains claim the same block section, we first compute the time required for each train to enter and traverse it. Precedence is then given to the train that would leave the block section first. FLFS is a compromise approach between two commonly used dispatching rules: (i) give priority to the fast trains over the slow ones and (ii) FCFS.

Ten heuristics belong to the family of the Arc Greedy Heuristics (AGH) introduced by [23]. All the heuristics of this family have the same algorithmic structure and are based on the idea of repeatedly extending a feasible selection. At each step, one unselected pair from the set A is chosen and one of the two alternative arcs is added to the current selection, until a complete selection is built or a positive length cycle is detected. The algorithms in the family differ only for the evaluation criterion applied to select the next alternative arc at each step. In this paper, we consider the following AGH criteria: AMCC (Avoid Most Critical Completion time), AMSP (Avoid Most Similar Pair), ALCP (Avoid Less Critical Pair), AMBP (Avoid Most Balanced Pair), AMPP (Avoid Maximum Product Pair), SMCP (Select Most Critical Pair), SMSP (Select Most Similar Pair), SLCP (Select Less Critical Pair), SMBP (Select Most Balanced Pair), SMPP (Select Maximum Product Pair). The criteria are described in detail by Pranzo et al. [23] and Strotmann [27].

The exact method that we consider in this paper, referred to in the following as BB, is the branch and bound algorithm described in D'Ariano et al. [11]. Also this algorithm is based on the alternative graph and computes optimal train schedules by using static and dynamic implications, including speed-ups based on the infrastructure topology. BB uses as initial solution the best one computed by FCFS, FLFS and AMCC.

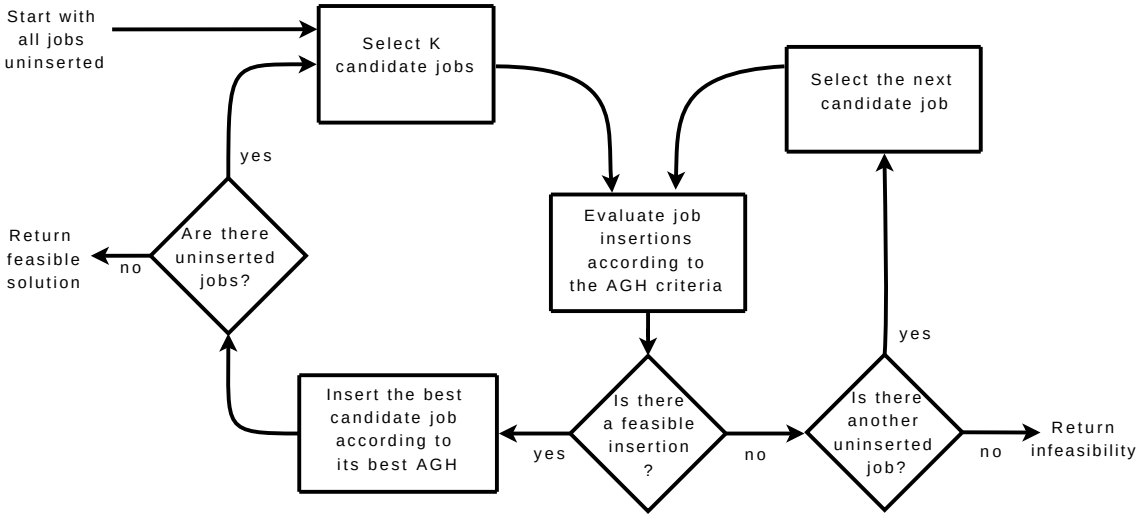

Figure 4: General scheme of the JGH procedure.

The new heuristic is called Job Greedy Heuristic (JGH). This is also based on the idea of repeatedly extending a feasible selection. However, differently from AGH, the decision taken at each step consists in the insertion of a complete job in the current partial schedule, until a solution is found or an infeasibility is detected. As shown in Figure 4, at each iteration the insertion procedure tries to extend a current selection by selecting all the alternative pairs involving arcs between one of the previously inserted

jobs and a new candidate job. The new selection is obtained by applying one or more arc greedy heuristics to the restricted problem and by retaining the best solution found. In order to choose the next job to insert, several candidate jobs are evaluated for insertion by building the new selection. The candidate job achieving the best evaluation is permanently inserted in the partial schedule and a new iteration begins. Possibly all the uninserted jobs are evaluated for insertion as candidate. If no feasible insertion is found the algorithm reports an infeasibility. To speed up the computation, JGH evaluates the insertion of at most K jobs at each step, where K is the minimum between the number of remaining uninserted jobs, a fraction Y of the total job number or a maximum number of insertions W. The K jobs chosen among the uninserted jobs are those with minimum release times. If no feasible insertion is found among the first K jobs, the algorithm tries to insert also the remaining uninserted jobs, one at a time ordered for increasing release time, until a feasible insertion is found, if any.

The last algorithm, referred to as BB+, is simply obtained from BB by adding JGH to the set of heuristics used to compute the initial solution. We test BB and BB+ to evaluate the contribution of JGH to the performance of the branch and bound procedure.

3.4 Coordination model and procedures

A well known approach to face large and hard scheduling problems consists of decomposing them into smaller affordable sub-problems, which can be solved independently from each other. The local solutions are then composed to obtain a solution to the original problem. In this section we apply this kind of approach to reschedule train movements in a large area with dense traffic. The computational complexity of the train scheduling problem is limited within an acceptable level by network decomposition. The scheduling decisions in each local area are taken in a fully parallel fashion by the local schedulers. Since the composed solution must be globally feasible, the main issue is to coordinate the local schedulers in such a way that the union of all local solutions is globally feasible.

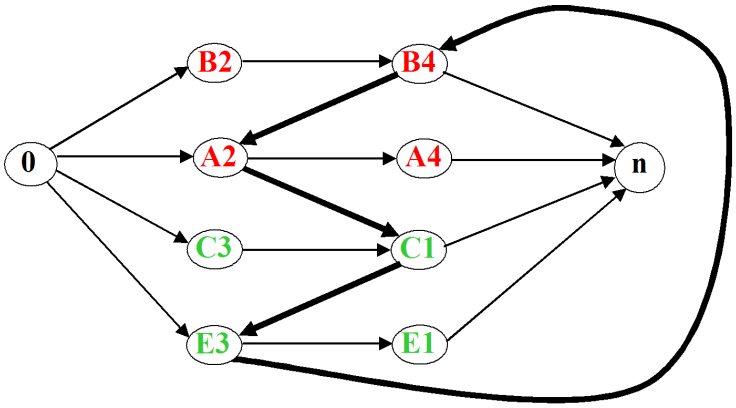

Figure 5: Positive length cycle in the border graph (in bold).

Given a division of the network, we adopt a compact representation of variables and constraints of each local area in order to limit the size of the set of data to be managed by the coordinator, avoiding the coordination task to become the bottleneck of the procedure. Each local dispatcher sends to the coordinator the entrance/exit time of each

train traversing its borders and, for each pair of trains entering/leaving the local area, the minimum temporal distance between the two events. This information is used to build 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. Border nodes are associated to the operations representing trains crossing the borders between 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 the graph associated to a solutions of a local area, the arc (b_i, b_j) is added to A_B with weight equal to the length of a longest path from b_i to b_j in the local graph. As proved in [6, 7], the local solutions produced by the local schedulers are globally feasible if and only if there are no positive length cycles in the resulting border graph. Figure 5 shows a possible positive length cycle in the border graph of the illustrative example of Figure 1. The bold arcs of the border graph correspond to longest paths in the local graphs. This information is sent by the local schedulers to the coordinator.

The coordination procedure is based on the border graph and makes use of the information provided by the local dispatcher and the heuristic rule described in [6, 7]. With respect to the former procedure, we added a pre-processing phase in which each local scheduler sends to the coordinator the precedences among trains that must be kept in order to find a feasible solution. These are all the longest paths between border nodes in the graph associated to the initial selection $(N, A \cup S^0)$ of its area, i.e., the selection S^0 is obtained by the constraints on the initial position of each train. These constraints are then propagated on the alternative graph by using the static and dynamic implications described in [11]. With reference to the example in Figure 1(b), the local scheduler of area X finds a positive length path from the exit of B to the entrance of E (i.e., there is the path composed of arcs $(Bout, A2)$,$(A2, C1)$ and $(C1, E3)$ in the alternative graph of area X). This information becomes a precedence constraint $(B4, E3)$ with positive weight in the alternative graph of area Y.

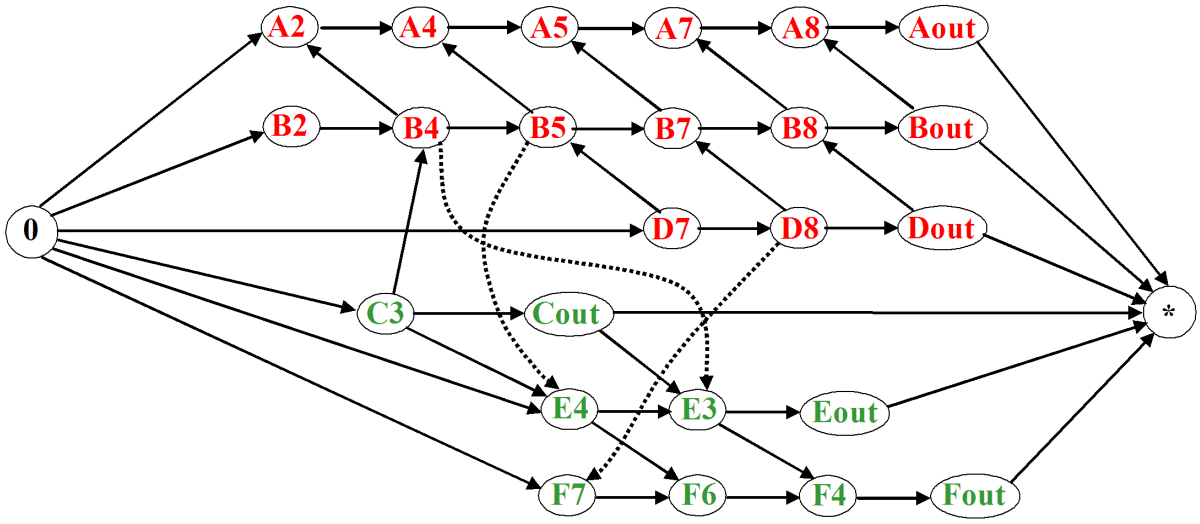

Figure 6: Initial selection for the alternative graph of area Y.

Figure 6 shows the alternative graph for the area Y of the example in Figure 1. Train A travels on block sections 2, 4, 5, 7, 8 (its job operations are $A2$, $A4$, $A5$, $A7$, $A8$ and $Aout$), train B travels on block sections 2, 4, 5, 7, 8 (its job operations are $B2$, $B4$, $B5$, $B7$, $B8$ and $Bout$), train D travels on block sections 5, 7, 8 (its job operations are $D7$,

$D8$ and $Dout$, since train D is already in block section 5 at t_0), train C travels on block sections 4, 3 (its job operations are $C3$ and $Cout$, since train C is already in block section 4 at t_0), train E travels on block sections 6, 4, 3 (its job operations are $E4$, $E3$ and $Eout$, since train E is already in block section 6 at t_0), train F travels on block sections 9, 7, 6, 4 (its job operations are $F7$, $F6$, $F4$ and $Fout$, since train F is already in block section 9 at t_0).

In Figure 6, the solid arcs between different jobs are the alternative arcs implied by the initial position of the trains (redundant arcs are not depicted). The other pairs of unselected alternative arcs are not depicted in Figure 6, i.e., the 7 alternative pairs: $((A5, E4), (E3, A4))$, $((B5, E4), (E3, B4))$, $((A5, F4), (F10, A4))$, $((B5, F4), (F10, B4))$, $((A8, F7), (F6, A7))$, $((B8, F7), (F6, B7))$ and $((D8, F7), (F6, D7))$. The 3 dotted arcs are not included in the initial selection of area Y. However, if the coordinator imposes the precedence constraint (arc) $(B4, E3)$ to the local scheduler of area Y, the initial selection with $(B4, E3)$ implies $(B5, E4)$, and the latter constraint implies $(D8, F7)$. Clearly, the alternative insertion of the dotted arcs would introduce a positive length cycle in the graph of area Y.

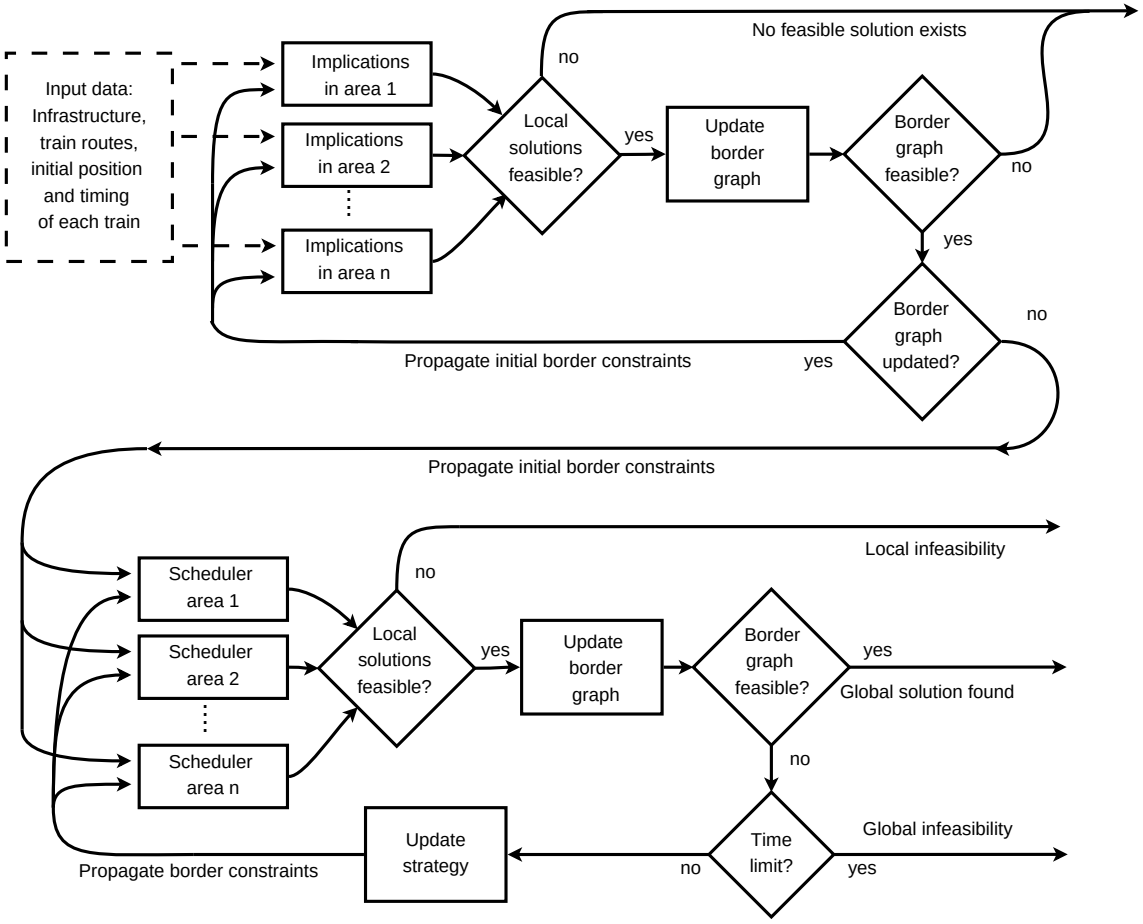

Figure 7: Scheme of the proposed coordination procedure.

Figure 7 presents the general coordination procedure for n local areas. The top part of the figure refers to a pre-processing phase to find an initial set of border constraints. These implications are computed for the block sections of each local area by the initial position and timing of trains. If a local area reports an infeasible starting situation (i.e., a

deadlock that is unavoidable due to the initial position of trains), the system reports that no feasible solution to the whole problem exists. Otherwise, the border graph is used to check the feasibility of the union of the local implications. At this level, a positive length cycle in the border graph proves that no feasible solution exists. On the other hand, a feasible border graph will result in updated constraints regarding the times and orders of trains at the entrance in each local area. Those constraints are propagated iteratively, until all starting implications have been computed. The pre-processing phase defines an initial set of border constraints for the following scheduling and coordination procedures.

In the bottom part of Figure 7, each local scheduler starts by computing the train scheduling solution of the corresponding local area. The local solutions, if locally feasible, are used to build the border graph. The border graph is used to check the global feasibility of the local solutions. A number of border constraints between each pair of adjacent dispatching areas have to be satisfied: (i) the exit time of a train from one area must be equal to the entrance time in the adjacent area, (ii) trains traversing a border on the same block section must keep the same order at the exit from an area and at the entrance in the adjacent area, (iii) locally feasible solutions should not cause deadlock situations from a global perspective. Such constraints cannot be checked directly by the single local dispatcher, since each sub-problem has a myopic view of the entire process. In [6, 7], we show that simple exchange of information between neighboring areas is sufficient when there are only two areas. In case of three or more areas, a global overview of all areas is necessary, i.e., the coordinator must take decisions in order to avoid any possible infeasibility. In case of a global infeasibility, the coordinator may either impose precedence constraints among some trains at border nodes or may impose the same value for the exit time from an area and the entrance time in the subsequent area for some train. This is done by an update strategy that imposes new constraints at the borders. Each local scheduler then computes a new schedule in which these constraints are satisfied. The iterative procedure terminates when a globally feasible solution is found or when a local infeasibility is found or when the time limit of computation is reached.

In the pre-processing phase, no update strategy is applied since all starting implications are unavoidable. During the scheduling phase, instead, each local scheduler computes a possible solution to its local problem. In order to recover global feasibility, the update strategy computes new coordination constraints.

The update strategy adopted in this paper works as follows. For each pair of adjacent local dispatchers, the strategy checks all border constraints for possible violation and takes the following actions in order to remove any violation. For each violated constraint (i) the coordinator sets the release time of the train entering the next area equal to the exit time of the same train from the previous area. For each violation (ii), 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 two schedules of both local areas. For a violation (iii), the coordinator detects a positive length cycle in the border graph. In this case, there must be at least one train, among those involved in the cycle, for which the exit time from an area is larger than the entrance time in the next area computed by the two associated local schedulers. The coordinator selects among these trains the earliest exit time τ scheduled by the local dispatchers, and then sets the release time of the associated train in the next area equal to τ. In other words, the coordinator sets only

one additional constraint for one train and an area with this violation. Then, the local scheduler of the subsequent area is asked to reschedule trains with the new release time. Chances are that in the next iteration this train is scheduled in a different way so there will be no more positive length cycle in the border graph.

3.5 Overview of the different techniques

Table 1 shows differences and similarities of the scheduling algorithms from the perspective of local or global decisions and local or global information used to take decisions.

Table 1: Comparison of train scheduling algorithms.

Algorithm	Decisional level	Information Level
FCFS	Local	Local
FLFS	Local	Local
AGH	Local	Global
JGH	Global	Global
BB	Global	Global
Coordination	Local	Global

To summarize, FCFS and FLFS are two representative local rules that take train ordering decisions at each railway junction by means of simple and local decision criteria. The information used is the time at which each train reaches the specific junction. Differently, AGH and JGH use global information by means of the alternative graph formulation. AGH takes local decisions, i.e., at each step the heuristic procedure decides the precedence between two trains at a specific block section. JGH is the only considered heuristic that applies global decisions by scheduling the whole path of a train at each step. Clearly, the exact methods implemented for the alternative graph formulation (i.e., BB and BB+) are based on global information and decisions. Finally, the coordinator uses local schedulers with global information on their area (i.e., BB, BB+ or JGH), while the coordination procedure is based on global information on the overall network and on local decisions at the borders between pairs of dispatching areas.

4 Computational experiments

This section presents our computational results on a large part of the Dutch railway network, and evaluates the dispatching and coordination approaches of Section 3. The algorithms are implemented in C++ and run on a PC equipped with a dual-core processor Intel Pentium D (3 GHz), 1 GB Ram and Linux operating system. We adopted a time limit of 10 seconds for each local scheduler of the distributed architecture and a time limit of 300 seconds for the overall centralized/distributed procedures.

4.1 Test case description

We study a railway network in the South-East of the Netherlands that spans over ten dispatching areas of the Dutch railway network. The network layout comprises a combination of single and double-tracks of different length, with a maximum distance between

area borders of around 300 km. In total, there are more than 1200 block sections and stopping platforms at stations. The network studied includes the major stations of Utrecht Central, Arnhem and Den Bosch, plus other 40 minor stations. The two main traffic directions are served by the line between Utrecht and Arnhem (towards Germany) and the line between Utrecht and Den Bosch (from Amsterdam towards Eindhoven and the southern part of the country).

For the distributed architecture, we consider the three network decompositions of Figure 8. For the 3-area division the border stations are Bunnik, Utrecht Lunetten and Nijmegen Dukenburg, while for the 6-area division the border stations are also Den Bosch Oost, Arnhem Zuid, Utrecht Terwijde and Utrecht Zuilen. Finally, the 9-area division considers the additional borders located at Utrecht Central Station (where most platforms can be occupied by up to two trains at the same time), Oosterbeek, Zaltbommel.

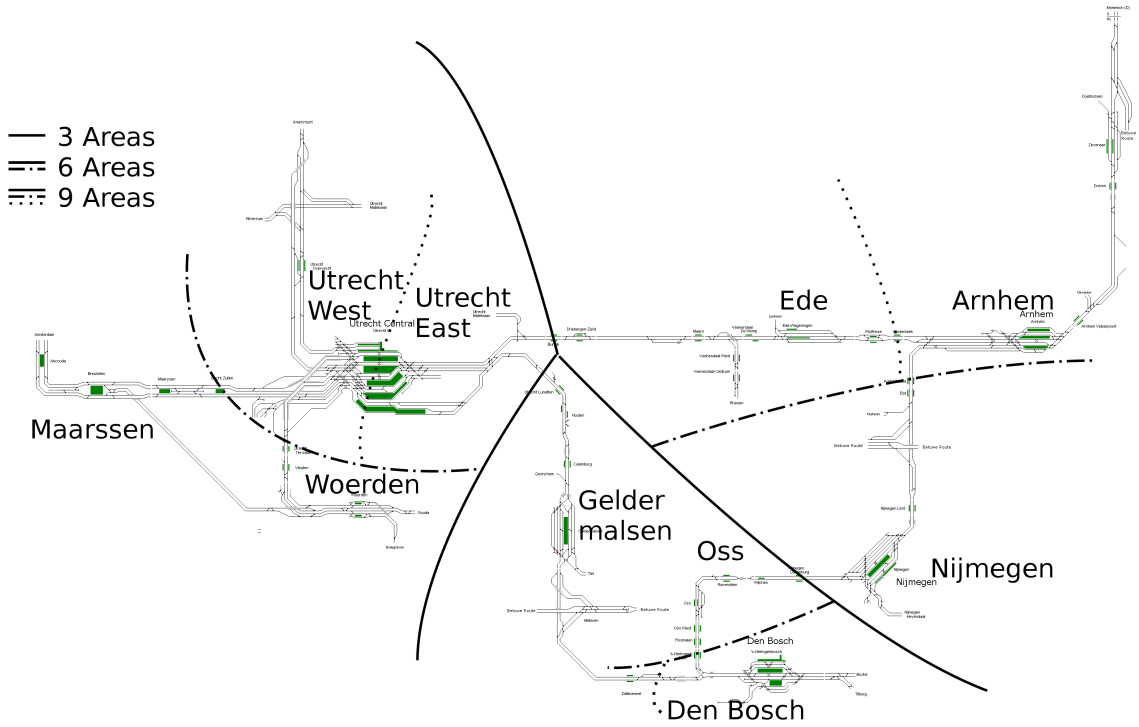

Figure 8: The studied railway network and three divisions into local areas.

The reference timetable is periodic with a periodicity of one hour. Train traffic includes local and intercity services. We consider three time horizons of traffic prediction in order to investigate delay propagation on graphs of increasing size.

Table 2 shows information on the border graphs for the three divisions. Column 1 reports the time horizon of reference (in minutes), Column 2-3, 4-5 and 6-7 show the number of arcs and nodes for the 3-area, 6-area and 9-area divisions, respectively. When enlarging the time horizon, we obtain a rather limited increase of the number of arcs and nodes in the border graph.

Table 3 reports information on the alternative graph for the centralized scheduler, and the local schedulers for 3, 6 and 9 areas. Column 1 reports the time horizon of reference (in minutes), Columns 2-5 the average number of scheduled trains, fixed arcs, nodes and alternative pairs of the associated alternative graph of each local area. When enlarging

Table 2: Border graph properties for each division.

Time Horiz	3-Area division		6-Area division		9-Area division	
	Arcs	Nodes	Arcs	Nodes	Arcs	Nodes
30 min	124	44	299	102	415	150
60 min	407	86	1044	208	1372	300
90 min	852	128	2189	310	2885	450

the time horizon, we obtain a linear growth of the number of trains and nodes while the number of alternative pairs increases quadratically.

Table 3: Alternative graph properties for each scheduler.

Time Horiz	Trains	Fixed Arcs	Nodes	Alternative Pairs
Centralized Scheduler				
30 min	99	3508	3081	3019
60 min	154	6968	6136	14528
90 min	205	10398	9171	34660
3-Area Local Schedulers				
30 min	40	1207	1055	1026
60 min	65	2398	2102	4935
90 min	89	3577	3141	11773
6-Area Local Schedulers				
30 min	25	628	550	529
60 min	44	1257	1096	2544
90 min	62	1877	1638	6065
9-Area Local Schedulers				
30 min	20	432	374	355
60 min	34	858	745	1705
90 min	49	1283	1113	4068

We consider random variations of the entrance times of all trains running in the network. The real-life stochasticity of train operations is modeled by the statistical fitting procedure of the Weibull distribution described in [8]. We analyze a total amount of 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. Trains experience a deviation on their entrance time in the overall network that we call "small perturbations" in this section. We also analyze "large perturbations" that are obtained by doubling the scale parameter of the Weibull distribution. The latter perturbations are artificially generated in order to study the effects of the dispatching and coordination procedures in the presence of more disturbed traffic situations.

Table 4: Description of the entrance delays.

Time Horiz	Small Perturbations		Large Perturbations	
	Max Delay (s)	Avg Delay (s)	Max Delay (s)	Avg Delay (s)
30 min	523	21.7	1351	301.5
60 min	558	22.7	1377	295.9
90 min	567	17.7	1485	277.2

The entrance delays are reported in Table 4 for three time horizons of traffic prediction. Each row of Table 4 reports the average value (in seconds) of the maximum and

average entrance delays over 5 different delay instances. Since there are 2 sets of timetable perturbations (small and large) and 3 time horizons of traffic prediction (30, 60 and 90 minutes), we generated 30 problem instances in total.

4.2 Tuning of JGH parameters

In a preliminary test phase, we tested various JGH parameters on 5 instances of the centralized problem with small perturbations and 3600 seconds of traffic predictions. Figure 9 reports the percentage of globally feasible solutions and the computation time (in seconds) for each JGH configuration.

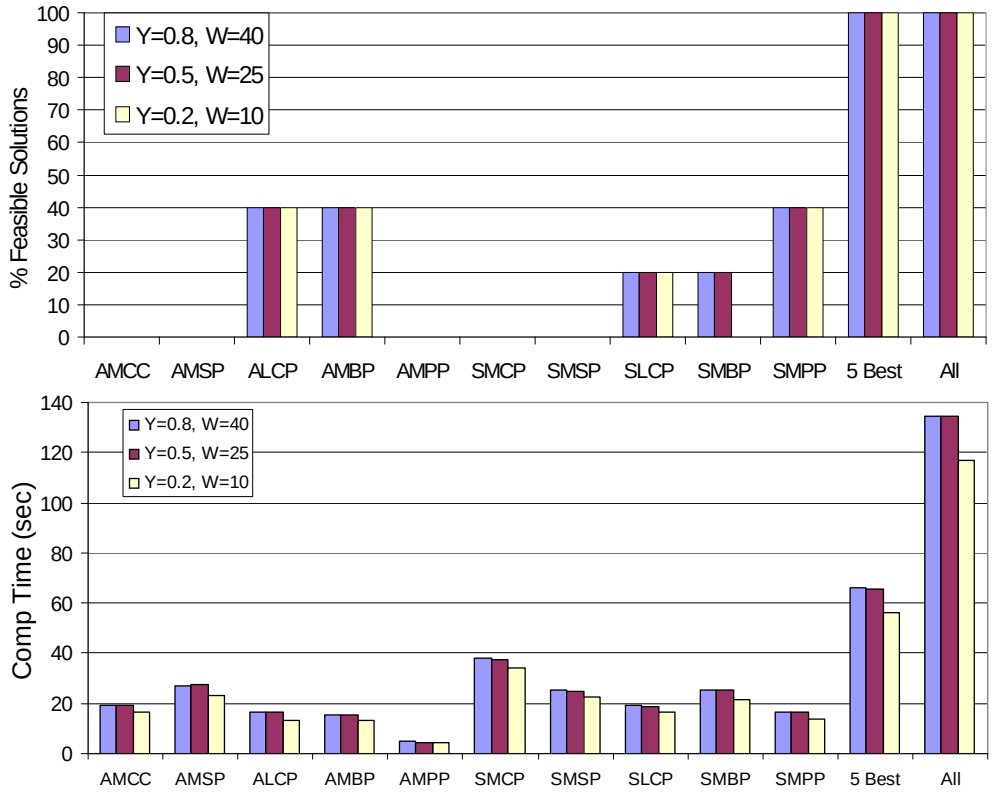

Figure 9: Feasibility (top) and computation time (bottom) of JGH configurations.

In Figure 9, we considered all 10 insertion criteria derived from the AGH heuristics. Since only 5 those criteria (i.e., ALCP, AMBP, SLCP, SMBP and SMPP) were able to compute a feasible insertion for each tested instance, we grouped them under the name "best 5". The whole set of 10 criteria is considered under the name "all". Since "best 5" is faster than "all" and reaches 100% feasibility, we take "best 5" as the best configuration.

Concerning the values Y and W, we tried three combinations: (Y=0.8, W=40); (Y=0.5, W=25) and (Y=0.2, W=10). From our computational results, the configuration with Y=0.2 and W=10 is the fastest setup.

4.3 Timetable perturbations

This subsection presents the performance of 12 combinations of the six dispatching algorithms described in Section 3 with various network divisions. Figure 10 reports the percentage of globally feasible solutions found by the 12 combinations for the management of the overall network. Each algorithm is evaluated over all the instances of Table 4. The algorithms are defined by the two-field code α β, where α is the algorithm and β is the number of local areas. Clearly, $\beta = 1$ requires only the centralized scheduler, otherwise we use the coordination system plus the local schedulers. Regarding the AGH family, we only show the performance of the two algorithms with the highest percentage of feasible solutions, that are AMCC and SMBP.

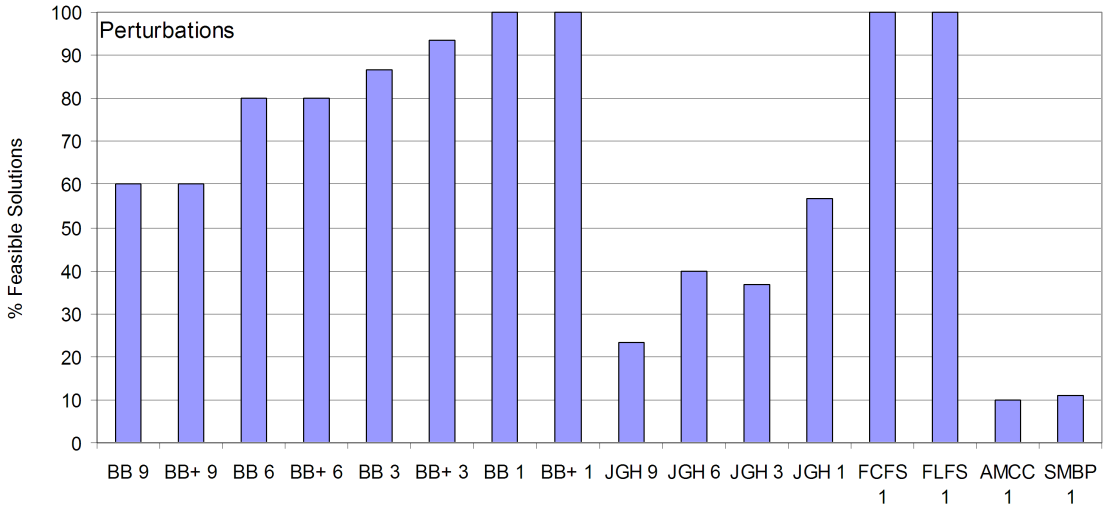

Figure 10: Feasibility of the algorithms for the timetable perturbations.

From the results of Figure 10 we decide to restrict the detailed assessment of our computational results to a subset of scheduling algorithms as follows. Concerning the algorithms for the centralized scheduler, the two practical dispatching rules (FCFS 1 and FLFS 1) are always able to compute a feasible solution for this set of instances, but we only select FCFS 1 since the two rules present very similar results in terms of solution quality. The other heuristics (AMCC 1, SMBP 1 and JGH 1) find less feasible solutions. Since the two best heuristics of the AGH family show poor performance, we decided to skip them from a detailed analysis. Both the versions of our branch and bound algorithm (BB 1 and BB+ 1) are taken since they offer good solutions in the given time limit of computation. In the distributed architecture, for further analysis we select only BB+ that offers more globally feasible solutions compared to both BB and JGH.

Tables 5 and 6 report more information on the selected algorithms for the small and large perturbations, respectively. Each row of these tables presents the average result over 5 delay instances for a given algorithm and time horizon of traffic prediction. The best value of each row is emphasized in bold. For each algorithm we show the percentage of globally feasible solutions, the percentage of globally optimal solutions, the maximum and average consecutive delays (in seconds) and the computation time (in seconds). For the branch and bound algorithms of the centralized scheduler (i.e., BB 1 and BB+ 1), the latter indicator is the time to compute the best solution. Under the percentage of optimal

Table 5: Performance of the procedures: Small perturbations.

Time Hor	Indicator	FCFS 1	JGH 1	BB 1	BB+ 1	BB+ 3	BB+ 6	BB+ 9
30 min	Feasible Solut (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**
30 min	Optimal Solut (%)	0	**60**	**60**	**60**	**60**	**60**	40
30 min	Max Cons Delay (s)	512	**189**	253	**189**	**189**	**189**	191
30 min	Avg Cons Delay (s)	12.2	5.8	7.3	5.7	4.9	**4.1**	4.4
30 min	Comp Time (s)	**0.2**	2	1	3	16	8	16
60 min	Feasible Solut (%)	**100**	**100**	**100**	**100**	**100**	**100**	80
60 min	Optimal Solut (%)	0	**40**	**40**	**40**	**40**	**40**	**40**
60 min	Max Cons Delay (s)	770	286	541	286	**283**	301	341
60 min	Avg Cons Delay (s)	16.4	9.7	12.3	9.7	6.5	**5.8**	8.2
60 min	Comp Time (s)	**0.2**	62	37	67	30	44	108
90 min	Feasible Solut (%)	**100**	20	**100**	**100**	**100**	**100**	40
90 min	Optimal Solut (%)	0	0	0	0	0	0	0
90 min	Max Cons Delay (s)	833	783	769	718	606	**480**	641
90 min	Avg Cons Delay (s)	19.3	18.4	17.7	17.2	10.6	**7.9**	13.3
90 min	Comp Time (s)	**1**	279	34	288	102	168	246

solutions, we report the percentage of instances for which the proven optimum has been found with respect to the total number of instances. The proven optimum is only known for those instances for which a branch and bound algorithm of the centralized scheduler is able to find the optimal solution in the given time limit of computation. The percentage of optimal solutions is indeed a lower bound on the real value.

From the results of Table 5, all the algorithms but JGH 1 and BB+ 9 are able to compute a globally feasible solution. When JGH 1 returns no solution, we impose a failure penalty for the maximum and average consecutive delays equal to the worst solution value found by the other algorithms. In order to compare the different results, we consider the same type of failure penalty for the results of Table 6.

Table 6: Performance of the procedures: Large perturbations.

Time Hor	Indicator	FCFS 1	JGH 1	BB 1	BB+ 1	BB+ 3	BB+ 6	BB+ 9
30 min	Feasible Solut (%)	**100**	60	**100**	**100**	**100**	**100**	**100**
30 min	Optimal Solut (%)	0	0	**60**	**60**	**60**	**60**	0
30 min	Max Cons Delay (s)	492	299	286	249	**231**	243	271
30 min	Avg Cons Delay (s)	23.5	20.0	17.9	18.9	16.0	12.8	**11.3**
30 min	Comp Time (s)	**0.2**	5	3	6	10	11	18
60 min	Feasible Solut (%)	**100**	60	**100**	**100**	60	40	40
60 min	Optimal Solut (%)	0	0	0	0	0	0	0
60 min	Max Cons Delay (s)	714	604	714	**586**	731	665	613
60 min	Avg Cons Delay (s)	35.1	39.4	35.1	39.3	34.7	**33.6**	33.9
60 min	Comp Time (s)	**0.4**	165	**0.4**	172	63	115	127
90 min	Feasible Solut (%)	**100**	0	**100**	**100**	**100**	40	0
90 min	Optimal Solut (%)	0	0	0	0	0	0	0
90 min	Max Cons Delay (s)	960	-	960	960	**770**	968	-
90 min	Avg Cons Delay (s)	40.3	-	40.3	40.3	**29.3**	35.3	-
90 min	Comp Time (s)	**1**	-	**1**	**1**	111	167	-

For both tables, the distributed approaches BB+ 3 and BB+ 6 find the same percentage of optimal solutions compared to the branch and bound algorithms of the centralized approach (i.e., BB 1 and BB+ 1). The three distributed approaches (i.e., BB+ 3, BB+ 6

and BB+ 9) result always in an average consecutive delay less than when considering the other algorithms. The latter result shows that the distributed management of the whole network can generate better quality solutions. However, FCFS 1, BB 1 and BB+ 1 are able to find more feasible solutions than the distributed approaches, since the latter approaches require to be coordinated by imposing additional constraints at the area borders. For this reason, BB+ 6 is able to compute less globally feasible solutions than BB+ 3, and BB+ 9 results in worse feasibility rates since the coordination problem becomes more difficult, specially for large time horizons. We also observe that the maximum consecutive delay of the FCFS 1 solutions is 50% larger than the best one obtained by the other algorithms. Differently, JGH 1 is the worst algorithm in terms of the percentage of feasible solutions but improves considerably the performance of the branch and bound algorithm, see for example BB+ 1 versus BB 1.

Concerning the computation time, FCFS 1 is clearly the fastest algorithm in both tables, even for large time horizons. The other algorithms require a few seconds in order to compute/search for a solution for the 30-minute time horizon. In general, computation time increases with the time horizon of traffic prediction: the distributed approaches BB+ 3 and BB+ 6 find a globally feasible solution in less than 3 minutes while BB+ 9 and all the centralized approaches need more time for the 90-minute time horizon. The branch and bound algorithm for the one-area problem finds its best solution after a short computation time and it is not able to improve this solution during the rest of its search. In particular, JGH 1 is quite time consuming for large time horizons and finds no feasible solutions within the given time limit of computation for the 90-minute time horizon.

4.4 Disrupted traffic situation

We next study how to reschedule trains in case of an infrastructure disruption that requires heavy modifications of the train routes in the timetable. A blockage of a single track is simulated on the double track line that connects Utrecht to Den Bosch, between the stations of Zaltbommel and Den Bosch. Trains of both traffic directions have to run on the other track or have to be globally rerouted (see Figure 11).

From an experimental point of view, the disruption of Figure 11 allows the assessment of the algorithms in a situation characterized by reduced capacity on the line Utrecht-Den Bosch and more dense traffic on the line Den Bosch-Nijmegen-Arnhem-Utrecht.

In the original timetable, 12 trains per hour (6 per direction) are scheduled on the disrupted line, four of which are local services between Utrecht and Den Bosch while the other eight are intercity services connecting the northern region of the country with the southern region. We apply the following modification of the train routes in order to recover the disrupted scenario: Four intercity trains (2 trains per hour per direction) and four local trains (2 trains per hour per direction) are still scheduled on the line Utrecht - Den Bosch. In the vicinity of the disruption, the trains are locally rerouted for a stretch of 6 km, along the only available track that now serves a bidirectional flow. This results in a shortage of capacity that leads to delays propagating throughout the network. The other four intercity trains (2 train per hour per direction) are globally rerouted, i.e., they change their routes but keep the same origin and destination stations. The latter trains are rerouted via the line going to Nijmegen and Arnhem, or vice versa. The running time required for the alternative trip between Utrecht and Den Bosch is around 40 minutes longer than the original trip time, which is 30-minute long.

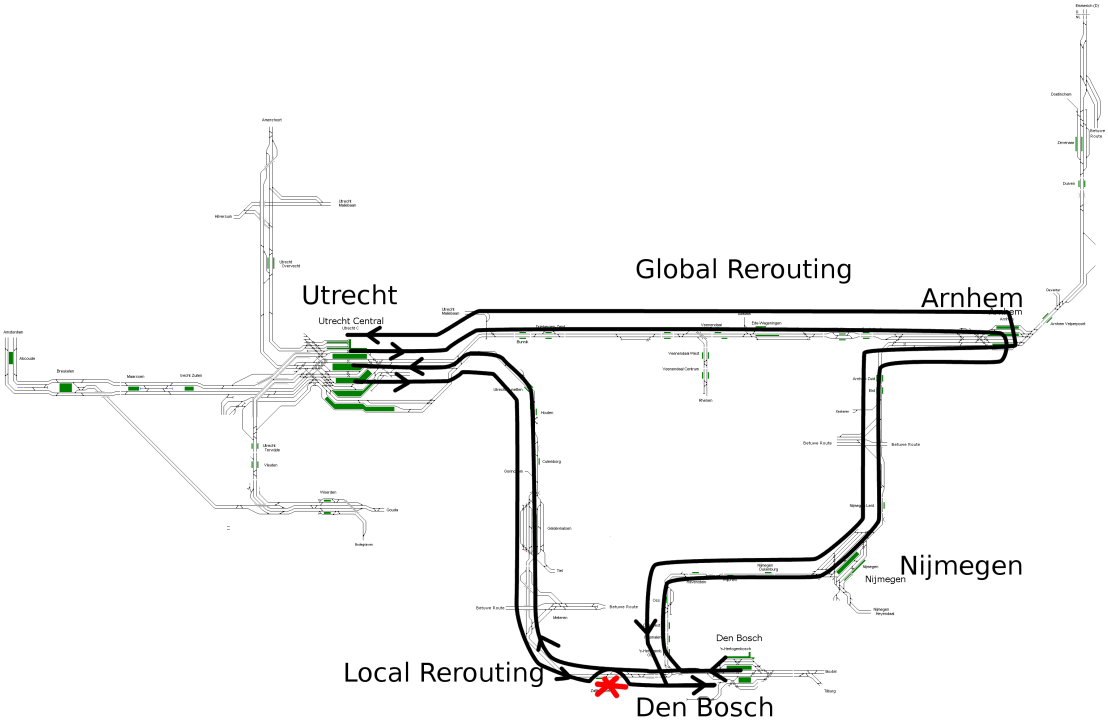

Figure 11: Disrupted timetable with local and global rerouting.

A graphical representation of the two timetables, original and disrupted, is reported in Figure 12. Every solid line indicates two trains running per hour per direction on a specific line. Light green lines are local services and dark blue lines are intercity services. The dotted line shows one international service scheduled per hour. The track blockage is represented by the cross between the stations of Geldermalsen and Den Bosch.

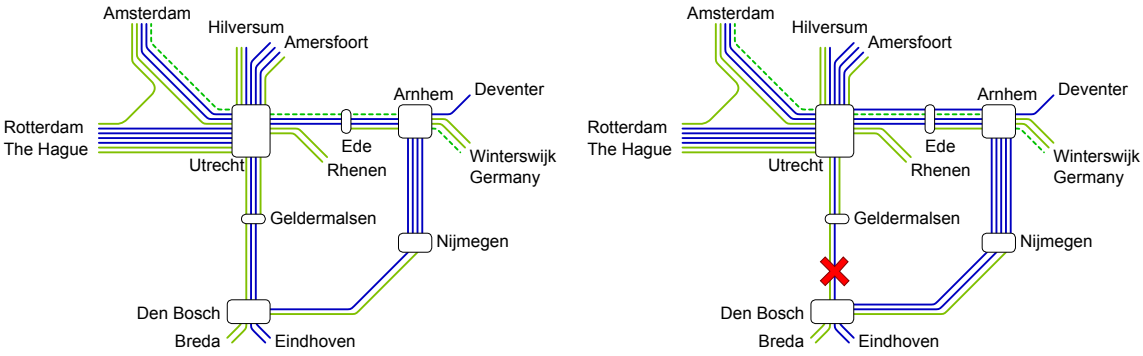

Figure 12: Available train services for the two traffic situations.

Figure 13 reports the performance of all the algorithms in delivering globally feasible solutions for the disrupted situation of Figure 12 and under the timetable perturbations described in Section 4.1. We limit our analysis of each algorithm to the 20 instances of Table 4 with 30-minute and 60-minute time horizons. The instances of the 90-minute time horizon are not considered because in this case the algorithms find a few feasible solutions only. In fact, the traffic management of the disrupted situation is more difficult compared to the original situation, since there is a serious risk of deadlock due to the

larger number of trains that share the single track section in both traffic directions.

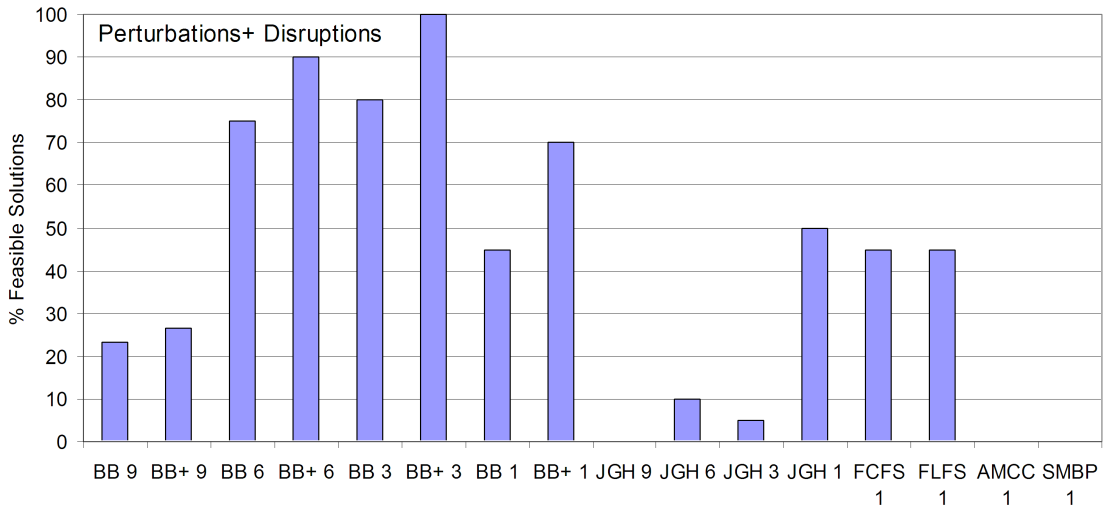

Figure 13: Feasibility of the algorithms for the disrupted situation.

We conclude from Figure 13 that in the disrupted situation the distributed approaches perform better than the centralized ones. Specifically, BB+ 3 is always able to compute a globally feasible solution for all the instances. Among the centralized approaches, BB+ 1 and JGH 1 obtain more feasible solutions than the others.

Tables 7 and 8 give more information on the same subset of algorithms reported in Tables 5 and 6. Each row of the new tables presents the average result over 5 delay instances for a given algorithm and time horizon of traffic prediction. The best value of each row is again emphasized in bold. For each algorithm we show the percentage of globally feasible solutions, the maximum and average consecutive delays (in seconds) and the computation time (in seconds). For the branch and bound algorithms of the centralized scheduler (i.e., BB 1 and BB+ 1), the latter indicator is the time to compute the best solution, if any, or the given time to search for a feasible solution. We do not report the percentage of globally optimal solutions since in this case BB 1 and BB+ 1 were not able to prove optimality within the time limit of computation. If one algorithm does not find a globally feasible solution for one instance, we consider the same type of failure penalty used in Tables 5 and 6.

Table 7: Performance of the procedures: Small perturbations and disruption.

Time Hor	Indicator	FCFS 1	JGH 1	BB 1	BB+ 1	BB+ 3	BB+ 6	BB+ 9
30 min	Feasible Solut (%)	**100**	80	**100**	**100**	**100**	**100**	40
30 min	Max Cons Delay (s)	700	577	688	577	688	**390**	598
30 min	Avg Cons Delay (s)	19.7	24.9	18.8	24.9	12.3	**9.0**	19.5
30 min	Comp Time (s)	**0.1**	8	0.3	10	30	16	48
60 min	Feasible Solut (%)	20	20	20	40	**100**	**100**	40
60 min	Max Cons Delay (s)	1225	1133	1225	1133	1225	**812**	1069
60 min	Avg Cons Delay (s)	24.5	24.6	24.5	24.5	17.8	**12.6**	22.8
60 min	Comp Time (s)	**0.4**	253	108	280	78	119	160

From the results of Table 7, the distributed approaches are always able to compute a globally feasible solution for small perturbations. The centralized approaches generate

feasible solutions for all the instances with 30-minute time horizon but often fail with larger instances. Regarding to the solution quality, for this set of instances the best algorithm is BB+ 6 and requires, on average, less than two minutes of computation.

Table 8: Performance of the procedures: Large perturbations and disruption.

Time Hor	Indicator	FCFS 1	JGH 1	BB 1	BB+ 1	BB+ 3	BB+ 6	BB+ 9
30 min	Feasible Solut (%)	40	60	40	80	**100**	**100**	80
30 min	Max Cons Delay (s)	555	467	544	450	393	**346**	**346**
30 min	Avg Cons Delay (s)	28.8	31.8	28.7	28.7	23.4	**17.7**	18.8
30 min	Comp Time (s)	**0.2**	9	180	67	45	17	29
60 min	Feasible Solut (%)	20	40	20	60	**100**	60	0
60 min	Max Cons Delay (s)	1137	**952**	1137	**952**	1137	1009	-
60 min	Avg Cons Delay (s)	59.9	60.1	59.9	59.9	**34.7**	43.5	-
60 min	Comp Time (s)	**0.2**	266	105	293	80	142	-

The algorithms show a slightly similar performance in Table 8 compared to Table 7. Again, the distributed approaches BB+ 3 and BB+ 6 yield a larger number of globally feasible solutions compared to the centralized approaches, even if this number is lower than in Table 7 due to the larger entrance delays. BB+ 9 gives lower feasibility rates compared to the other distributed approaches. The average consecutive delay is also better minimized by the distributed approaches with respect to the centralized ones. On the other hand, the maximum consecutive delay obtained by JGH 1 (and BB+ 1) for the instances of the 60-minute time horizon is smaller than the one of BB+ 3 and BB+ 6. Finally, since the computation time of JGH 1 and BB+ 1 is more than three minutes for the 60-minute time horizon, these algorithms should only be applied for shorter time horizons during disrupted operations.

4.5 Discussion

The computational experiments report the performance of the centralized and distributed approaches for various traffic disturbances. The limits of the algorithms are evaluated in terms of their ability to find globally feasible solutions for time horizons of traffic prediction of increasing length. On the whole, for the original traffic situation and for any type of timetable perturbation the branch and bound algorithm of the centralized approach and the distributed approaches can easily compute a globally feasible solution in a few seconds for small instances, with 30-minute time horizon. The new scheduling algorithm (JGH) is quite effective in finding good solutions even if this often fails in finding a feasible solution. The combination of JGH with BB (i.e., BB+) clearly outperforms both JGH and BB in terms of number of feasible solutions and solution quality. In this practical case, the solutions are also often globally optimal, and the distributed approach outperforms the centralized one in terms of average consecutive delay minimization. The problem becomes considerably hard to tackle as the magnitude of the time horizon increases. Increasing the number of areas considered in the distributed approach increases the complexity for the coordination problem, since the percentage of feasible solutions found by the proposed heuristic decreases accordingly.

When dealing with the disrupted traffic situation, only using a distributed approach into a limited number of areas it is possible to compute a feasible solution for all the timetable perturbation instances. This is a consequence of the lack of spare capacity in

the network that causes a major propagation of the entrance delays. Specifically, the local rerouting of trains generates bidirectional traffic on the single track adjacent to the disrupted track (nearby Zaltbommel), while the global rerouting requires more trains sharing the block sections of the alternative routes. Centralized algorithms experience increasing difficulty in managing the increasing complexity of the train scheduling problem and the high probability of conflicts and deadlocks. The distributed approach is the most suitable to reschedule trains after the disruption and this is quite effective in dealing with local and global reroutings. However, the problem of coordinating multiple local scheduling solutions is still a complicated issue and the probability of finding a feasible solution decreases for an increasing number of local areas. For instance, the distribution into 9 areas results in feasibility rates comparable or even worse than those of the centralized approaches. Nevertheless, we observe that when distributed algorithms do not find a globally feasible solution this is always due to one or more local infeasibilities, i.e., the time limit of computation is never reached for the coordination procedure.

The study on large time horizons is useful in order to quantify the propagation of train delays in the network. When enlarging the time horizon up to 60 minutes, there is a serious increase of the consecutive delays, specially in the blocked track case. In the traffic situation without disruption, the 90-minute time horizon seems to be less perturbed than the 60-minute time horizon since the entrance delays are progressively absorbed by the time reserves of the timetable in the second hour of traffic prediction.

5 Conclusions and future research

This paper compares centralized and distributed approaches for managing the disturbance handling process accurately. We propose an extensive study of dispatching and coordination algorithms in terms of feasibility, quality and computation time. A new dispatching algorithm, JGH, is proposed to schedule one train per time in the centralized approach. We also extend a distributed approach for the coordination of two areas to the general case of k areas. From the computational results we conclude that when dealing with short time horizons of traffic prediction, the proposed algorithms are able to compute good quality solutions in a very short time of computation. In case of larger time horizons of traffic prediction, severe traffic disturbances and blocked tracks, the distributed approach presents a better feasibility performance than the centralized one.

Further research should be dedicated to a number of issues: (i) development of more robust coordination algorithms for a large number of areas; (ii) development of coordination algorithms able to drive local schedulers towards globally optimal other than feasible solutions; (iii) analysis of different railway network decompositions, disrupted traffic situations and alternative timetables; (iv) use of the proposed approaches in order to validate draft country-wide timetables at the detailed level of signals and block sections; (v) multi-objective approaches for taking multi-criteria train scheduling decisions, considering e.g. the objectives of infrastructure managers and train operating companies.

References

[1] Cheng, Y.-H. and Yang, L.-A. (2009) A Fuzzy Petri Nets approach for railway traffic control in case of abnormality: Evidence from Taiwan railway system. *Expert Systems*

with Applications **36(4)** 8040–8048.

[2] Chou, Y.H., Weston, P.F. and Roberts, C. (2007) Dynamic Distributed Control for Realtime Rescheduling of Railway Networks. In: I.A. Hansen, A. Radtke, J. Pachl and E. Wendler (editors), Proceedings of the 2nd International Seminar on Railway Operations Modelling and Analysis, Hannover, Germany.

[3] Chou, Y.H., Weston, P.F. and Roberts, C. (2009) Collaborative Rescheduling in a Distributed Railway Control System. In: I.A. Hansen, E. Wendler, U. Weidmann, M. Luethi, J. Rodriguez, S. Ricci and L.G. Kroon (editors), Proceedings of the 3rd International Seminar on Railway Operations Modelling and Analysis, Zurich, Switzerland.

[4] Coffman, E.G., Elphick, M.J. and Shoshani, A. (1971) System deadlocks. *Computing Surveys* **3(1)** 67–78.

[5] Corman, F., D'Ariano, A., Pacciarelli, D. and Pranzo, M. (2010) A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B* **44(1)** 175–192.

[6] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M. (2011). Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport* **2(3)** 219–247.

[7] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M. (2009) Centralized versus distributed approaches to railway traffic control. In: H.K. Lo, W. Lam, S.C. Wong and J. Leung (editors), Proceedings of 11th International Conference on Advanced Systems for Public Transport, Hong Kong.

[8] Corman, F., D'Ariano, A., Pranzo, M., Hansen, I.A. (2011) Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transportation Planning and Technology*, to appear.

[9] D'Ariano, A. (2008) Improving Real-Time Train Dispatching: Models, Algorithms and Applications. PhD Thesis, TRAIL Thesis Series T2008/6, The Netherlands.

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

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

[12] D'Ariano, A. and Pranzo, M. (2009) An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. *Networks and Spatial Economics* **9(1)** 63-84.

[13] Fay, A. (2000). A fuzzy knowledge-based system for railway traffic control. *Engineering Applications of Artificial Intelligence* **13(6)** 719–729.

[14] Hansen, I.A. and Pachl, J. (2008) Railway Timetable and Traffic: Analysis, Modelling and Simulation. Eurailpress, Hamburg, Germany.

[15] Iyer, R.V. and Gosh, S. (1995) DARYN, A distributed decision-making algorithm for railway networks: Modeling and simulation. *IEEE Transactions on Vehicular Technology* **44(1)** 180–191.

[16] Jia, L.-M. and Zhang, X.-D. (1994) Distributed intelligent railway traffic control: A fuzzy-decision making-based approach. *Engineering Applications of Artificial Intelligence* **7(3)** 311–319.

[17] Lamma, E., Mello, P. and Milano, M. (1997) A distributed constraint-based scheduler. *Artificial Intelligence in Engineering* **11(2)** 91–105, 1997.

[18] Lee, T.S. and Gosh, S. (2001) Stability of RYNSORD: A decentralized algorithm for railway networks under perturbations. *IEEE Transactions on Vehicular Technology* **50(1)** 287–301.

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

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

[21] Pacciarelli, D. (2003) Deliverable D3: Traffic Regulation and Co-operation Methodologies - code WP4UR_DV_7001_D. Project COMBINE 2 "enhanced COntrol centres for fixed and Moving Block sIgNalling systEms - 2" Number: IST-2001-34705.

[22] Parodi, G., Vernazza, G. and Zunino, F. (1996) Stability and deadlock avoidance in distributed system for traffic control. *IEEE Transactions on Vehicular Technology* **45(4)** 732–743.

[23] Pranzo, M., Meloni, C. and Pacciarelli, D. (2003) A new class of greedy heuristics for job shop scheduling problems. *Lecture Notes in Computer Science* **2647** 223–236.

[24] Rodriguez, J. (2007) A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B* **41(2)** 231–245.

[25] Şahin, İ. (1999) Railway traffic control and train scheduling based on inter-train conflict management. *Transportation Research Part B* **33(7)** 511–534.

[26] Salido, M.A., Abril, M., Barber, F., Ingolotti, L., Tormos, P. and Lova, A. (2007) Domain-dependent distributed models for railway scheduling. *Knowledge-Based Systems* **20(2)** 186–194.

[27] Strotmann, C. (2007) *Railway scheduling problems and their decomposition*. PhD thesis, Universität Osnabrück, Germany.

[28] Törnquist, J. and Persson, J.A. (2007) N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B* **41(3)** 342–362.

[29] Vernazza, G. and Zunino, R. (1990) A distributed intelligence methodology for railway traffic control. *IEEE Transactions on Vehicular Technology* **39(3)** 263–270.

[30] Wegele, S., Slovák, R. and Schnieder, E. (2007) Real-time decision support for optimal dispatching of train operation. In: I.A. Hansen, A. Radtke, J. Pachl and E. Wendler (editors), Proceedings of the 2nd International Seminar on Railway Operations Modelling and Analysis, Hannover, Germany.

[31] Zhu, P. (2001). Betriebliche Leistung von Bahnsystemen unter Störungsbedingungen. PhD thesis, Berichte der Institute für Automatisierungstechnik, TU Braunschweig, Germany. (In German)