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

Reordering and local rerouting strategies to manage train traffic in real-time

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

RT-DIA-126-2008 **Giugno 2008**

(1) Department of Transport and Planning, Delft University of Technology,
Stevinweg, 1 - 2628 CN Delft, The Netherlands.
(2) Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre,
via della vasca navale, 79 - 00146 Roma, Italy.
(3) Dipartimento di Ingegneria dell'Informazione, Università di Siena,
via Roma, 56 - 53100 Siena, Italy.

This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation"

ABSTRACT

Traffic controllers regulate railway traffic by sequencing train movements and setting routes with the aim of ensuring smooth train behaviour and limiting as much as possible train delays. In this paper, we describe the implementation of a real-time traffic management system, called ROMA (Railway traffic Optimization by Means of Alternative graphs), to support controllers in the everyday task of managing disturbances. We make use of a branch and bound algorithm for sequencing train movements, while a local search algorithm is developed for rerouting optimization purposes. The compound problem of routing and sequencing trains is approached iteratively computing an optimal train sequencing for given train routes, and then improving this solution by locally rerouting some trains. An extensive computational study is carried out, based on a dispatching area of the Dutch railway network. We study practical size instances, and include in the model important operational constraints, including rolling stock and passenger connections. Different types of disturbances are analysed, including train delays and blocked tracks. Comparison with common dispatching practice shows the high potential of the system as an effective support tool to improve punctuality.

1 Introduction

The continuous growth of passenger and freight railway traffic is increasing the pressure on European railway companies, which must face the challenge of providing a satisfactory level of service with a limited possibility of changing the existing infrastructure. Effective timetable design and real-time traffic management policies play therefore a key role in this context. In fact, there is an extensive literature on decision support tools for the timetable design problem [8]. This is usually a long procedure, involving intensive negotiation among different stakeholders and taking advantage from the availability of sophisticated decision support systems (see, e.g. the Dutch timetable design system DONS [18]). On the opposite, real-time traffic management received little attention in the literature, and is still mainly under control of human dispatchers, whose computer support is often limited to graphical interfaces and simple automatic route setting systems. Possible reasons are the inherent complexity of the real-time process and the strict time limits for taking decisions, which leave small margins to computerized decision support systems. Another reason is that the effects of real time processes are often limited in time and space, and depend on the quality of the timetable. However, the quality offered by real-time control coincides with the actual level of railway service, and any improvement to this process has a direct impact on the customer satisfaction. For these reasons, this paper focuses on the development of models and algorithms for real-time railway traffic management.

The main problem faced by traffic controllers is to recover disturbances occurring in real-time, such as train delays or temporary unavailability of some routes, which make the timetable infeasible. This problem is called Conflict Detection and Resolution (CDR) problem, which consists of modifying the timetable in order to make it compatible with the real-time traffic situation. Similar real-time problems are faced also in other transportation contexts, such as the air traffic control [4, 5].

Possible control actions include changing dwell times at scheduled stops, changing train speeds along lines or train orders at junctions, stations and passing points. Other control actions involve major modifications such as changing train routes or even canceling journeys. We address the CDR problem defining feasible train routes and scheduling exact arrival/departure times at stations as well as at a set of relevant points in the network, with the objective of optimizing train punctuality.

Due to the strict time limit available for computing a conflict-free timetable, traffic controllers usually perform manually only a few changes in the timetable. Experienced dispatchers have developed strategies that allow them foreseeing possible disruptions well in advance to take compensatory control actions based on local information. In general, minor changes are preferred to extensive rescheduling in order to alter as little as possible the original timetable. Simple computerized systems have been developed to support dispatchers in their activity. For example, the Dutch automatic route setting system ARI (Automatische Rijweg Instelling, described in [3]) may perform automatically some CDR actions. However, such systems do not provide an effective support when dealing with strong disturbances, and extensive rescheduling is necessary to obtain feasibility. This paper describes models and algorithms for supporting this activity when dealing with several late trains and even in presence of strong disturbances, such as blocked tracks. Precisely, we report on the implementation of a real-time traffic management system, called ROMA (Railway traffic Optimization by Means of Alternative graphs), able to solve the CDR problem in real time within a dispatching area of practical size.

ROMA decomposes the CDR problem into two sub-problems: (i) given a route for each train, define train ordering and timing; (ii) given a solution to sub-problem (i), define new train routes potentially leading to better schedules. The two sub-problems are then solved iteratively until no improvement is possible within a time limit of computation.

Sub-problem (i) is modeled with an alternative graph [21], which allows predicting accurately the future evolution of the railway traffic on the basis of the actual train positions and speeds, signaling and safety system constraints. The problem is modeled as a job shop scheduling problem with no-store constraints. Additional real-world constraints due to passenger satisfaction are also considered, such as minimum transfer time between connected train services. This sub-problem is solved with the truncated branch and bound procedure of D'Ariano et al. [9] with the objective of minimizing the maximum consecutive delay at stations and at a set of relevant points of the dispatching area.

Sub-problem (ii) is solved by a local search approach. A train route is modified if the new solution improves the objective function. We restrict the search to promising routes by exploiting mathematical properties of a solution.

Computational experiments are based on the dispatching area of the Dutch rail network between Utrecht and Den Bosch. Given a perturbation at the border or within the area, a new feasible timetable is produced within a short time period, compatible with real-time purposes. We study the performance of ROMA under severe disturbances and considering relevant real-world constraints. We also present a comparison between the branch and bound procedure and a scheduling algorithm similar to that of the ARI system, currently adopted by Dutch traffic controllers.

The paper is organized as follows: Section 2 reviews recent contributions dealing with train scheduling and routing problems; Section 3 defines the conflict detection and resolution problem and presents the general architecture of ROMA; Section 4 deals with model and optimization algorithms for real-time train scheduling and local rerouting. In Section 5, we describe and evaluate the computational results.

2 Review of the related literature

The optimization of train routing and scheduling is recognized as an interesting possibility to improve railway performance. The survey of Cordeau et al. [8] reviews a large number of papers dealing with different problems arising in timetable design and real-time traffic management. In view of their extensive survey, we limit our review to recent papers dealing with train scheduling and routing problems.

Kroon et al. [19] prove the NP-completeness of the general problem of routing trains through railway stations to design a conflict-free timetable, and show polynomially solvable special cases. Zwaneweld et al. [32] reduce the same problem to a weighted node packing problem, and develop a branch and cut procedure based on dominance rules and valid inequalities, which is able to solve the problem for practical size instances at all Dutch stations. The weighted node packing model is also the basis of other works on the train routing problem. For instance, Caimi et al. [6] address the problem of finding robust routings first by searching a feasible solution with a fixed-point iteration method, and then by looking for better solutions with a local search algorithm.

Delorme et al. [10] model the integrated train routing and scheduling problem with a set packing formulation, that is stronger than the node packing one. They develop a

GRASP (Greedy Randomized Adaptative Search Procedure) metaheuristic and test it on a number of mixed-traffic instances for a junction located in the North of France. Delorme [11] extends the set packing model taking into account different objectives to be considered in lexicographic order and presents a method to measure routing robustness. Gandibleux et al. [15] develop an ant colony metaheuristic to solve the set packing problem and present results based on a junction and a station located in the North of France. For the same problem, Lusby et al. [20] propose a branch and bound procedure, which exploits the mathematical structure of the dual of the linear relaxation in order to efficiently generate good lower bounds.

Carey and Carville [7] deal with the problem of routing and scheduling trains at complicated stations. A combinatorial model is proposed to avoid train conflicts and to minimize schedule deviations on the basis of a weighted combination of costs. Heuristic techniques are designed according to train planners' objectives. Several practical constraints are also considered as minimum time headway distances and platform occupation constraints for multi-platform stations.

Fioole et al. [14] model the problem of assigning the rolling stock to the timetable services, including constraints on departure and arrival time, number of passengers and other constraints on the rolling stock type. They also consider combining and splitting of trains according to the Dutch standard. Multi-objective criteria are adopted, and mixed integer programming techniques are used to solve the problem. The authors were able to compute a significant part of the weekly rolling stock schedule of the Dutch railway.

Higgins et al. [16] formulate the train scheduling problem as a non-linear mixed integer program aiming to the improvement of timetables and to the development of decision support systems for real-time traffic management. Their model is applied to a long single line track. The authors present a branch and bound algorithm based on priority rules and using a shortest path method for estimating the lower bound. The criteria for conflict resolution takes into account train priorities, current train delays and expected remaining delay due to conflicts. Successively, Higgins and Kozan [17] propose several metaheuristics for the real-time train scheduling problem, including local search, genetic algorithms, tabu search and hybrid techniques.

Adenso-Diaz et al. [1] consider the problem of managing real-time timetable disturbances for a regional network. They propose an automated conflict resolution system for the Spanish National Railway Company. A mixed integer programming model is adopted and heuristic solution techniques are developed.

Şahin [28] studies the real-time conflict resolution problem on a single-track railway. Conflicts between trains are solved in the order they appear. An algorithm based on look-ahead strategies predicts potential consecutive delays and takes ordering decisions at merging or crossing points. The problem is formulated as a job shop scheduling problem and the objective is to minimize average consecutive delays.

Oliveira and Smith [25] model the train scheduling problem for a single track railway network as a job shop scheduling problem, and include in the model several additional real world constraints. They point out the importance of considering constraints related to the satisfaction of required train services, such as passenger or rolling stock connections at station. Their objective function is the minimization of the total delay.

Dorfman and Medanic [13] propose a discrete-event model for scheduling trains on a single line and a greedy strategy to obtain sub-optimal schedules. The model behavior is similar to those of human dispatchers. The authors show that adding non-local informa-

tion, the system can also prevent deadlocks. The approach can quickly handle timetable perturbations and performs satisfactorily on three time-preference criteria.

Dessouky et al. [12] describe a train dispatching system based on reordering decisions. A branch and bound procedure has been implemented and deadlock avoidance checks are adopted to reduce the search space. The resulting approach is able to find exact solutions for a single-track network with 14 train routes within two hours of computation time. The development of effective heuristics is addressed for real-time purposes.

Rodriguez [27] proposes a heuristic approach to train routing problem and consequent train reordering problem with operational purposes. The algorithm is tested on a complex rail junction and is able to provide a satisfactory solution within three minutes of computation time for instances up to 24 trains.

Mazzarello and Ottaviani [23] describe the architecture of a real-time traffic management system that has been implemented within the European project COMBINE, in order to test the feasibility of a completely automated system for conflict resolution and speed regulation. The problem is modeled with the alternative graph of Mascis and Pacciarelli [21], and the future evolution of train dynamics is predicted on the basis of a continuous feedback with train position and speeds. Rerouting actions are chosen on the basis of priority rules and ordering decisions are taken with the objective of minimizing a maximum delay. A detailed description of models and algorithms used within COMBINE is presented in [22].

Törnquist and Persson [31] present a model for the train rescheduling problem in a rail network with several merging and crossing points. The problem is formulated as a mixed-integer linear program and solved with commercial software packages. Four strategies are proposed for reducing the solution space, based on restrictions on reordering and rerouting actions. Computational experiments are based on instances with a single delayed train.

Due to the significant complexity of real-time train routing and scheduling problems, even recent approaches are able to provide proven optimal solutions only for small instances or for simple perturbations. However, practical real-time problems may include the presence of several late trains and blocked zones, which require implementing local rerouting actions and train reordering at merging and crossing points. Our approach combines rerouting and rescheduling strategies for solving real-time disturbances in a regional railway network of practical size, including the presence of blocked zones and strong perturbations, which require extensive timetable modifications to reach a feasible schedule. We also analyze the relevant case in which a double track corridor is blocked in one of the two directions, which requires trains traveling in opposite directions to share the only track available.

3 Conflict detection and resolution system

In this section, we formally define the CDR problem and describe the architecture of our CDR system ROMA. In its basic form a railway network is composed of track segments and signals. Signals are used to control railway traffic and to guarantee a safety distance between trains. There are signals before every junction as well as along the lines and inside the stations. A detailed description of different aspects of railway signaling systems and traffic control regulations can be found in [26].

A *block section* is a track segment between two block signals. The *running time* of a

train on a block section is the time required to traverse the block section. This is known in advance since all trains travel at their scheduled speed whenever possible. An additional delay may occur when a train reaches the end of a block section and the subsequent block section is still occupied by another train. The running time of a train on a block section starts when its head (the first axle) enters the section. However, the previous block section remains blocked by the train for a certain amount of time that we call *setup time*. This time takes into account the time between the entrance of the train head in a block section and the exit of its tail (the last axle) from the previous one, plus additional time margins to release the occupied route (see, e.g. [24]).

A *conflict* occurs whenever a train requires a block section which is not available, i.e., occupied by another train or temporarily blocked. Given a timetable, the current infrastructure status with possible blocked tracks and the current train positions and speeds, a conflict detection and resolution system addresses the following problems:

- Conflict detection: check whether the timetable is conflict-free or detect conflicts.
- Conflict resolution: build a new feasible timetable compatible with the status of the network, by defining routes, orders and times for all circulating trains.

The general architecture of ROMA is shown in Figure 1. A human dispatcher can interact with the system adding/removing constraints or changing the timetable. The overall system performs three basic actions: (*i*) `load information`, in charge of loading all necessary data and computing some preprocessing, (*ii*) `disruption recovery`, which assigns an initial feasible route to each train and (*iii*) `real-time optimization`, which computes the final schedule. The next three subsections address each module separately.

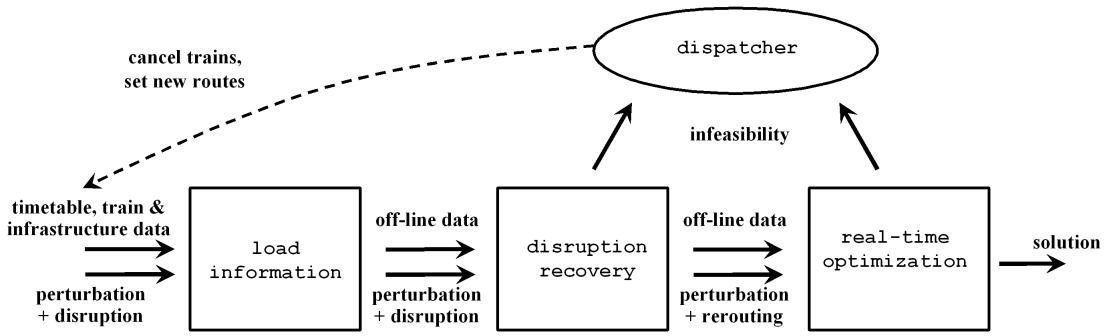

Figure 1: Architecture of ROMA

3.1 Load information

ROMA makes use of the following information. The timetable contains a list of arrival/departure times for a set of relevant points on the network, including all the stations visited by each train. The infrastructure consists of a set of available block sections delimited by signals. Infrastructure data include the status and length of each block section and other characteristics, such as speed limitations and the traversing direction.

The data associated with each train include speed and position at the entrance of the network, acceleration and braking curves (calculated on the basis of traction force/speed

diagrams and maximum speeds) and a list of routing options, in the order given by the dispatcher. A *route* is a sequence of block sections and is feasible if no block section in the route is blocked.

Finally, setup times and running times for each pair (train, block section) are computed by the module on the basis of available rolling stock and infrastructure data.

3.2 Disruption recovery

After the loading phase, the `disruption recovery` module checks if there are blocked zones in the network, which make some train routes infeasible. In this case, this module assigns a new feasible route to each disrupted train, avoiding the blocked zones. This is performed simply sorting the routing options on the basis of a priority list and then choosing the feasible train route with highest priority, called the *default routing*.

If no feasible route is available for a train the system asks for an external support by the human dispatcher. In such cases, emergency timetables are used and train routes are strongly modified, e.g. enabling a train to reverse the running direction.

3.3 Real-time optimization

This module is in charge of computing a first feasible schedule and then looking for better solutions in terms of punctuality. The module architecture is described in Figure 2.

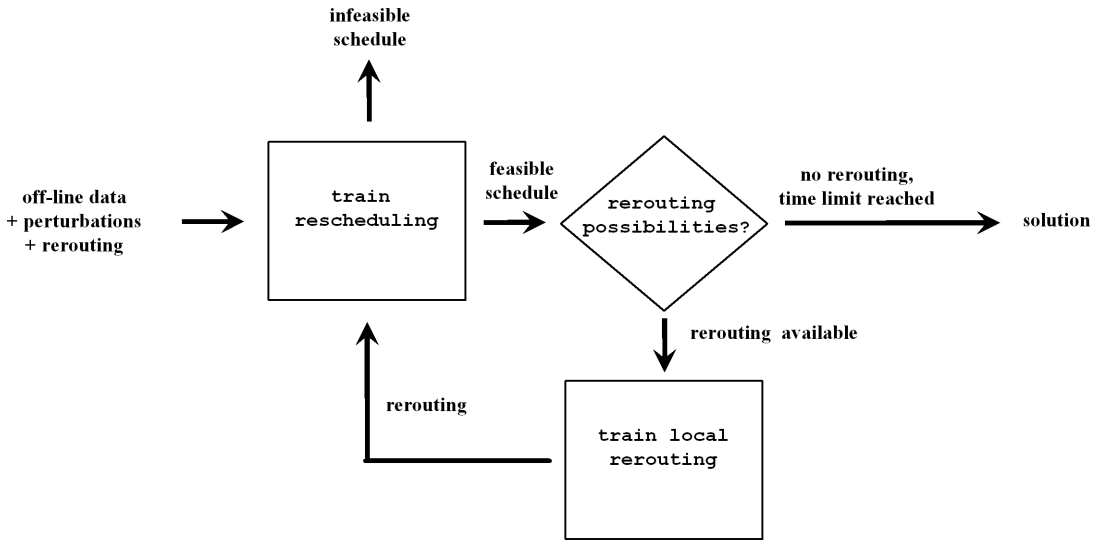

Figure 2: Architecture of the `real-time optimization` module

Given a timetable, a set of routing options associated with each train and the current status of the network, the `train rescheduling` module returns a feasible schedule for the given train routes. Specifically, the first run of this module considers the default routings defined by the `disruption recovery` module. If no feasible schedule is found within a predefined time limit of computation, the human dispatcher is in charge of avoiding deadlocks taking some decisions that are forbidden to the automated system, such as the cancellation of a connection or even a train journey. When a feasible schedule is found, the `train local rerouting` module verifies whether a rerouting option, leading

to a better solution, exists. We limit ourselves to consider local rerouting only, such as parallel corridors along the lines or alternative platforms in the stations. However, our procedure may work as well with more complex rerouting options, provided that these are predefined in a list of available rerouting options. For each changed route, running times and setup times are modified accordingly. Whenever some route is replaced, the `train rescheduling` module computes a new conflict-free timetable by thoroughly rescheduling train movements. The iterative rescheduling and rerouting procedure returns the best solution found when a time limit is reached or no local rerouting improvement is possible. In the following section we introduce the mathematical formulation of the problem and the algorithms used by ROMA.

4 Model and algorithms for traffic optimization

The train scheduling problem can be formulated as a job shop problem with additional constraints [25, 28, 30]. Trains are viewed as jobs passing through a prescribed sequence of block sections, which are viewed as machines. An *operation* o_i corresponds to the traversing of a block section u for a train T_V, i.e., is associated with the pair $< u, T_V >$. The journey of a train T_V is therefore a sequence of operations $< u_1, T_V >, < u_2, T_V >, \ldots, < u_k, T_V >$, and the operation associated with the pair $< u_j, T_V >$ is the successor of $< u_{j-1}, T_V >$ and the predecessor of $< u_{j+1}, T_V >$. The running time p_i of o_i depends on block section characteristics and on the train speed profile, and o_i cannot be interrupted from its starting time t_i to its completion time $t_i + p_i$. Each block section can host at most one operation at a time. The job shop scheduling problem consists of defining starting times for all operations such that each operation of a job starts after the completion of its predecessor and no machine processes two operations simultaneously. Additional constraints of the train scheduling problem include setup times, arrival and departure times, connections between train services and the fact that a job cannot be stored between consecutive operations, called *no-store* constraint. A no-store constraint imposes that a train, having reached the end of a track segment, must wait for the subsequent segment to be available before leaving the current segment, thus preventing other trains from entering it at the same time.

We model the train scheduling problem with an *alternative graph*. This is a triple $\mathcal{G} = (N, F, A)$, where N is a set of nodes, F is a set of fixed arcs, and A is a set of pairs of alternative arcs. Each node corresponds to an operation. A fixed arc (i, k) represents a *precedence relation* $t_k \geq t_i + p_i$ constraining the starting time t_k of o_k with respect to the starting time t_i of its predecessor o_i. In Figure 3, fixed arcs are depicted with solid arrows. Alternative arcs represent the no-store constraints and the constraints that a block section cannot host two trains at the same time. Their lengths represent the setup times. If two operations o_i and o_j require the same block section, there is a potential conflict and a processing order must be defined between them. Specifically, if o_i is processed first, then o_j must wait for the completion of o_i, the starting of the successor of o_i plus the setup time a_{ij} between o_i and o_j. By letting o_k and o_h be the successors of o_i and o_j respectively, we model the two possible processing orders with the pair of arcs $((k, j), (h, i)) \in A$. The length of (k, j) is the setup time a_{ij} between o_i and o_j, while the length of (h, i) is the setup time a_{ji} between o_j and o_i. In the example of Figure 3, alternative arcs are depicted with dashed arrows. Set N includes two dummy operations o_0 and o_n, with zero processing

time, "start" and "finish" respectively. For each operation $o_i \in N$ there is a path in F from o_0 to o_i and from o_i to o_n.

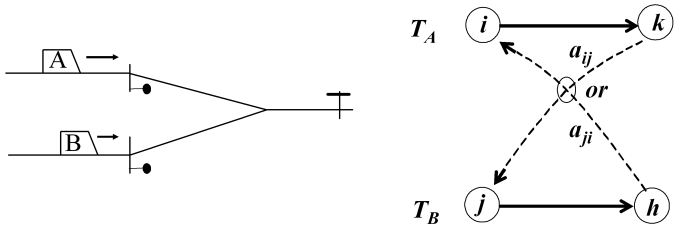

Figure 3: Two trains approaching a conflict point. Nodes i and j represent trains T_A and T_B entering the junction, nodes k and h represent T_A and T_B entering the subsequent block section.

An alternative graph represents all the scheduling alternatives when train routes are defined. If a train can be assigned to different routes, then F is a variable of the problem and A depends on the choice of F. Note that, each arc of F associated with the traversing of a block section has length equal to the running time of the associated train on the associated block section. In fact, this depends also on the train speeds on the previous/following block sections. Changing part of a route for a train may therefore cause adjusting its running time on several block sections that have not been modified. In such cases, we replace with new fixed arcs all the arcs of F that are associated to a different block section or to the same block section but with a different running time. Once F is defined, a *selection* S is a set of arcs obtained from A choosing only one arc from each pair and $G(F, S)$ indicates the graph $(N, F \cup S)$. In the example of Figure 3, if (k, j) is selected, T_A precedes T_B. Conversely, if (h, i) is selected, T_B precedes T_A. If $t_i < t_j < t_k + a_{ij}$ or $t_j < t_i < t_h + a_{ji}$ then a conflict between T_A and T_B is detected and must be solved choosing one of the two arcs.

A selection S is *consistent* if $G(F, S)$ has no positive length cycles. In fact, a positive length cycle represents a deadlock situation (i.e., an operation preceding itself), which is infeasible. In the problem we deal with in this paper, negative and zero length cycles never occur. However, negative and zero length cycles allow to model more general scheduling situations [22]. If S is consistent, the value of a longest path from i to j in $G(F, S)$ is $l^{F,S}(i, j)$. The longest path from the start node 0 to the finish node n in $G(F, S)$ is called the *critical path* $\mathcal{C}(F, S)$ of the graph. A conflict-free schedule is associated with a complete consistent selection on the corresponding graph. Given a set F and an initial selection S^{in}, potentially empty, the objective of the scheduling problem is to find a complete consistent selection S such that $S^{in} \subseteq S$ and such that the length $l^{F,S}(0, n)$ of $\mathcal{C}(F, S)$ is minimized. The selection S^{in} represents the precedence constraints implied by the initial positions of the trains and/or by their order of entrance into the network.

We next model our objective function. Let α_i be the planned arrival time of a train T_A at a relevant point in the timetable, which can be infeasible in case of perturbation. Let τ_i be its earliest possible arrival time computed according to its initial position, initial speed, assigned route and the maximum speed profile (allowed by the train characteristics and infrastructure). Note that τ_i does not take into account possible conflicts with other trains, and therefore is a lower bound on the feasible arrival time t_i of T_A. The value $\max\{0, \tau_i - \alpha_i\}$, called the *initial delay* is then a delay which cannot be recovered. We call *consecutive delay* the quantity $t_i - \max\{\tau_i, \alpha_i\}$, which is the additional delay due to the

solution of conflicts between T_A and the other trains. Adding an arc from o_i to o_n, with length $-\max\{\tau_i, \alpha_i\}$, for each relevant point, we have that $l^{F,S}(0, n)$ equals the maximum consecutive delay.

For the sake of simplicity, in the following figures we only show the relevant point associated to each node and the train associated to each job. Figure 4 shows an example of a network with two relevant points, namely the arrivals of trains T_A and T_B at station platform Q. Here, the two arcs (Q, n) represent the contributions of the two train delays at station platform Q to the objective function. Arcs $(0, 3)$ and $(0, 6)$ constraint the departure times of the two trains to be greater or equal to the ones in the timetable. Horizontal arcs $(Q, 3)$ and $(Q, 6)$ model the dwell times.

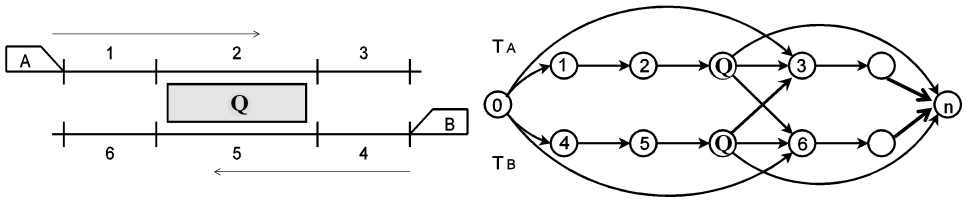

Figure 4: Passenger connections

In the alternative graph of Figure 4 a *passenger connection* is shown. There are two trains, T_A and T_B, stopping at station platform Q. To let passengers moving from one train to another, each train must depart sufficiently later with respect to the other. We model this constraint with two diagonal fixed arcs $(Q, 3)$ and $(Q, 6)$, which constraint the departure time of a train with respect to the arrival time at the platform of the other. In general, the arc length depends on the distance between the two stopping platforms.

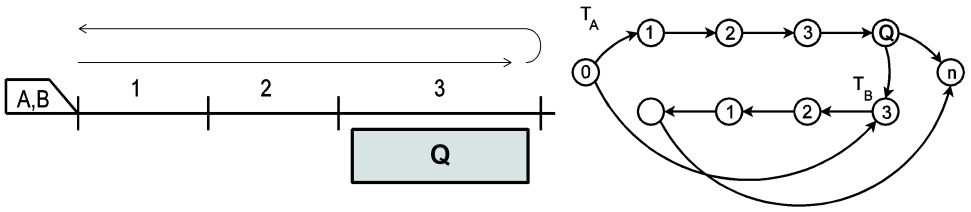

Figure 5: Rolling stock connections

Figure 5 shows an example of *rolling stock connection*. A train stopping at station platform Q and departing in the opposite direction is shown. This constraint is modeled adding a fixed arc from the node associated with the stopping of train T_A at station platform Q, to the node associated with the departure of train T_B, i.e., node 3. The same model can be used for representing more general constraints, such as the case in which part of the rolling stock of train T_A is used to form train T_B or the case when T_A carries part of the crew of T_B.

Due to the inherent complexity of the compound routing/scheduling problem, we produce a solution solving the two problems separately, as in Figure 2. In the scheduling optimization phase we sequence the trains being fixed their respective routes. In the routing optimization phase we assume that the output of the `train rescheduling` module is an optimal schedule, for the given route assigned to each train, and we try to improve

the solution modifying the route of some train. The algorithms for these two phases are described in the following sub-sections.

4.1 Scheduling algorithms with fixed train routes

We briefly describe the scheduling algorithms used in our paper when F is fixed. Two scheduling algorithms have been implemented. The first algorithm is the branch and bound ($B\&B$) described in [9], which is able to solve at optimality large scheduling instances within a short computation time. The second algorithm simulates the practice of traffic management adopted in the Netherlands, which is based on the ARI system [3]. This semi-automated system detects and solves train conflicts automatically when delays are contained in a predefined time-window specified by the dispatcher. In such cases, a precedence relation is automatically assigned according to the following rules:

- If the conflicting trains require the same track segment, then the order in the timetable is kept.
- If the conflicting trains require different incompatible track segments, then precedence is given to the train that requested the route first.

When train delays exceed the time-window, dispatchers must decide the precedence on the basis of a list of *what-if scenarios*, and on their intuition and experience.

In order to evaluate the effectiveness of a completely automated system, we implemented an automated version of the ARI system, simulating the behavior of the dispatchers through the following priority rule. Precedence is given on the basis of the train type (first intercity, then regional and then freight) and, in case of tie, priority is given to the train with the smallest number of scheduled stops after the conflicting point. Clearly, when train delays are contained in the predefined time-window, the two above rules of the ARI system are applied.

4.2 A local search algorithm for routing optimization

We describe the routing optimization procedure to improve the solution provided by the `train rescheduling` module. Recall that the length $l^{F,S}(0,n)$ of the critical path $\mathcal{C}(F,S)$ of the graph is the maximum consecutive delay for that solution. This value can be reduced either changing the train sequencing, i.e., the set S, or modifying the train routes, i.e., the set F. Let $S(F)$ be the sequencing computed by the `train rescheduling` module for fixed F. In this section, we start from $G(F,S)$ and aim at finding a new set F' and therefore a new alternative graph (N,F',A') and a new solution $G(F',S')$. We use the notation $l^{F',S(F)}(0,n)$ to denote the longest path in $G(F',S')$ such that all alternative pairs in $A \cap A'$ are selected as in $S(F)$. With this notation, F' is preferred to F if $l^{F',S(F)}(0,n) < l^{F,S(F)}(0,n)$.

Given a set F and two consistent selections S and S', let $\mathcal{C}(F,S)$ be a critical path of $G(F,S)$ and $P \subseteq S$ be the set of arcs of S belonging to $\mathcal{C}(F,S)$. A well known property of the job shop scheduling problem [2] states that if $P \subseteq S'$ then $l^{F,S'}(0,n) \geq l^{F,S}(0,n)$. We next restate this property for the problem in which set S is fixed and set F can be modified. Given a *rerouting option* δ, i.e. a chain of fixed arcs of F, we denote with $l(\delta)$ its length.

Property 4.1 *Consider a set F and the corresponding critical path $\mathcal{C}(F, S(F))$. Let F' be a new arc set obtained from F replacing a rerouting option $\delta_1 \subseteq F$ for some train with a different rerouting option δ_2 with the same terminal nodes of δ_1. If $\delta_1 \subseteq \mathcal{C}(F, S(F))$ and $l(\delta_1) \leq l(\delta_2)$, then $l^{F',S(F)}(0, n) \geq l^{F,S(F)}(0, n)$ and F is preferred to F'.*

Proof. Simply observe that the new solution $G(F', S(F))$ contains the path $\mathcal{C}'$ obtained by $\mathcal{C}(F, S(F))$ substituting rerouting option δ_1 with rerouting option δ_2. Denoting with $l(\mathcal{C}')$ the length of $\mathcal{C}'$, and since the length of δ_1 is smaller or equal to the length of δ_2, we have $l^{F',S(F)}(0, n) \geq l(\mathcal{C}') \geq l^{F,S(F)}(0, n)$. ∎

When a new set F' is built, the solution can be further improved by rescheduling train operations for fixed routing. However, changing a route which is not contained in the critical path and then computing an optimal schedule may also lead to better solutions. This section focuses on a set of promising route modifications, strictly containing the changes on the critical path. To this aim, let us introduce the following notation. Given a node $i \in \mathcal{C}(F, S)$, we call *ramification* $R(i)$ all the sequences of fixed arcs from 0 to any node $j \in N$ such that there is a path from j to i in $G(F, S(F))$. We call *ramified critical path* the set $\mathcal{R}(F, S) = \bigcup_{i \in \mathcal{C}(F,S)}[R(i)] \cup \mathcal{C}(F, S)$, i.e., the critical path plus the sets $R(i)$ computed for all the nodes of the critical path.

Figure 6 shows a small railway network with three trains (denoted as T_A, T_B and T_C) and two alternative graphs associated with different routes for train T_A. In each graph, the critical path and ramified critical path are depicted with bold black and bold grey arrows, respectively. The selected alternative arcs are depicted with dotted arrows.

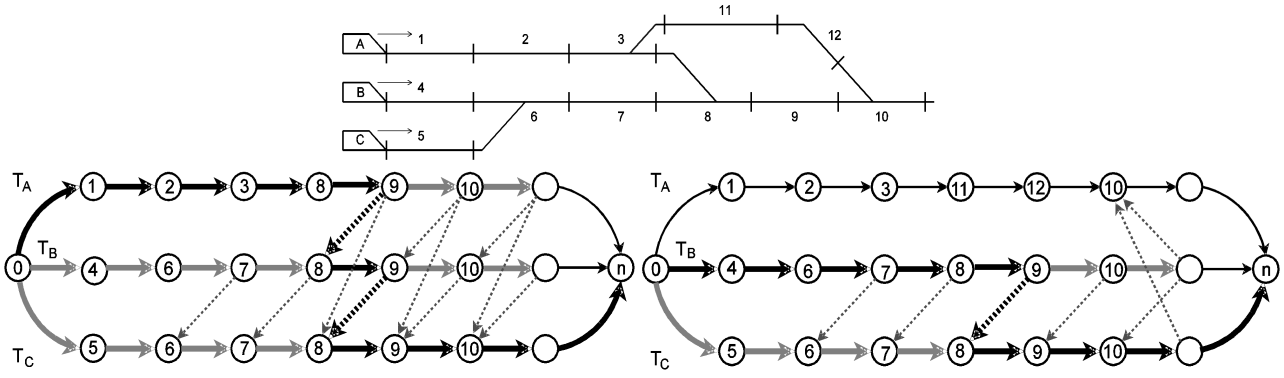

Figure 6: A small example and alternative graphs with different train paths

In the graph on the left side of Figure 6, the trains are sequenced in the order T_A, T_B, T_C. The right side graph is obtained changing the route of T_A. In this graph, the train order is the same for all trains, but T_A has no longer the conflicts with T_B and T_C on block sections 8 and 9. Consequently, the two graphs have different ramified critical paths. In particular, the whole route of train T_A does not belong anymore to the ramified critical path. For this reason, this may be possible that a longer route for train T_A yields a smaller value for the maximum consecutive delay.

The following property motivates the search for better routes among the routes of the ramified critical path. We denote by P^F the set of arcs of F belonging to the ramified critical path $\mathcal{R}(F, S)$.

Property 4.2 *Given two sets F and F', if $P^F \subseteq F'$, then $l^{F',S(F)}(0, n) \geq l^{F,S(F)}(0, n)$.*

Proof. Simply observe that the new solution $G(F', S(F))$ contains the path $\mathcal{C}(F, S(F))$. □

We exploit the two above properties to design a local search algorithm for routing optimization. The neighborhood of a solution $G(F, S)$ contains all the solutions $G(F', S(F'))$ in which F' is obtained from F changing a single route on the ramified critical path. If the route is contained on the critical path, this can be replaced only with a shorter route, as stated by Proposition 4.1. Notice that computing $G(F', S(F'))$ requires the execution of a rescheduling algorithm, which can be computationally expensive. We therefore limit the search for better solutions to a subset of promising neighbors only, as follows.

Given a rerouting option δ, let u and v be its first and last nodes, and $l(\delta)$ be the route length in $G(F, S)$. We estimate the potential of a new local rerouting option δ as the quantity $\Pi(\delta) = l^{F,S}(0, v) - l^{F,S}(0, u) - l(\delta)$, since the length $l^{F,S}(0, v)$ might decrease to $l^{F,S}(0, u) + l(\delta)$ when changing route. We restrict the neighborhood to the ψ routes with highest potential, where ψ is a parameter of our local search procedure. If none of the ψ routes improves the solution, we evaluate the next ψ routes with highest potential. The procedure continues until an improvement is found or no rerouting option is available or the time limit is reached.

For each of the ψ best routes we define a neighbor F' replacing an old route of F from u to v with a new route, and then execute the train rescheduling procedure. Among the ψ new solutions $G(F', S(F'))$ we choose the one having the shortest critical path, i.e., the one minimizing the maximum consecutive delay. In case of tie, we choose the solution with minimum average consecutive delay. In Figure 7, the pseudo-code of the routing optimization procedure is given.

5 Computational tests

We report on our computational experiments on a large sample of practical size instances. The experiments are based on the dispatching area of Utrecht Den Bosch, a bottleneck area of the Dutch railway network. We study the network simulating different traffic conditions for different kinds of disturbances, i.e., for different kinds of entrance delays and blocked tracks. A new timetable containing feasible arrival and departure times is computed after each disruption, with the objective of minimizing the maximum consecutive delay. Routing and scheduling algorithms are implemented in C++ language and executed on a laptop equipped with a 1.6 GHz Pentium M processor. Computational times and delays are always expressed in seconds. Each run of the compound routing/scheduling procedure is terminated after 180 seconds of computation. This choice makes the code compatible with real-time rail operations.

5.1 Test structure

The dispatching area under study is shown in Figure 8. This includes the Den Bosch station and the line connecting Utrecht (Ut) to Den Bosch (Ht), which is around 50 km long. The network is composed of 191 block sections and includes 21 platforms. There are two main tracks, divided into one long corridor for each traffic direction, a dedicated stop for freight trains (Ozbm) and seven passenger stations: Utrecht Lunetten (Utl), Houten (Htn), Houten Castellum (Hc), Culemborg (Cl), Geldermalsen (Gdm), Zaltbommel (Zbm)

Procedure `RoutingOptimization`
Input a solution $G(F, S)$,
Output best solution found,
While (rerouting options available) & (time limit not reached) & (max consecutive delay > 0) **do**
Build neighborhood for $G(F, S)$ according to properties 4.1 and 4.2, and insert all δ_i in list L_1,
local best $= +\infty$,
While $L_1 \neq \emptyset$ **do**
Add to list L_2 the ψ routing options with the highest potential $\Pi(\delta_i)$,
$L_1 = L_1 \setminus L_2$,
For all rerouting options in L_2
Create the alternative graph $\mathcal{G} = (N, F', A')$,
Reschedule trains for graph $\mathcal{G}$ according to a chosen algorithm ($B\&B$ or ARI),
If (*local best* $>$ current rerouting option in L_2)
Then *local best* $=$ current rerouting option in L_2,
Endfor
If *local best* $<$ *current solution*
Then *current solution* $=$ *local best* , $L_1 = \emptyset$,
Else $L_2 = \emptyset$
Endwhile
Endwhile
Return *current solution*.

Figure 7: Pseudo-code of the routing optimization procedure

and Den Bosch (Ht). Each traffic direction has nine entrances: Utrecht (Ut), Dordrecht (Ddr), Nijmegen (Nm), Beutuweroute, Geldermalsen Yard, Oss (Oss), Eindhoven (Ehv), Den Bosch Yard and Tilburg (Tl). There are several potential conflict points along each corridor due to possible different train speeds and four critical crossings: Geldermalsen station (block sections: 104, 105, 109, 113, 114 and 117), Dordrecht corridor (block sections: 101 and 102), Beutuweroute corridor (block section 4) and Den Bosch station (block sections: 142, 143, 146, 150, 151, 152, 154, 156, 157, 160, 161, 166, 167, 170, 171, 172, 180, 181, 182 and 183). Two extensions of the network, which are still under construction, are included (block sections: 96, 98, 128, 129 and from 131 to 140).

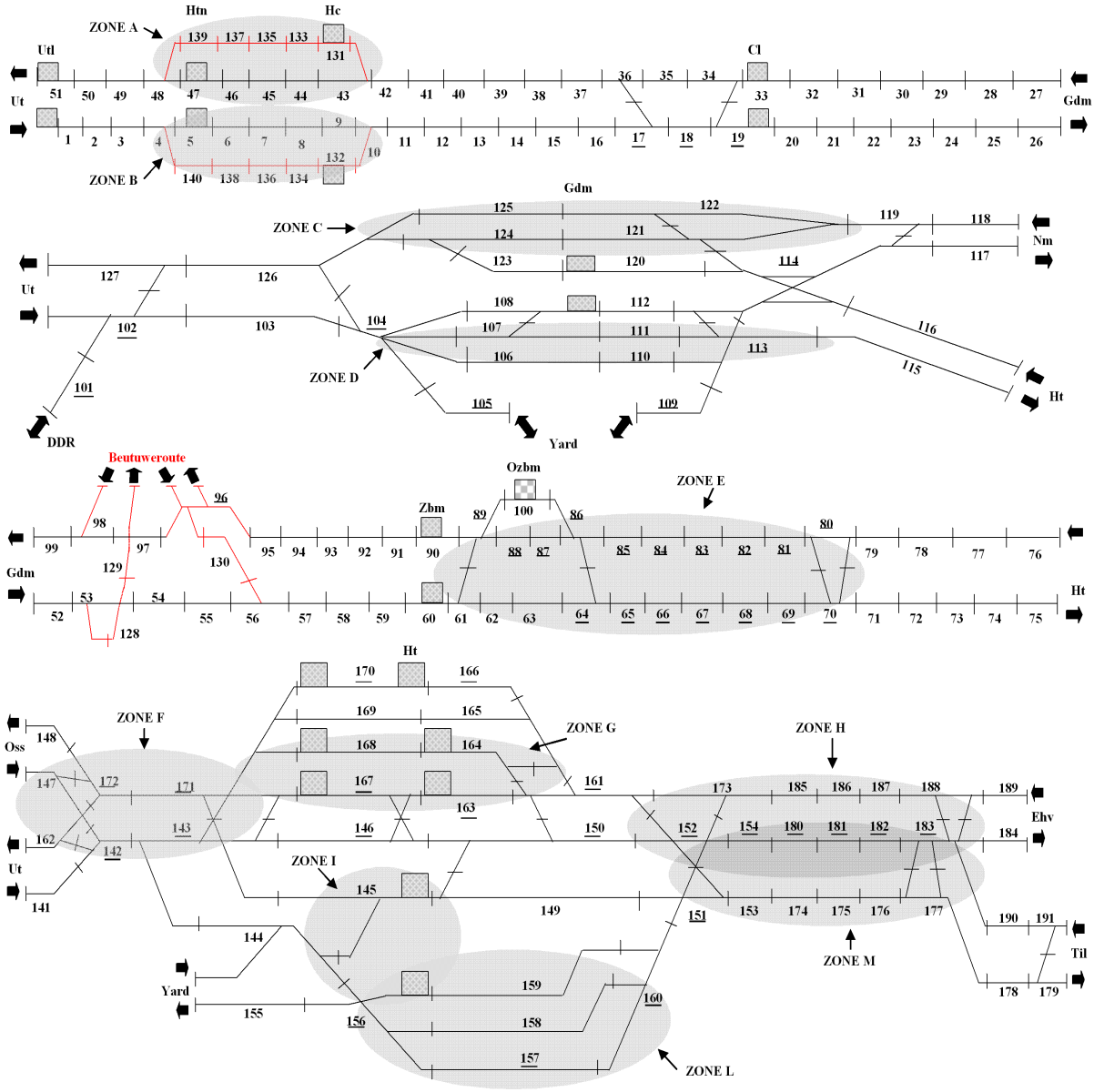

Figure 8: Utrecht Den Bosch rail network and the available rerouting zones

We consider a provisional timetable for 2007, which is hourly, cyclical and extended on the entire railway area. During a peak hour, in the dispatching area around Geldermalsen, 26 passenger and freight trains are scheduled in both directions. At Den Bosch station

the number of trains per hour increases up to 40. With this timetable, a scheduling instance for one hour of traffic contains an average of 3600 pairs of alternative arcs, the exact value depending on the route chosen for each train. In fact, the infrastructure offers several possibilities of train reordering and local rerouting. For each train, a default route and a set of local rerouting options are given. For example, a train is provisionally assigned to a default platform, but it is allowed to stop at different nearby platforms. Considering all possible alternative rerouting options yields to a set of 356 routes. Figure 8 shows all possible rerouting zones in grey color, labeled from A to M.

We also include constraints on the minimum transfer time between connected train services. Rolling stock connections are located in Zaltbommel and Den Bosch stations. Passenger connections are modeled in Den Bosch station for the traffic directions from Ossen to Utrecht and vice versa. The minimum time for passenger connections varies from two to five minutes, depending on the distance between the arrival platforms.

We test the *CDR* system ROMA under severe real-time traffic disruptions. We consider 13 configurations of randomly generated blocked tracks. Table 1 describes the test cases considered to test ROMA under strong disorder. Each disruption is obtained making unavailable a set of block sections, as reported in column 2. Each row presents average results on 24 configurations of entrance delay, with a maximum entrance delay varying from 1000 up to 1800 seconds and an average entrance delay of around 320 seconds. The values of the entrance delay are randomly chosen in a time window of typical train delays. In order to evaluate the effects of each perturbation on the remaining part of the schedule, the delayed trains are chosen among those entering the network in the first 30 minutes of the timetable.

In total, there are 312 instances with passenger and rolling stock connection constraints plus 312 instances with rolling stock connection constraints and without passenger connection constraints.

Table 1: Description of the test cases

	Unavailable Block Sections	%Unavailable Routes	%Changed Routes
Perturbation		0.0	0.0
Disruption 1	83	40.4	12.8
Disruption 2	131 132	30.9	5.1
Disruption 3	163 167	20.2	10.3
Disruption 4	122 107	30.3	17.9
Disruption 5	67 145 167	65.2	38.8
Disruption 6	67 175 186	67.4	41.0
Disruption 7	168 164 67 175	66.6	25.6
Disruption 8	5 186 163 66	69.1	43.6
Disruption 9	135 136 122 110 67 159 186 167	82.3	41.0
Disruption 10	158 66 110 186 167 163	69.4	41.0
Disruption 11	67	40.4	15.4
Disruption 12	125 106	30.3	5.1

The set of rerouting options used by the `disruption recovery` module is larger than that of the `routing optimization` module. In fact, for some blocked tracks, the only feasible rerouting option may force a train to miss a scheduled stop. This option is allowed for disruption recovery and forbidden for the routing optimization. In such cases, the delay at that stop is not considered in the experiments. Due to this fact, it may happen that the disrupted schedule exhibits a smaller delay than the undisrupted schedule.

The last two columns in Table 1 show the percentage of routes disrupted due to the presence of blocked zones and the percentage of changed routes by the `disruption`

`recovery` module to restore a feasible routing with respect to the original timetable. The percentage of unavailable routes is obtained by checking how many of the 356 original routes pass through an unavailable block section. In our tests this number varies up to 82% of the total number available, as shown in column 3. Similarly, the percentage of changed routes is obtained as the number of default routings (one for each train) passing through an unavailable block section, divided by the total number of trains. On average, a moderate amount of rerouting actions, about 22%, is sufficient to handle the disturbances in the reference test cases.

All the test cases are evaluated using the local search rerouting procedure of Section 4.2 and the two scheduling algorithms of Section 4.1.

5.2 System configuration

The real-time purpose of ROMA imposes strict time limits to produce a new feasible timetable, which is attained limiting the execution of the scheduling algorithm and the number of rerouting possibilities. Specifically, reducing the time limit of the scheduling algorithm allows evaluating a larger number of rerouting possibilities, at the price of less accurate schedules. In Table 2, we present a comparison between two configurations, which let different importance to rescheduling and rerouting strategies. In "Config. 1", we let $\psi = \infty$ and allow only 10 seconds of maximum computation time for $B\&B$. In "Config. 2", we let $\psi = 5$ and allow 30 seconds to $B\&B$. The main difference between the two configurations is that "Config. 2" relies more on the scheduling procedure, since this allows larger computation times to the scheduling algorithm and reduces the number of rerouting options explored by the local search algorithm ($\psi = 5$). Each row in Table 2 describes average results on the 624 instances of Section 5.1. Specifically, "Default Routing" reports on the solutions provided by the `disruption recovery` module, whereas "Routing Optimization" reports on the solutions given by the iterative rerouting and rescheduling procedure. Columns 2–4 and 5–9 show the results with the default routing and the optimal routing, respectively. All the values are expressed in seconds. Columns 2 and 5 show the maximum consecutive delays. Columns 3 and 6 show the average consecutive delays computed at the borders of the dispatching area. Columns 4 and 7 indicate the average computation time of the two procedures over the 624 instances. The percentage of changed routes (column 8) consists of counting how many trains have been rerouted in the overall process with respect to the original timetable. Being a simple counter, this indicator may be very sensitive to the number of trains running in the timetable. Finally, column 9 ("Time Limits") shows the number of times the compound problem is terminated after 180 seconds of computation.

Table 2: Results for varying ψ and the time allowed to the $B\&B$

Average Results	Default Routing			Routing Optimization				
	Delay Max	Delay Avg	Time Tot	Delay Max	Delay Avg	Time Tot	%Changed Routes	Time Limits
Config. 1	279.8	50.4	2.1	245.3	44.8	33.9	23.8	76
Config. 2	279.1	50.4	5.2	251.6	45.3	33.2	23.6	64

As expected, the scheduling solutions obtained for the default routing are slightly better for "Config. 2", but the time needed to produce the first schedule is more than double with respect to "Config. 1". However, "Config. 1" gives better results after the routing

optimization procedure, with similar computation time limits. For this reason, in the rest of our computational tests "Config. 1" will be set as the CDR system configuration.

5.3 Scheduling and routing strategies

In this section, we compare the performance of the $B\&B$ and ARI scheduling algorithms on the 624 instances of Section 5.2, without and with routing optimization.

Each row of Table 3 reports the average performance of the two scheduling algorithms on the 624 instances. In general, $B\&B$ provides much better solutions in terms of maximum and average consecutive delays with respect to ARI but requires more computation time. In fact, ARI always finds a solution in less than 10 seconds, whereas $B\&B$ requires on average more than 30 seconds. On the other hand, the improvement of the maximum consecutive delay when using $B\&B$ over ARI is about 40%. The average delay for the two scheduling algorithms exhibits a similar behavior but the improvement when passing from ARI to $B\&B$ is about 25%. When comparing the percentage of rerouted trains similar values are obtained (around 23% of the circulating trains).

Table 3: $B\&B$ versus ARI

Average Results	Default Routing			Routing Optimization			
	Delay Max	Delay Avg	Time Tot	Delay Max	Delay Avg	Time Tot	%Changed Routes
ARI	489.4	66.9	0.6	417.0	60.5	8.1	23.4
B&B	279.8	50.4	2.1	245.3	44.8	33.9	23.8

Figure 9 shows the reduction of the maximum consecutive delay from the worst configuration, obtained with ARI and the default routing, up to the best configuration, obtained with $B\&B$ and routing optimization. The overall reduction is about 50%, about 43% of which due to the scheduling algorithm. The routing optimization procedure contributes with an additional reduction of about 7%. The overall improvement is therefore largely due to the use of an advanced scheduling algorithm.

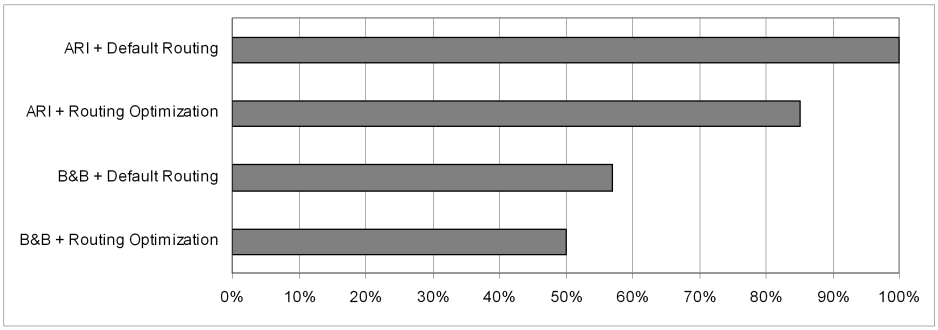

Figure 9: Maximum consecutive delay with four configurations of ROMA

Table 4 shows the average consecutive delays at each station and at the borders of the dispatching area ("Out"). Each row shows the results obtained with the different scheduling/routing strategies shown in the first two columns. When comparing the two scheduling algorithms using the default routing, although the objective function minimizes the maximum consecutive delay, $B\&B$ is also able to produce better solutions in terms of average consecutive delays at all stations. When comparing the solutions using the local

rerouting optimization strategy the average delay is more effectively reduced. In fact, the routing optimization and *ARI* [resp. *B&B*] decrease the average delay in four [resp. five] stations.

Table 4: Average consecutive delays at stations

Scheduling	Routing	Out	Htn	Cl	Gdm	Zbm	Hc	Ht
ARI	Default	67.0	17.1	21.0	24.4	94.1	20.7	97.5
ARI	Optimization	59.8	14.0	19.9	26.3	88.7	23.1	84.7
B&B	Default	50.4	16.9	20.8	6.0	78.6	9.6	66.9
B&B	Optimization	44.8	15.1	19.4	6.6	73.7	8.7	56.5

5.4 Passenger connections

Table 5 evaluates the effects of adding/removing passenger connection constraints. In this section we compare the different cases in which all connections are maintained or no connection is enforced. This approach differs from that of other works on delay management (see, e.g., Schöbel [29]), in which one aims to decide which connections have to be maintained or cancelled in order to minimize the inconvenience for the passengers. Each row of table 5 shows the average results for the 312 instances of Section 5.1 without ("Off") or with ("On") passenger connections and using *ARI* or *B&B*. When passenger connections are in use the maximum and average consecutive delays increase of about 10%. This is therefore the cost of taking into account such constraints in presence of disturbances. The computation times required to solve the problem, with or without passenger connections constraints, do not vary significantly. Moreover, the use of *B&B* with passenger connections gives better results even when compared to *ARI* without passenger connections.

Table 5: Effects of passenger connections

Average Results	Passenger Connections	Default Routing			Routing Optimization		
		Delay Max	Delay Avg	Time Tot	Delay Max	Delay Avg	Time Tot
ARI	Off	456.9	62.6	0.6	391.0	57.1	11.0
B&B	Off	266.2	48.5	2.2	234.7	43.2	34.4
ARI	On	521.9	71.3	0.6	427.6	62.4	11.2
B&B	On	293.5	52.2	2.1	255.9	46.3	33.3

Table 6 shows the effects of passenger connections at all stations. We report on the average consecutive delay measured at each station and at the borders of the dispatching area ("Out"). All these solutions are computed using *B&B* and the routing optimization procedure. Each row refers to the average results on the same instances of Table 5. In our tests the passenger connections constraints are active only in Zaltbommel (Zbm) and Den Bosch (Ht) stations, but delays may propagate among the trains. When the passenger connections are "On" the average consecutive delay increases mainly in the stations where the passenger connections are active. In fact, on average, delays increase more than 5 seconds at Den Bosch and almost 4 seconds at Zaltbommel, whereas this increase is less evident at the other stations.

Table 6: Influence of passenger connections at stations

Passenger Connections	Out	Htn	Cl	Gdm	Zbm	Hc	Ht
Off	43.2	14.6	19.1	7.2	71.8	8.6	53.8
On	46.3	15.5	19.6	5.9	75.6	8.7	59.2

5.5 Timetable disruptions

We next show the ability of ROMA to attain feasible solutions when a large part of the network is unavailable. In Table 7, we study the influence of strong timetable disruptions. Each row refers to average results over 48 instances, which correspond to the 24 perturbations of Section 5.1 with and without passenger connection constraints. Here, we report results on each disruption scheme of Table 1.

Table 7: Effects of timetable disruptions

Average Results	Default Routing			Routing Optimization			
	Delay Max	Delay Avg	Time Tot	Delay Max	Delay Avg	Time Tot	%Changed Routes
ARI							
Perturbation	364.5	45.0	0.6	267.0	35.8	22.4	2.9
Disruption 1	495.8	66.8	0.6	398.2	57.7	15.7	15.6
Disruption 2	338.5	40.4	0.6	254.6	33.2	8.8	7.2
Disruption 3	378.6	49.7	0.5	273.7	39.7	17.0	12.6
Disruption 4	420.3	49.3	0.5	339.3	43.7	9.2	18.4
Disruption 5	572.4	88.8	0.6	517.0	81.1	9.5	31.4
Disruption 6	570.1	84.8	0.8	486.5	78.8	11.2	41.1
Disruption 7	569.1	75.3	0.7	484.6	69.2	9.2	26.8
Disruption 8	566.1	81.7	0.7	489.4	75.3	6.9	44.1
Disruption 9	590.9	86.8	0.7	552.2	83.2	2.0	41.5
Disruption 10	607.2	88.4	0.7	513.3	78.7	6.9	41.5
Disruption 11	524.6	69.0	0.6	464.8	62.2	15.0	16.6
Disruption 12	365.2	45.1	0.5	280.8	38.2	11.2	6.8
B&B							
Perturbation	183.8	20.8	0.5	127.0	15.2	17.7	3.3
Disruption 1	294.9	49.7	0.8	271.3	46.0	36.4	15.1
Disruption 2	173.6	18.7	0.5	124.5	13.7	8.4	7.8
Disruption 3	196.0	24.4	0.4	148.0	18.6	18.4	12.3
Disruption 4	184.0	21.3	0.4	125.0	15.2	9.9	19.2
Disruption 5	376.9	77.7	4.1	353.5	73.4	61.2	31.5
Disruption 6	345.4	79.1	3.9	321.7	73.0	55.9	41.3
Disruption 7	352.6	67.8	4.5	320.5	61.1	69.3	26.7
Disruption 8	331.7	69.2	3.4	312.8	62.4	29.9	44.3
Disruption 9	337.4	71.9	3.2	323.6	67.7	11.5	41.7
Disruption 10	351.0	78.7	4.2	326.2	71.0	44.7	41.8
Disruption 11	327.5	55.5	2.3	308.0	50.0	68.7	17.1
Disruption 12	183.9	20.9	0.4	127.5	15.4	8.9	7.2

When large part of the network is unavailable the maximum consecutive delay increases. In this case, due to the larger number of trains running on the same block sections, the network results increasingly congested. When comparing the situation with and without blocked tracks (rows "Perturbation"), the delay may even be double (Disruption 5 with $B\&B$) and the percentage of rerouted trains ranges from 3% to more than 40%. Moreover, for some disruptions (5, 7 and 11) the average computation time of the $B\&B$ based CDR system exceeds one minute of CPU time.

5.6 Computation time versus solution quality

We next show the performance of ROMA when varying the time limits allowed for finding a solution. In Figure 10, we plot the maximum and average consecutive delays of the best solutions found under different time limits for the 48 perturbations and the 144 disruptions classified as 5, 7 and 11 in Table 1. The former case allows the maximum routing flexibility, being available all the 356 original routes, while the three latter cases are those requiring on average the longest computation time (see Table 7). The consecutive delays are depicted after each half minute of computation, up to 5 minutes. For each case, we show the average results over 48 instances corresponding to the 24 perturbations of Section 5.1 with and without passenger connection constraints.

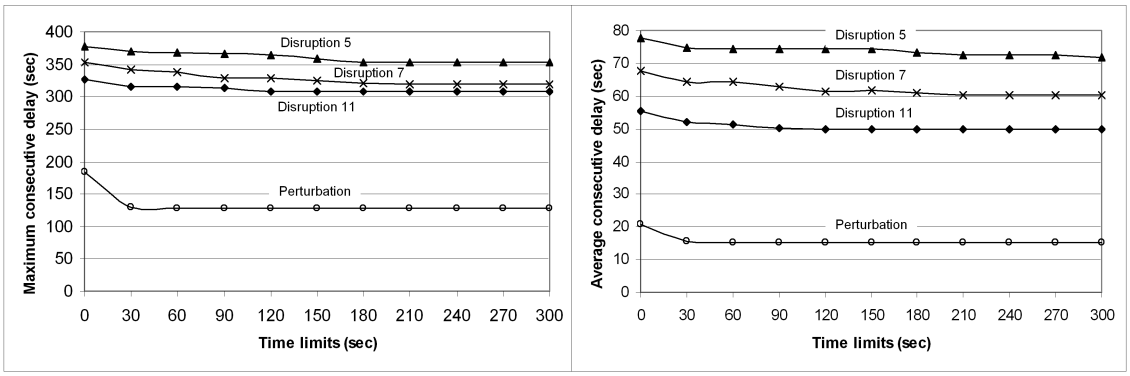

Figure 10: Maximum and average consecutive delays at different time limits

At time 0, the consecutive delays when using the routes prescribed by the `disruption recovery` module are shown. We observe that for real-time purposes the algorithm is quite effective, especially in the perturbation case, when limited rerouting allows to obtain a good solution already after 30 seconds. As far as the disruptions 5, 7 and 11 are concerned, a more extensive search among the available rerouting options is needed in order to find a local minimum. However, in all cases, increasing the computation time over 180 seconds does not improve the solution quality significantly.

6 Conclusions

Our work shows the effectiveness of ROMA to reduce the propagation of real-time timetable disturbances. ROMA is able to optimize railway traffic also when the timetable is not conflict-free. This fact enables its usage when managing railway traffic in case of severe traffic disturbances, such as when emergency timetables are required and traffic dispatchers need support to solve conflicts.

While the scheduling algorithm is able to solve large instances within a short computation time, the routing optimization procedure does not exploit all the potential offered by routing flexibility. A limited attempt to investigate larger neighborhoods, in which several routes are changed simultaneously, shows that there are further margins for improving upon the local minima found by our local search algorithm. Further research should therefore be addressed to the analysis of larger neighborhoods within short computation time as well as to the development of more sophisticated rerouting metaheuristics.

More general research lines should address: (*i*) the problem of proposing several alternative solutions to the dispatcher, e.g. non-dominated solutions with respect to the minimization of maximum and average secondary delays, and (*ii*) the problem of considering train dynamics when choosing different train orders at conflict points, since in the current system train dynamics are computed separately from the scheduling problem in a preprocessing step.

Acknowledgments

We thank ProRail (The Netherlands) for providing the instances and Prof. Ingo Hansen for his helpful comments and suggestions. This work is partially supported by the programs "Towards Reliable Mobility" of the Transport Research Centre Delft, the Dutch foundation "Next Generation Infrastructures" and ProRail.

References

[1] B. Adenso-Díaz, M. Oliva González, P. González-Torre, On-line timetable rescheduling in regional train services, *Transportation Research Part B* **33** 387–398 (1999) .

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

[3] N. Berends, N. Ouburg, Beschrijving ARI-functionaliteit, ProRail Internal Specification (in Dutch) (2005).

[4] D. Bertsimas, S. Stock Patterson, The Traffic Flow Management Rerouting Problem in Air Traffic Control: A Dynamic Network Flow Approach, *Transportation Science* **34** (3) 239–255 (2000).

[5] L. Bianco, P. Dell'Olmo, S. Giordani, Scheduling models for air traffic control in terminal areas, *Journal of Scheduling* **9** 223–253 (2006).

[6] G. Caimi, D. Burkolter, T. Herrmann, Finding delay-tolerant train routings through stations. In H. Fleuren (Ed.), Operations Research Proceedings 2004: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), pp. 136-143, Springer Verlag (2005).

[7] M. Carey, S. Carville, Scheduling and platforming trains at busy complex stations, *Transportation Research Part A* **37** (3) 195–224 (2003).

[8] J.F. Cordeau, P. Toth, D. Vigo, A Survey of Optimization Models for Train Routing and Scheduling, *Transportation Science* **32** (4) 380–420 (1998).

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

[10] X. Delorme, J. Rodriguez, X. Gandibleux, Heuristics for railway infrastructure saturation, *Electronic Notes in Theoretical Computer Science*, **50** (1) 39-53 (2001).

[11] X. Delorme, Modelisation et resolution de problemes lies a lexploitation dinfrastructures ferroviaires. PhD thesis, University of Valenciennes et du Hainaut Cambresis (2003).

[12] M.M. Dessouky, Q. Lu, J. Zhao, R.C. Leachman, An exact solution procedure to determine the optimal dispatching times for complex rail networks, *IIE Transaction* **38** (2) 141–152 (2006).

[13] M.J. Dorfman, J. Medanic, Scheduling trains on a railway network using a discrete event model of railway traffic, *Transportation Research Part B* **38** 81–98 (2004).

[14] P. Fioole, L.G. Kroon, G. Maroti, A. Schrijver, A rolling stock circulation model for combining and splitting of passenger trains, *European Journal of Operational Research* **174** (2) 1281–1297 (2006).

[15] X. Gandibleux, J. Jorge, S.Angibaud, X. Delorme, J. Rodriguez, An ant colony optimization inspired algorithm for the set packing problem with application to railway infrastructure. In Proceedings of the Sixth Metaheuristics International Conference (MIC2005), pp. 390-396 (2005).

[16] A. Higgins, E. Kozan, L. Ferreira, Optimal scheduling of trains on a single line track, *Transportation Research Part B* **30** 147–161 (1996).

[17] A. Higgins, E. Kozan, Heuristic techniques for single line train scheduling, *Journal of Heuristics* **3** 43–62 (1997).

[18] J.S. Hooghiemstra, L.G. Kroon, M.A. Odijk, M. Salomon, P.J. Zwaneveld, Decision Support Systems Support the Search for Win-Win Solutions in Railway Network Design, *Interfaces* , **29** (2) 15-32 (1999).

[19] L.G. Kroon, H. Romeijn, P.J. Zwaneweld, Routing trains through railway networks: complexity issues, *European Journal of Operational Research* **98** 485–498 (1997).

[20] R. Lusby, J. Larsen, D. Ryan, M. Ehrgott, Routing Trains Through Railway Junctions: A New Set Packing Approach. Technical report 2006-21. Informatics and Mathematical Modelling, Technical University of Denmark (2006).

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

[22] A. Mascis, D. Pacciarelli, M. Pranzo, Scheduling Models for Short-term Railway Traffic optimization. In *M. Hickman, P. Mirchandani and S. Voß(Eds.), Lecture Notes in Economics and Mathematical Systems 600: Computer-aided Systems in Public Transport. Springer.* 71–90 (2008).

[23] M. Mazzarello, E. Ottaviani, A Traffic Management System for Real-Time Traffic optimization in Railways, *Transportation Research Part B* **41** (2) 246–274 (2007).

[24] L. Nie, I.A. Hansen, System analysis of train operations and track occupancy at railway stations, *European Journal of Transport and Infrastructure Research* **5** (1) 31–54 (2005).

[25] E. Oliveira, B.M. Smith, A Job-Shop Scheduling Model for the Single-Track Railway Scheduling Problem, School of Computing Research Report 2000.21, University of Leeds, England (2000).

[26] J. Pachl, Railway Operation and Control. VTD Rail Publishing, Mountlake Terrace, USA (2002).

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

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

[29] A. Schöbel, A model for the delay management problem based on mixed-integer-programming, In C. Zaroliagis (ed.), *Electronic Notes in Theoretical Computer Science* **50** (2001).

[30] B. Szpigel, Optimal Train Scheduling on a Single Track Railway. In Operational Research '72, M. Ross (ed.), pp. 343–352, Amsterdam, The Netherlands (1973).

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

[32] P.J. Zwaneveld, L.G. Kroon, S.P.M. van Hoesel, Routing trains through a railway station based on a node packing model, *European Journal of Operational Research* **128** 14–33 (2001).