A Bilevel Rescheduling Framework for Optimal Inter-Area Train Coordination *

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

1 **Centre for Industrial Management, Katholieke Universiteit Leuven Celestijnenlaan 300A, 3001 Heverlee, Belgium**
`francesco.corman@cib.kuleuven.be`

2 **Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre via della Vasca Navale 79, 00146 Roma, Italy**
`a.dariano@dia.uniroma3.it, pacciarelli@dia.uniroma3.it`

3 **Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Siena via Roma 56, 53100 Siena, Italy**
`pranzo@dii.unisi.it`

Abstract

Railway dispatchers reschedule trains in real-time in order to limit the propagation of disturbances and to regulate traffic in their respective dispatching areas by minimizing the deviation from the off-line timetable. However, the decisions taken in one area may influence the quality and even the feasibility of train schedules in the other areas. Regional control centers coordinate the dispatchers' work for multiple areas in order to regulate traffic at the global level and to avoid situations of global infeasibility. Differently from the dispatcher problem, the coordination activity of regional control centers is still underinvestigated, even if this activity is a key factor for effective traffic management.

This paper studies the problem of coordinating several dispatchers with the objective of driving their behavior towards globally optimal solutions. With our model, a coordinator may impose constraints at the border of each dispatching area. Each dispatcher must then schedule trains in its area by producing a locally feasible solution compliant with the border constraints imposed by the coordinator. The problem faced by the coordinator is therefore a bilevel programming problem in which the variables controlled by the coordinator are the border constraints. We demonstrate that the coordinator problem can be solved to optimality with a branch and bound procedure. The coordination algorithm has been tested on a large real railway network in the Netherlands with busy traffic conditions. Our experimental results show that a proven optimal solution is frequently found for various network divisions within computation times compatible with real-time operations.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems; G.1.6 Optimization; I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Train Delay Minimization, Schedule Coordination, Bilevel Programming

Digital Object Identifier 10.4230/OASIcs.ATMOS.2011.15

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

1 Introduction

This paper deals with a multi-area train scheduling problem faced by traffic controllers at regional railway control centers. Typically, the real-time traffic control at the national level is organized into a set of regional traffic control centers each coordinating several dispatchers. For instance, the Dutch network is subdivided into a national center in Utrecht, four regional centers (Amsterdam, Eindhoven, Rotterdam and Zwolle) and more than sixty dispatching areas.

The real-time traffic management of each regional area is hierarchically organized into two decision levels. At the lower level, dispatchers control local areas with knowledge of the traffic flow limited to their respective areas. When train operations are perturbed, each dispatcher regulates traffic by minimizing the deviation from the off-line scheduled timetable and by computing a *locally feasible schedule* in his/her dispatching area. However, the decisions taken locally may influence the quality and even the feasibility of the train schedules of other areas. At the higher level, the coordinator is responsible for the traffic management over a railway network of k areas with a global overview of the traffic flow and controls the rescheduling decisions taken by the k dispatchers (see Figure 1). The coordinator goals are to ensure the *global feasibility* of train schedules (i.e., the union of all locally feasible schedules must be feasible) and to pursue the overall quality of the local solutions at the regional level. To reach these goals, the coordinator may impose constraints to the local solutions provided by the dispatchers.

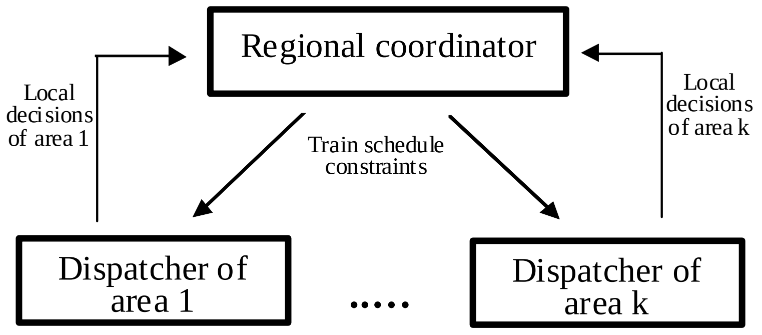
Regional coordinator
Local decisions of area 1
Local decisions of area k
Train schedule constraints
Dispatcher of area 1
.....
Dispatcher of area k

Figure 1 Interaction between coordinator and dispatchers.

Due to the complexity of the overall train rescheduling problem, decision support systems (DSSs) are needed to help dispatchers and coordinators to manage railway traffic under this two-level hierarchy. As far as the dispatcher problem is concerned, many DSSs are described in the literature, based on exact and heuristic solution procedures. Recent surveys on models and algorithms for the dispatcher problem can be found in Ahuja et al. (2005), D'Ariano (2010) and Lusby et al. (2011). Most of the approaches are based on a macroscopic view of the network, in which a line between two stations is aggregated into a single resource. However, the recent trend is to increase the level of detail in the optimization models in order to ensure that a feasible model solution can also be implemented in practice. In the recent literature on microscopic models, the train scheduling problem is formulated as a job shop scheduling problem with additional constraints (see e.g. D'Ariano et al., 2007 and Mannino and Mascis, 2009).

Differently from the dispatcher problem, the coordinator problem at the regional control centers has not received much attention in the literature on multi-area train scheduling, although poor coordination between areas may result in poor overall performance, with a risk of inter-area deadlocks. The few papers existing on the coordinator problem mainly focus on certifying the global feasibility of the local solutions or detecting global infeasibility and suggesting possible coordination actions for recovery (Mazzarello and Ottaviani (2007), Strotmann (2007) and Corman et al. (2011)). A number of important open problems remain for both academic researchers and practitioners, such as the optimization of coordinator performance and the definition of general methods to find globally feasible schedules when infeasibility is detected.

A stream of research on methodologies for railway traffic regulation and coordination of local areas started with the European project COMBINE 2 (Pacciarelli, 2003). Train movements in the local dispatching areas are modeled by an alternative graph formulation (Mascis and Pacciarelli, 2002), while a higher level of control considers aggregate information about the local solvers. The implementation of these methodologies are reported in Mazzarello and Ottaviani (2007) for two test cases of the Dutch railway network, and a practical pilot is also described for one of the two test cases.

In Strotmann (2007), a two-level approach for rescheduling trains between multiple areas is considered. At the lower level local solutions are computed in each area by greedy heuristic scheduling procedures while, at the higher level, a coordinator is used to check whether neighboring areas have consistent solutions. The coordination procedure imposes train ordering constraints at the borders between areas with an iterative approach until a feasible schedule to the global problem is found or the procedure fails in finding a globally feasible schedule.

The coordinator problem has been recently addressed by Corman et al. (2011) on a complex and busy Dutch railway network divided into two dispatching areas. A coordination framework is proposed to support distributed scheduling, that combines microscopic modeling of train movements at the local level with an aggregate view of the situation at the global level. An exact algorithm by D'Ariano et al. (2007) is used at local level to solve the dispatcher problem in each area, while heuristic procedures are proposed to solve the coordinator problem.

So far, to the best of our knowledge no paper addresses the problem of assessing the performance of the coordinator. This lack of research motivates the current paper. This work is based on the above-described framework and develops a new coordination procedure to compute optimal solutions to the coordinator problem or at least to assess the quality of the feasible solutions found.

With the coordination procedure developed in this paper, the coordinator exchanges information with each dispatcher. We formally define the *border* between two or more dispatching areas as a set of block sections, called *border block sections*, which are shared between neighboring areas. The order of the trains traversing a border block section must therefore be the same in the areas sharing it and in case of conflict between the dispatchers the coordinator may impose a common order or time windows of passing times for some trains that must be respected by all dispatchers. Each dispatcher computes a locally feasible detailed schedule satisfying a given set of constraints at the area border, such as a partial order of trains passing the border or a time window for the entry/exit event of each train into/out of the area. The local solution is computed by solving a train scheduling problem with minimization of train delays. An alternative graph formulation (Mascis and Pacciarelli, 2002) models the dispatcher problem. The blocking time theory is used to compute arc weights (see, e.g., Hansen and Pachl (2008)) so that train movements are modeled at a microscopic level of detail compliant with the safety system and the operating rules. The exact algorithm of D'Ariano et al. (2007) is then adopted to solve the alternative graph of each dispatching area.

After the computation of local solutions, each dispatcher sends back to the coordinator aggregate information on the solution found, including lower and upper bounds and a set of time lags between every pair of entry/exit events at the area border.

The coordinator builds a *border graph* whose nodes are the entry/exit events at each border block section plus two dummy nodes 0 and n that are needed to compute the objective function. Two properties are proved: *(i)* The first property allows to prove global feasibility of the union of locally feasible schedules; *(ii)* The second property allows to prove global optimality of the union of locally feasible schedules, for a given set of coordination constraints.

Properties *(i)* and *(ii)* of the coordinator problem enable the development of a branch and bound procedure through which the coordinator can guide the search towards a globally optimal solution. The idea is to define a list of alternative sets of coordination constraints whose union covers all coordination actions, each associated with a *coordinator graph* used to model implications of the constraints set and to compute a lower bound on the optimum. If for a set of constraints the lower bound is equal to or greater than the current upper bound, the set is removed from the list. Otherwise, a branch is performed by producing two new sets of constraints and adding them to the list. The procedure is guaranteed to converge to the global optimum if the solutions provided by the dispatchers at each step are locally optimal. Otherwise, an optimality gap is always associated with the current best global solution.

The coordination framework is tested on a large and busy region of the Dutch network spanning ten dispatching areas. Experimental results show that a near-optimal global solution is found within the tight time windows required for real-time traffic control. The branch and bound algorithm for the coordinator problem is also compared with the heuristic proposed by Corman et al. (2011) and with the centralized approach described in D'Ariano et al. (2007).

2 Mathematical formulation

Following the two-level hierarchy of Vicente and Calamai (1994), in our formulation the coordinator is the leader of a bilevel program and the dispatchers are the followers. This hierarchy is adopted also in railway practice, as described in Section 1. We next describe the models adopted for the problems faced by dispatchers and coordinators. Both problems are formulated with alternative graphs (Mascis and Pacciarelli, 2002).

The alternative graph is a triple $\mathcal{G} = (N, F, A)$, where $N = \{0, 1, \ldots, n\}$ is a set of nodes, F is a set of directed arcs (*fixed*) and A is a set of pairs of directed arcs (*alternative*). The nodes are associated with events, such as the start or completion of the schedule (nodes $0/n$) or the start of an operation (nodes $1, \ldots, n-1$). Each arc (i, j) is either fixed or alternative and has an associated weight w_{ij}. The set A contains pairs of *alternative arcs*, which model the sequencing decisions of the problem. If $((k, j), (h, i)) \in A$, arc (k, j) is the alternative to arc (h, i). We call t_i the start time associated with event i. A *selection* S is a set of alternative arcs, at most one arc from each alternative pair. A selection, in which exactly one arc is chosen from each pair in A, is a *feasible schedule* (or a *solution*) if the graph $(N, F \cup S)$ has no cycles with positive weight (Mascis and Pacciarelli, 2002).

Given a solution S, $l^S(i, j)$ denotes the weight of a longest path from i to j in $(N, F \cup S)$. A feasible timing t_i for operation i is then $t_i = l^S(0, i)$. Note that t_0 is a constant equal to 0. A feasible schedule is an *optimal solution* if $l^S(0, n)$ is minimum over all the feasible schedules.

The general alternative graph formulation can be viewed as the following disjunctive program:

$$\begin{array}{ll} \min t_n - t_0 & \\ s.t. & \\ t_j - t_i \geq w_{ij} & (i,j) \in F \\ (t_j - t_k \geq w_{kj}) \vee (t_i - t_h \geq w_{hi}) & ((k,j),(h,i)) \in A \end{array}$$

In the alternative graph formulation of the dispatcher problem (*dispatcher graph*), each operation represents the event that a train enters a block section or a platform. Variable t_i, for $i = 1, \ldots, n-1$, is used to model the start time of operation i, i.e., the entrance time of the train to the associated block section or platform. A train route corresponds to a *job*, i.e., a sequence of operations. Fixed constraints in F must be satisfied by any feasible timing for each train on its specific route. For each operation i, let $\sigma(i)$ be the operation which follows i on the route of the associated train. In a solution, the precedence relation $t_{\sigma(i)} \geq t_i + w_{i\sigma(i)}$ must hold, where $w_{i\sigma(i)} > 0$ is the minimum running time for operation i. Fixed constraints are also used to model *time windows* of <earliest, latest> entrance times for the trains running in the dispatching area. The earliest entrance time (release) is represented by a fixed arc from node 0 to the first node of the corresponding job, while the latest entrance time (deadline) is a fixed arc from the first node of the job to node 0 with negative weight. Additional fixed constraints can also model train delays and other railway constraints, as shown in D'Ariano et al. (2007), D'Ariano (2010), D'Ariano et al. (2008), Corman et al. (2009) and Corman et al. (2010).

Alternative pairs in A model the train sequencing decisions. For each pair i and j of operations associated with the entrance of two trains to the same block section, we define $k = \sigma(i)$, $h = \sigma(j)$ and introduce the disjunction $(t_j - t_{\sigma(i)} \geq w_{\sigma(i)j}) \vee (t_i - t_{\sigma(j)} \geq w_{\sigma(j)i})$, where $w_{\sigma(i)j} > 0$ and $w_{\sigma(j)i} > 0$ are minimum setup times. With this constraint, the follower train can enter the block section only after that the feeder train enters the next block section plus the setup time. The choice of one of the two arcs corresponds to choosing the train sequence on the associated block section.

A train schedule in the dispatcher graph specifies a value for the start time of each operation. The schedule is feasible (deadlock-free and conflict-free) if it satisfies all constraints belonging to the set F and exactly one constraint for each alternative pair belonging to the set A. The objective function is the minimization of the maximum consecutive delay of all trains at a set of relevant points, i.e., the scheduled stops and the exit points of the dispatching area. This objective function corresponds to the quantity $t_n - t_0$ by associating suitable weights with the arcs ending at node n, as in D'Ariano et al. (2008).

2.1 Global feasibility

Let us consider a region divided into k dispatching areas, let $\mathcal{G}^x = (N^x, F^x, A^x)$ be the alternative graph associated to dispatching area $x = 1, \ldots, k$, let S^x be a locally feasible schedule for area x and let $S = \bigcup_{x=1}^{k} S^x$ be the union of the k selections.

In principle, the feasibility of the global solution S can be checked by building a global alternative graph $\mathcal{G} = (\cup_x N^x, \cup_x F^x, \cup_x A^x)$ of the whole region and then selecting the arcs for each dispatching area according to the dispatcher decisions. If there are no positive weight cycles in $\mathcal{G}(\mathcal{S})$ the local solutions are globally feasible. However, a drawback of $\mathcal{G}$ is its size that increases linearly with the number of block sections and quadratically with the number of trains in the network.

In order to reduce the amount of data managed by the coordinator, let us define a set of *border nodes* N_B composed of the dummy nodes 0 and n and of all the nodes associated with the entrance of a train into a border block section and to the entrance in the subsequent block section (which corresponds to the exit of the train from a border block section), for all the k dispatching areas. Given a locally feasible selection S^x, its *graph compression* of $\mathcal{G}^x(S^x)$ is obtained by contracting all the nodes in $N^x \setminus N_B$ and then deleting all the redundant arcs. By construction, there is an arc (i, j) in the graph compression if and only if there is a directed path from i to j in $\mathcal{G}^x(S^x)$, and (i, j) is weighted with $l^{S^x}(i, j)$.

A *border graph* $BG(S)$ is defined as follows. The set of nodes is composed of the set of border nodes N_B. The set of arcs is obtained by the graph compression of $\mathcal{G}^x(S^x)$ for each dispatching area $x = 1, \ldots, k$, i.e., there is an arc (i, j) with weight w_{ij} in $BG(S)$ if the weight of the longest path from i to j in $\mathcal{G}^x(S^x)$ is $w_{ij} < \infty$, for some $x = 1, \ldots, k$. Clearly, redundant arcs in $BG(S)$ can be deleted. The following property holds.

▶ **Theorem 2.1** (Feasibility property)**.** Consider a global area composed of k local areas. Given a locally feasible schedule S^x for each dispatching area, $S = \bigcup_{x=1}^{k} S^x$ is a globally feasible selection if and only if the border graph $BG(S)$ has no positive weight cycles.

▶ **Corollary 2.2** (Global objective function)**.** Consider a global area composed of k local areas and a locally feasible schedule S^x for each dispatching area $x = 1, \ldots, k$. If the associated border graph contains no positive weight cycles, the weight of the longest path from 0 to n in the border graph is the maximum consecutive delay of the corresponding globally feasible schedule $S = \bigcup_{x=1}^{k} S^x$.

2.2 Global optimality

The coordinator problem consists of defining the *set of border constraints* φ to impose on k dispatchers $x = 1, \ldots, k$ at the border of their areas in such a way that the k locally feasible schedules $S^x(\varphi)$ are globally feasible and the maximum consecutive delay over all trains and the whole network is minimized. Specifically, φ includes constraints of two types:

(i) time windows of <earliest, latest> entrance/exit times of a train into and output of a border block section, which must be satisfied by the dispatching solutions provided by all the areas sharing the border block section;

(ii) a sequencing between two trains passing a border block section, which must be satisfied in all the areas sharing the border block section.

Note that each dispatcher can schedule trains in its dispatching area independently from the others and is only constrained to compute a solution $S^x(\varphi)$ compliant with the border constraints φ. We assume that each dispatcher pursues the minimization of maximum consecutive delay in its dispatching area. Moreover, the coordinator may require the following information from the dispatcher of area x:

(a) a lower bound $LB_x(\varphi)$ on the local objective function of area x for a given set of border constraints φ,

(b) a lower bound on the weight of a longest path between any pair of border nodes in area x in any locally feasible solution for a given φ,

(c) the objective function value $UB_x(S^x)$ of a locally feasible solution S^x of area x for a given φ or, alternatively, the information that a locally feasible solution does not exist or cannot be found within the available computation time,

(d) the graph compression of a locally feasible solution $S^x(\varphi)$ for area x and given φ, i.e., the weights of a longest path for each pair of border nodes in area x.

Information (a) and (b) can be used to define a lower bound on the global optimum for a given φ. In fact, the global objective function is the maximum consecutive delay at a set of points in the network that includes those of any dispatching area. Thus, $LB_x(\varphi)$ is also a lower bound for the global objective function. An additional lower bound can be computed by the coordinator by building an alternative graph, called the *coordinator graph* $\mathcal{G}^C(\varphi)$. In $\mathcal{G}^C(\varphi)$ the set of nodes is N_B, the set of fixed arcs F^C is obtained with information (b) and the set of pairs of alternative arcs A^C is given by all the alternative pairs defining precedences between each pair of trains at each border block section. Constraints of type (ii) in φ define a partial selection of A^C, while constraints of type (i) define release dates and deadlines constraints for the border nodes.

The weight $\pi(\varphi)$ of a longest path from 0 to n in $\mathcal{G}^C(\varphi)$ is also a lower bound on the global objective function. This can be computed in a fast way by means of existing graph search algorithms. For example, the algorithm of Floyd and Warshall requires a computing time $O({N_B}^3)$. We call $GLB(\varphi)$ the global lower bound computed as $GLB(\varphi) = \max\{\pi(\varphi), LB_1(\varphi), \ldots, LB_k(\varphi)\}$.

Information (d) can be used to define an upper bound on the global optimum. In fact, from Corollary 2.2 the maximum consecutive delay $GUB(S)$ of a globally feasible schedule $S = \bigcup_{x=1}^{k} S^x(\varphi)$ is the weight of a longest path on the border graph built with information (d). The following result holds.

▶ **Proposition 2.3** (Optimality property)**.** A globally feasible schedule $S = \bigcup_{x=1}^{k} S^x(\varphi)$ is an optimal solution for a given set of coordination constraints φ if $GUB(S) = GLB(\varphi)$.

Proposition 2.3 suggests a branch and bound strategy to find the global optimum to the coordinator problem. Figure 2 describes the interactions between coordinator and dispatchers at each node of the branch and bound tree. The procedure is illustrated in Section 3.

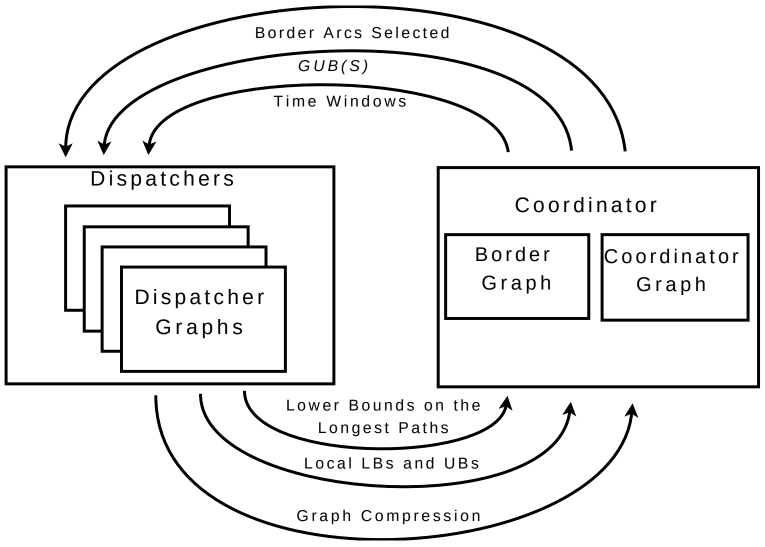

Figure 2 Exchange of relevant data from the coordinator to the dispatchers and vice versa.

Each dispatching area is controlled by a dispatching algorithm. If no local solution is found for an area, a local infeasibility is returned. In this case, the human dispatcher is asked to take some dispatching actions that the dispatching algorithm is not allowed to take, like rerouting some trains or even cancelling a scheduled service. At a global level, the dispatching areas are checked by the coordinator using the border graph and controlled using the coordinator graph. If no global solution is found by the coordination algorithm, a global infeasibility is found. In this other case of infeasibility, the human regional coordinator is asked to recover the situation. This is achieved by imposing further constraints to the coordinator and/or dispatcher graphs.

3 Branch and bound algorithm

This section describes the branch and bound algorithm to solve the coordinator problem and describes its main components. This algorithm is based on the data exchange architecture of Figure 2. At the root node, the dispatchers exchange information (b) of Section 2.2 with the coordinator, that is used to implicate possible arcs in other areas and to set lower bounds on the longest paths between border nodes, which are represented as weighted arcs in the coordinator graph. This exchange of information continues until no value can be increased and terminates with a coordinator graph $\mathcal{G}^C(\emptyset)$ that is used in the subsequent computation.

A starting global upper bound GUB is computed with the starting heuristic described in Corman et al. (2011). The branch and bound procedure starts from the root node $\varphi = \emptyset$. At the generic step of the procedure, the branch and bound nodes contain information φ on the border constraints and are organized in an *active node list* L. Each element is labeled as:

- α if the dispatchers find feasible local schedules with conflicting border decisions;
- β if the dispatchers find feasible local schedules without conflicting border decisions;
- γ if at least one dispatcher does not find a locally feasible solution within the time limit.

Labels are used to guide the order of node exploration during the search. Priority is given to nodes labeled α, then β and finally γ. The intuition behind this choice is that good global upper bounds can be found by first exploring the conflicting border decisions. Nodes labeled α are explored with the FIFO (First In First Out) criterion, whereas the β and γ nodes are visited with a LIFO (Last In First Out) criterion.

Let $l^C(i, j)$ be the longest path from i to j in the coordinator graph. When a *current active node* φ is removed from list L, the coordinator applies the following constraint propagation rules to $\mathcal{G}^C(\varphi)$ in order to enlarge the selection φ as much as possible without branching:

- If $((i, j), (h, k))$ is an unselected pair of alternative arcs in $\mathcal{G}^C(\varphi)$, representing a border decision between areas x and y, and $l^C(0, h) + w_{hk} + l^C(k, n) \geq GUB$, then arc (h, k) is forbidden, and arc (i, j) is implied by φ;
- If $((i, j), (h, k))$ is an unselected pair in $\mathcal{G}^C(\varphi)$, representing some border decision, and $l^C(k, h) + w_{hk} > 0$, then arc (h, k) is forbidden, and arc (i, j) is implied by φ.

In case it is possible to improve the current best upper bound GUB starting from φ, i.e., if $\mathcal{G}^C(\varphi)$ does not contain positive cycles and $\pi(\varphi) < GUB$, the dispatchers are asked to solve their local problems. The dispatchers data (selected border arcs in S^x, local lower bounds $LB_x(\varphi)$ and upper bounds $UB_x(S^x)$, longest path weights) are then sent back to the coordinator which builds the border graph $BG(S)$. In order to compute $LB_x(\varphi)$, the single machine Jackson preemptive schedule described in D'Ariano et al. (2007) is used unless the dispatcher x is able to solve the local problem to optimality, in which case $LB_x(\varphi) = UB_x(S^x)$. If the maximum local lower bound $\max_x\{LB_x(\varphi)\}$ computed by the dispatchers is greater than $\pi(\varphi)$ the value of the global lower bound $GLB(\varphi)$ is updated.

If $BG(S)$ does not contain positive cycles a globally feasible schedule exists and GUB is possibly updated. If $GUB > GLB(\varphi)$, then φ may still improve GUB and a branch is performed. The branch is performed differently for the root node with respect to the other nodes of list L.

For all nodes in L but the root a binary branching strategy is performed as follows:

1. If φ is labeled α, branch on an unselected alternative pair $((i,j),(h,k)) \in A^C$ that was selected in a conflicting way by the local dispatchers. Two new nodes $(\varphi \cup \{(i,j)\})$ and $(\varphi \cup \{(h,k)\})$ are generated and stored in L.
2. If φ is labeled β or γ, branch on the time windows. Choose a time window $< l, u >$ such that $(u - l)$ is minimum over all the time windows in φ and generate two nodes by dividing the time window into two parts of equal size (i.e., $< l, \lfloor (u+l)/2 \rfloor >$ in the first node and $< \lceil (u+l)/2 \rceil, u >$ for the second node). The values l, u for all time windows are integers expressed in minutes, i.e., we consider a minimum size for the time windows equal to 60 seconds.

The branching strategy is different at the root node only if the starting heuristic finds a globally feasible schedule and the root node is of type α. Let us call $(a_1, \bar{a}_1), \ldots, (a_p, \bar{a}_p)$ the p pairs of alternative arcs of the coordinator graph that are selected in a conflicting way by the local dispatchers at the root node. Without loss of generality, let us assume that in the starting feasible schedule the first arc of each pair is selected. The procedure stores $p+1$ nodes, labeled α, in L with the following constraints. For $i = 1, \ldots, p$, the i-th node is described by the constraints $\{a_1, \ldots a_{i-1}, \bar{a}_i\}$. The $(p+1)$-th node is described by the constraints $\{a_1, \ldots, a_p\}$. In this way the procedure skips the intermediate nodes with constraints $\{a_1, \ldots, a_i\}$ with $i < p$. The branch and bound procedure stops when $L = \emptyset$ (*proven optimum*) or a time limit of computation is reached (*open instance*).

4 Description of the test cases

The dispatcher and coordinator procedures have been tested on a large part of the railway network in the South-East of the Netherlands. The network spans ten dispatching areas of the Dutch railway network and includes more than 1200 block sections and station platforms. There are four major stations with complex interlocking systems and dense traffic (Utrecht Central, Arnhem, Den Bosch and Nijmegen), plus another 40 minor stations. The maximum distance between borders of the network is approximately 300 km. We consider the timetable used during operations in 2008, that is an hourly timetable cycle and schedules for local and intercity services, plus international services from/to Germany. The hourly traffic in this regional network is around 25% of the all rail traffic in the Netherlands.

Table 1 summarizes the dispatcher and coordinator graphs for the three network divisions and for the centralized approach. For each delay case, we analyze 30-minute traffic predictions. Column 1 reports the number of areas for each network division. Columns 2-5 give the average number of trains, nodes, fixed arcs and alternative pairs of the dispatcher graphs. Similar information is given for the coordinator graph in Columns 6-9. In the case of 1 area the dispatcher solves the global alternative graph and there is no coordination graph.

Table 1 Alternative graphs for various network divisions.

Network	Dispatcher Graph				Coordinator Graph			
Division	Trains	$\lvert N \rvert$	$\lvert F \rvert$	$\lvert A \rvert$	Trains	$\lvert N_B \rvert$	$\lvert F^C \rvert$	$\lvert A^C \rvert$
1 area	99	3081	3508	3019	-	-	-	-
3 areas	40	1055	1477	1026	21	44	124	30
5 areas	29	656	751	634	43	98	291	73
7 areas	23	477	547	456	48	118	347	86

Stochastic entrance perturbations are considered in order to study delay propagation in the overall network. For each time network division of Table 1, we generate 40 delay instances with an average entrance delay of around 280 seconds, and a maximum entrance delay of around 1650 seconds. In total, 40% of the trains in the hourly timetable are delayed at their network entrance by more than 5 minutes.

5 Computational results

This subsection reports the performance of the branch and bound algorithm for the coordinator problem for the four network divisions and for the 40 instances of 30-minute traffic prediction of the previous section. In the case of 1 area we use the centralized approach described in D'Ariano et al. (2007). For each instance, a globally feasible solution is always computed in a few seconds.

The solution procedures have been implemented in C++ using a Linux Operating System and a high performance computing cluster composed of 8 nodes, each node having 2 Dual Core, 64 bit, AMD Opteron CPUs running at 1800 Mhz and 8 GB RAM. The nodes are connected via a Gigabit Ethernet network. A Message Passing Interface (MPI) architecture (Message Passing Interface Forum, 1994) is adopted in order to achieve efficient inter-process communication and concurrent parallel execution.

Figure 3 shows the percentage of instances for which an optimal solution has been found by the algorithms. The 5 and 7 area problem specifications obtain 95% proven optimal solutions after 30 seconds of computation.

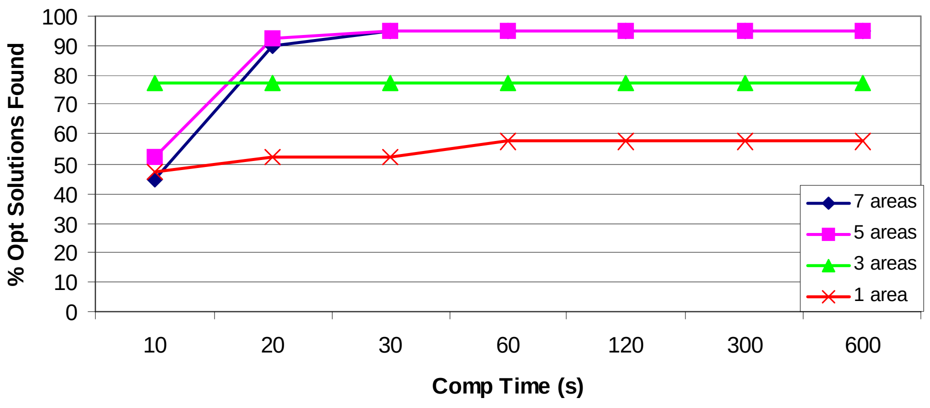

Figure 3 Percentage of optimal solutions found for different network divisions.

Regarding the solution quality, Figure 4 shows the optimality gap $(UB - LB)/LB$ (in percentage) of the solutions for the four network divisions. In the cases with 5 and 7 areas, an average optimality gap smaller than 2% is achieved in the first 20 seconds of computation.

A trade-off is found between the relative complexity of the dispatcher and coordinator problems. As the number of dispatching areas increases, the dispatcher problem is reduced in size for each area, and is therefore easier. At the same time, the complexity of the coordinator problem increases. After 20 seconds of computation, the approach with 5 local areas outperforms the other approaches with smaller or larger numbers of areas. In the other cases there is a larger optimality gap, mostly due to the larger instances to be solved by the dispatchers, which result in larger local upper bounds and smaller local lower bounds. In

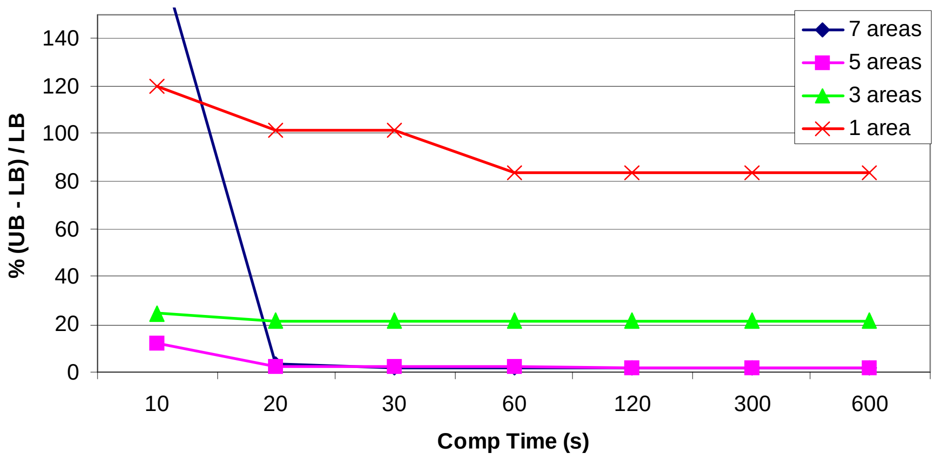

Figure 4 Optimality gap for the different network divisions.

fact, a key element of the branch and bound algorithm for the coordinator problem is that $GLB(\varphi)$ can be set to $UB_x(S^x)$ if the dispatcher problem of area x is solved to optimality. When this occurs frequently, many nodes can be pruned from L thus reducing the optimality gap.

6 Conclusions

This paper presents a novel approach to solve the problem of coordinating the task of multiple dispatchers in presence of disturbances. The problem is formulated as a bilevel program with the objective of minimizing delay propagation. An aggregate coordinator graph is adopted to model coordination constraints while detailed dispatcher graphs model the problem in each dispatching area. Mathematical properties of the proposed formulations allow the development of a branch and bound algorithm to solve the problem. From our computational results we find that distributed approaches are able to deliver better solutions than a centralized approach. Good solutions are produced in a short amount of computation time, compatible with real-time management.

A number of questions remain that require further investigation. We observed that the network division is important to generate feasible and optimal solutions. However, further research is needed to establish a relation between the size and shape of dispatching areas and the effectiveness of the coordinator algorithm. We also observed that the lower bounds can be improved significantly when some dispatcher problems are solved to optimality. This observation suggests a new solution approach for huge instances, in which the size of a dispatching area is artificially reduced only with the aim of obtaining larger lower bounds. We believe that this idea has potential but we did not explore it, yet. There is also a need for more effective starting heuristics, capable of finding feasible schedules with a large number of areas, and effective coordination policies to drive local dispatchers towards global feasibility.

References

1 R. K. Ahuja, C. B. Cunha, and G. Şahin. Network Models in Railroad Planning and Scheduling. In H. J. Greenberg and J. C. Smith, editors, *TutORials in Operations Research*, pages 54–101. 2005.

2 Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2009. Evaluation of green wave policy in real-time railway traffic management. Transportation Research, Part C, 17 (6), 607–616.

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

4 Corman, F., D'Ariano, A.,Pacciarelli, D., Pranzo, M., 2011. Centralized versus distributed systems to reschedule trains in two dispatching areas. Public Transport: Planning and Operations, 2(3), 219–247.

5 A. D'Ariano. Improving real-time train dispatching performance: Optimization models and algorithms for re-timing, re-ordering and local re-routing, *4OR: A Quarterly Journal of Operations Research*, 8(4) 429-432, 2010.

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

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

8 I. A. Hansen and J. Pachl, editors. *Railway Timetable and Traffic: Analysis, Modelling and Simulation.* Eurailpress, Hamburg, 2008.

9 R. M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR Spectrum*, to appear, 2010.

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

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

12 Message Passing Interface Forum. MPI: A message passing interface standard. *The International Journal of Supercomputer Applications and High Performance Computing*, 8 (3), 1994.

13 D. Pacciarelli. Deliverable D3: Traffic Regulation and Co-operation Methodologies - code wp4ur_dv_7001_d. In *project COMBINE 2 "enhanced COntrol centres for fixed and Moving Block sIgNalling systEms" Number: IST-2001-34705.* 2003.

14 C. Strotmann. *Railway Scheduling Problems and their decomposition.* PhD thesis, Universität Osnabrück, 2007.

15 L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: a bibiliography review. *Journal of Global Optimization*, 5:291–306, 1994.