19th International Symposium on Transportation and Traffic Theory

Optimal inter-area coordination of train rescheduling decisions

F. Corman¹, A. D’Ariano², D. Pacciarelli²*, M. Pranzo³

¹Transport and Planning Department, Delft University of Technology; Centre for Industrial Management, Katholieke Universiteit Leuven, Celestijnenlaan 300A, 3001 Heverlee, Belgium
francesco.corman@cis.kuleuven.be

²Dipartimento 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

³Dipartimento 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.

Keywords: Train Delay Minimization; Schedule Coordination; Bilevel Programming.

* Corresponding author. Tel.: +39 06 5733 3238; fax: +39 06 5733 3612.
E-mail address: pacciarelli@dia.uniroma3.it

1877–0428 © 2011 Published by Elsevier Ltd.
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, coordinators are responsible for the traffic management over a railway network of \( k \) areas with a global overview of the traffic flow and control the rescheduling decisions taken by 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.

![Figure 1: Interaction between coordinator and dispatchers.](image)

Due to the complexity of the overall train rescheduling problem, decision support systems (DSSs) are needed to help dispatchers and coordinators 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 [11,16,20,26] and heuristic solution procedures [1,5,6,9,12,24]. Recent surveys on models and algorithms for the dispatcher problem can be found in [2,8,15]. 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 safety and operational constraints [4,6,8,9,11,24].

Differently from the dispatcher problem, the coordinator problem to be solved 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 (dispatching) solutions or detecting global infeasibility and possible coordination actions for recovery [7,18,25]. 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 [22]. Train movements in the local dispatching areas are modeled by an alternative graph formulation [17], while a higher level of control considers aggregate information about the local solvers. The implementation of these methodologies are reported in [18] for two test cases of the Dutch railway network, and a practical pilot is also described for one of the two test cases.

In [25], 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 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. Hypothetical examples are studied with up to 16 trains and 73 block sections.

The coordinator problem has been recently addressed by [7] 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 [11] 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. The contribution of this paper can be summarized as follows:

- The coordinator problem is defined as a bilevel optimization problem [27] and the approach of [7] is improved to assess the quality in addition to the feasibility of global schedules;
- Mathematical properties of the inter-area coordinator problem are provided and exploited in the solution algorithms;
- A branch and bound algorithm is developed in order to solve the coordinator problem;
- The algorithm is tested on a portion of the Dutch railway network with busy traffic and various network divisions.

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 [17] models the dispatcher problem. The blocking time theory is used to compute arc weights (see, e.g., [14]) 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 [11] is then used 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.

Based on this information, 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:
The first property allows to prove global feasibility of the union of locally feasible schedules; 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 global 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 set 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 an optimal or 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 [7] and with the centralized approach described in [11].

The outline of the paper is as follows. Section 2 describes the formulation of the multi-area train scheduling problem and explores its properties, which are then used to design a branch and bound algorithm for the coordinator problem. Section 3 shows the test region and presents the computational results of the branch and bound algorithm for various network divisions. Section 4 concludes the paper and highlights further research directions for the coordinator problem.

2. MULTI-AREA TRAIN SCHEDULING PROBLEM

This section introduces the terminology used in this paper and describes the mathematical formulation of the dispatcher and coordinator problems. We then give an illustrative example showing how global infeasibility can be detected and resolved. Mathematical properties of the problem are used to design a branch and bound algorithm for the coordinator problem.

2.1. TERMS AND DEFINITIONS

A rail network is composed of railway lines connecting stations. Each line, and its stations, are divided into block sections, i.e., portions of the railway delimited by signals. Careful scheduling is necessary since safety rules require that at most one train can occupy a block section at any time. Detailed descriptions of railway signaling systems and traffic control regulations can be found in [3,8,13,14].

The passage of a train through a particular block section is an operation. A train route is an ordered sequence of operations to be processed during a service. The timing of a train route specifies the start time $t_i$ of all the operations in the route. Each operation requires a running time, which depends on the actual speed profile followed by the train while traversing the block section. Safety regulations impose a minimum distance separation between trains, which translates into a minimum setup time (time headway) between the exit of a train from a block section and the entrance of the subsequent train into the same block section. This time includes the interval between the entrance of the head of the train into a block section and the exit of its tail (the last axle) from the previous one, plus additional time margins to release the occupied block section and to cover the sight distance (see, e.g., [21]).

The timetable describes the movement of all trains circulating in the network by specifying, for each train, the planned arrival/passing times at a set of relevant points along its route (e.g., stations, junctions, and the exit point of the network). At stations, a train is not allowed to depart from a platform stop before its scheduled departure time and is considered late if it arrives at the platform later than its scheduled arrival time.
Timetables are designed to satisfy all traffic regulations. However, unexpected events occur during operations, which cause delays with respect to the operations scheduled in the timetable. The delay may propagate to other trains, causing a domino effect of increasing disturbances. Rescheduling must cope with temporary infeasibility by adjusting the timetable of each train. The dispatcher’s task is to regulate traffic with the main objective of minimizing train delays in such a way that the new schedule is compliant with railway rules and with the actual position of each train.

A potential conflict occurs when two or more trains contend for the same block section simultaneously, requiring a decision on the train order. A set of trains causes a deadlock when each train in the set contends for a block section ahead which is not available, due to occupancy/reservation by another train in the set. We define an entrance perturbation as a set of trains delayed at their entrance in a dispatching area, due to the propagation of delays from previous areas or to other sources of delay. The total delay is the difference between the estimated train arrival time and the scheduled time at a relevant point in the network, and can be divided into two parts. The initial delay is caused by earlier failures and disturbances, while the consecutive delay is caused when solving potential train conflicts, and represents the amount of time that a train is delayed beyond the earliest possible departure from a relevant point. Note that, the initial delay is a constant delay that cannot be avoided, while the consecutive delay is a variable delay that depends on the rescheduling decisions taken by traffic controllers. In this paper we therefore address the minimization of consecutive delays. Specifically, we deal with the maximum consecutive delay, due to both practical and algorithmic considerations. From the practical point of view, we observe that small delays can be absorbed more easily by timetable reserves with respect to large delays. Therefore, a solution with many small delays can be preferable, from a global perspective, to a solution with few large delays. The minimization of the maximum consecutive delay better achieves this result with respect, e.g., to the minimization of the average delay. From the algorithmic point of view, the minimization of the maximum delay makes possible the computation of effective lower bounds and constraint propagation strategies, which can be used to design effective solution algorithms [11].

2.2. MATHEMATICAL FORMULATION

The coordinator problem is formulated in accordance with the two-level hierarchy of [27] in which 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, which allow an accurate representation of the problem at the level of signal aspects and operational rules [6, 8, 9, 11, 18].

The alternative graph is a triple \( G=(N,F,A) \), 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 \( I, \ldots, n-1 \)). 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 [11].

Given a solution \( S \), \( \hat{l}(i,j) \) 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 \( t_i = \hat{l}(0,i) \). Note that \( t_0 \) is a constant equal to 0. A feasible schedule is an optimal solution if \( \hat{l}(0,n) \) is minimum over all the feasible schedules. The general alternative graph formulation can be viewed as the following disjunctive program:

\[
\begin{align*}
\text{min} \quad & t_n - t_0 \\
\text{s.t.} \quad & t_j - t_i \geq w_{ij}, \quad (i,j) \in F \\
& (t_j - t_k \geq w_{kj}) \lor (t_l - t_n \geq w_{lh}), \quad ((k,j),(h,i)) \in A
\end{align*}
\]
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 $i=1,...,n-1$, 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 $t_{\sigma(i)} \geq t_i + w_{\sigma(i)}$ must hold, where $w_{\sigma(i)}>0$ 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 [6,8,9,11]. 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 $(t_i - t_{\sigma(i)} \geq w_{\sigma(j)}) \lor (t_i - t_{\sigma(i)} \geq w_{\sigma(j)})$, where $w_{\sigma(i)}>0$ and $w_{\sigma(j)}>0$ are the minimum required 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.

Figure 2 shows the alternative graph formulation for two trains (A and B) running along paths merging at a junction. Here, $i$ and $j$ correspond to the entrance of A and B in block section 1, while $k$ and $h$ correspond to the entrance of A and B in block section 2. A fixed arc $(i,k)$ (depicted with solid arrows) represents the path of train A through block section 1. The arc weight $w_{ik}$ corresponds to its running time $p_i$. Similarly, the arc weight $w_{jh}$ corresponds to the running time $p_j$ of train B through block section 1. Since the junction cannot host A and B at the same time (i.e., two jobs in the alternative graph require the same resource), there is a potential conflict. In this case, a processing order must be defined between the incompatible operations, and we model this constraint by a suitable pair of alternative arcs from the set $A$ (depicted with dashed arrows). Each alternative arc models a possible precedence relation between two operations, and the weight of an alternative arc is the minimum time headway between the associated trains.

![Diagram of the dispatcher graph for two trains at a merging point.](image)

In the example of Figure 2, given a fixed arc $(i,k)$, we say that $k$ is the successor of $i$ and denote it by $k = \sigma(i)$. By letting $i$ and $j$ be the two conflicting operations at the merging point, we model the two possible processing orders with the pair of arcs $((k,j),(h,i)) \in A$. If $i$ is scheduled before $j$, the alternative arc $(k,j)$ is selected and A precedes B. The weight $w_{ij}$ of the alternative arc $(k,j)$ corresponds to the setup time between $i$ and $j$ ($t_j \geq t_i + w_{ij}$). Similarly, if $j$ is scheduled before $i$, then B precedes A and the weight of $(h,i)$ is the setup time $w_{hi}$ ($t_i \geq t_h + w_{hi}$). Specifically, if $i$ is processed first, then $j$ must wait for the completion of the running time $p_i$ and the start of $\sigma(i)$ plus the setup time $w_{\sigma(i)}$ between $i$ and $j$.

Figure 3 shows the alternative graph formulation for two trains (A and B) running in the opposite direction at a crossing point. For the two conflicting operations $i$ and $j$, we introduce the pair of arcs $((k,j),(h,i)) \in A$. If $i$ is
scheduled before \( j \), the arc \((k,j)\) is selected and \( A \) precedes \( B \). Similarly, if \( j \) is scheduled before \( i \), the arc \((h,i)\) is selected and \( B \) precedes \( A \).

![Diagram](image-url)  

Figure 3: The dispatcher graph for two trains at a crossing point.

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 can be represented by the quantity \( t_n - t_0 \) by associating suitable weights with the arcs ending at node \( n \), as in [9].

### 2.3. GLOBAL FEASIBILITY

In principle, the feasibility of a global solution can be checked by building a global alternative graph 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 the resulting global graph the local solutions are globally feasible. However, a drawback of the global alternative graph formulation is the size of the resulting graph 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 introduce the following propositions. Proof of these propositions is straightforward and therefore omitted.

**Proposition 2.1 (Node Contraction)** Given a feasible selection \( S_x \) for a dispatching area \( x=1,...,k \), let \( G_x(S_x) = (N_x,F_x \cup S_x) \) be the associated graph and let \( i \in N_x \) be a node that does not belong to the border of the area. Let \( F_i \) be the set of arcs \((h,i)\in F_x \cup S_x\), with \( h \in N_x \setminus \{i\} \), and let \( F_i = \{(i,j) \in F_x \cup S_x, j \in N_x \setminus \{i\}\} \). Let \( C(i) = \{(h,j) : (h,i) \in F_x \cup S_x ; (i,j) \in F_x \cup S_x \} \), each having weight \( w'_{hj} = w_{hi} + w_{ij} \). The contraction of node \( i \) produces a graph \( G'(S_x) = (N_x \setminus \{i\}, F_x \cup S_x \setminus C(i) \cup (F_i \cup F_j)) \). Graph \( G'(S_x) \) is equivalent to \( G(S_x) \), i.e., for any pair of nodes \( h \in N_x \setminus \{i\} \) and \( k \in N_x \setminus \{i,h\} \) the weight of a longest path from \( h \) to \( k \) is always equal to \( l_{S_x}(h,k) \). Moreover, after the computation of all values \( t_n \), \( h \in N_x \setminus \{i\} \), \( t_i \) can be obtained from \( G'(S_x) \) as \( t_i = \max\{t_i + w_{hi} : (h,i) \in F_i\} \).

**Proposition 2.2 (Arc Deletion)** Given a feasible selection \( S_x \) for dispatching area \( x \), let \( G_x(S_x) = (N_x,F_x \cup S_x) \) be the associated graph and let \((i,j) \in F_x \cup S_x\) be an arc of weight \( w_{ij} \). If \( l_{S_x}(i,j) > w_{ij} \) or \( l_{S_x}(i,j) = w_{ij} \) and there is a longest path from \( i \) to \( j \) in \( G_x(S_x) \) different from arc \((i,j)\), then \((i,j)\) is redundant and can be removed from the graph producing a graph \( G''(S_x) = (N_x,F_x \cup S_x \setminus \{(i,j)\}) \). \( G''(S_x) \) is equivalent to \( G(S_x) \).

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 feasible selection \( S_x \), the graph compression consists in contracting all the nodes in \( N_x \setminus N_B \) and then deleting all the redundant arcs. By construction, the resulting graph is equivalent to \( G'(S_x) \), i.e., there is an arc \((i,j)\) in the compressed graph if and only if there is a directed path from \( i \) to \( j \) in \( G'(S_x) \), and \((i,j)\) is weighted with \( l_{S_x}(i,j) \).
Let us now consider a region divided into $k$ dispatching areas, let $S^x$ be a locally feasible schedule for dispatching area $x = 1, ..., k$ and let $S = \bigcup_{x=1}^{k} S^x$ be the union of the $k$ selections. 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 $G(S^x)$ for each dispatching area $x = 1, ..., k$, 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 $G(S^x)$ is $w_{ij} < \infty$, for some $x = 1, ..., k$. Clearly, redundant arcs in $BG(S)$ can be deleted. The following property holds.

**Theorem 2.3 (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.

**Proof.** Consider the global alternative graph $(N,F,A)$ associated with the whole region, i.e., with $N = \bigcup_{x=1}^{k} N^x$, $F = \bigcup_{x=1}^{k} F^x$ and $A = \bigcup_{x=1}^{k} A^x$. If $S$ is a globally feasible selection, $(N,F,U,S)$ does not contain positive cycles, and this property is preserved in the graph compression obtained by contracting all nodes in $N \setminus N_B$, i.e., in $BG(S)$. Let us now prove that if $BG(S)$ has no positive cycles then $S$ is a globally feasible selection. To this aim, we first prove that $S$ is a valid global selection, i.e., exactly one arc is selected for each alternative pair of $A$. This must be proved only for those pairs that are shared among different areas, i.e., that are associated with a pair of trains $\theta_1$ and $\theta_2$ traversing a border block section. Assume by contradiction that a dispatching area sharing that block section schedules $\theta_1$ before $\theta_2$ and another area schedules $\theta_2$ before $\theta_1$. Then, in the former area there will be a positive weight path between the node associated with the entrance of $\theta_1$ into the border block section and that associated with the entrance of $\theta_2$. In the latter area there will be a path from the entrance of $\theta_2$ to the entrance of $\theta_1$, i.e., there will be a positive cycle in $BG(S)$, a contradiction. So, if $BG(S)$ has no positive cycles $S$ is a valid global selection.

It remains to prove that there are no positive cycles in $(N,F \cup S)$. Such a cycle can either belong entirely to a local area, or can involve nodes belonging to several areas. The former case cannot occur since all the local selections $S^1, ..., S^k$, are locally feasible schedules. In the latter case, a positive cycle must involve at least two border nodes. Let us consider two consecutive border nodes $i$ and $j$ in a positive cycle, i.e., there are no other border nodes between $i$ and $j$ in the cycle. All the nodes between $i$ and $j$ in the cycle must therefore belong to the same dispatching area, say the $x$-th area, and the weight of the path from $i$ to $j$ is less than or equal to $f^x(i,j)$. In this case, the border graph contains arc $(i,j)$ of weight $f^x(i,j)$. By repeating the same argument for each pair of consecutive border nodes in the positive cycle of the global graph we obtain that a cycle with positive weight is contained also in the border graph. Therefore, if $BG(S)$ does not contain positive weight cycles the same holds for $(N,F \cup S)$, i.e., $S$ is a globally feasible selection, and this concludes the proof.

**Corollary 2.4 (Global objective function)** Consider a global area composed of $k$ local areas and a locally feasible schedule $S^x$ for each dispatching area $x = 1, ..., k$. 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 $S = \bigcup_{x=1}^{k} S^x$.

**Proof.** From Theorem 2.3 it follows that $S$ is a globally feasible selection. To complete the proof it is sufficient to observe that the border graph is a graph compression of $(N,F,U,S)$ obtained by contracting all nodes in $N \setminus N_B$. Since the graph compression does not modify the weight of the longest path from $0$ to $n$, the thesis follows.

### 2.4. TWO-AREA EXAMPLE

We give an illustrative example with two dispatching areas (shown as $x$ and $y$ in Figure 4). Figure 4 (a) presents the infrastructure layout with the relevant block sections (labelled 1-12). Area $x$ includes block sections from 1 to 7, while area $y$ includes block sections 6-12. The border between the areas consists of block sections 6 and 7. Four trains (labelled A-D) are scheduled to traverse both areas. Trains A and B run on block sections 1, 3, 5, 6, 8, 9, 11, while trains C and D run on block sections 12, 10, 8, 7, 5, 4, 2. We suppose that at the start time $t_0=0$ each train is at the end of its first block section, e.g., at time 0 train A is at the end of block section 3 and is ready to enter block section 5. We assume the running time of each train on each block section is 2 time units while the setup time between any pair of trains at each block section is 1 time unit. According to the timetable, the scheduled exit times of A from block sections 6 and 11 are 11 and 17, respectively. The scheduled exit time of B at block sections 6 and 11 is 14 and 20, the scheduled exit time of C (D) at block sections 7 and 2 are 11 and 17 (14 and 20).
Figure 4: A small network and the traffic situation at $t_0$ (a), the corresponding alternative graph (b), the dispatcher graphs given the division into 2 areas (c) and the resulting border graph (d).
Figure 4 (b) shows the global alternative graph formulation of the problem. A node of the graph is identified by the pair (train, block section), except for the last node of every train, representing the exit from the network and identified by the pair (train, out), and for the dummy nodes 0 and n. For instance, for train A running on block section 5, node A5 represents the entrance of the train into the block section, its running time is the weight of the fixed (solid black) arc (A5,A6), while its setup time with train B on block section 5 is the weight of the alternative (dotted grey) arc (A6,B5). The fixed arcs from node 0 are weighted with the release time of the associated train in the area, while the weights on the arcs into node n ensure the correspondence between the longest path in a solution and its maximum consecutive delay. The starting position of trains implies the selection of the alternative pairs reported in Figure 4 (b). Train B must follow train A, and train D must follow train C on each block section of their routes. In Figure 4 (b) we do not show the eight unselected alternative pairs modeling blocking constraints on block sections 5 and 8, e.g., ((A6,C5),(C4,A5)). The (solid black) arcs entering node n, e.g., (Aout,n), model the objective function for the concerned trains.

Figure 4 (c) depicts the optimal local solutions found by each dispatcher for areas x and y. In both schedules, the starting train positions constrain train A to precede train B and train C to precede train D. The maximum consecutive delay, i.e., the weight of the longest path from 0 to n, is equal to -5, that corresponds to the latest train being early by 5 time units. This is obtained in area x, e.g., by the path 0, A5, A6, A8, B6, C5, n; and in area y, e.g., by the path 0, C8, C7, D8, D7, A8, n.

In the solution for area x, the sequence of trains on block section 5 is ABCD, train C leaves the network at time 12, train D at time 15, while A and B traverse the border with area y at times 4 and 7, respectively. In the solution for area y, the sequence of trains on block section 8 is CDAB, train A leaves the network at time 12, train B at time 15, while C and D traverse the border with area x at times 4 and 7, respectively. Note that the order of trains at each border block section is the same for both areas, however the two solutions are clearly incompatible.

Figure 4 (d) represents the border graph based on the incompatible dispatcher schedules of Figure 4 (c). The weight on the arcs reports the weight of the corresponding longest paths in the dispatcher graphs. There is a positive cycle A8, B6, C5, D7, A8 with weight 4. In this case, the coordinator must recover the overall schedule feasibility by asking the dispatchers to resolve their local problems so that the arc (B6,C5) (or the arc (D7,A8)) is forbidden. Note that in this case the coordinator cannot change the order of trains at the border, since this is constrained by the initial train positions. However, the coordinator can force one of the two dispatchers to change its solution by restricting the time windows for crossing the area borders, as discussed in the next section.

2.5. GLOBAL OPTIMALITY

In this section we formally define the coordinator problem and its mathematical properties. The coordinator problem consists of defining the set of border constraints φ to impose on k dispatchers x=1,...,k at the border of their areas in such a way that the k locally feasible schedules S(φ) 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(φ) 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,φ) 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 $\varphi$,

(c) the objective function value $UB_x(S')$ of a locally feasible solution $S'$ of area $x$ for a given $\varphi$ 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 $S'(\varphi)$ for area $x$ and given $\varphi$, 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 $\varphi$. 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 $G^C(\varphi)$. In $G^C(\varphi)$ 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 $\varphi$ 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 $G^C(\varphi)$ 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 follows:

$$GLB(\varphi)= \max \{ \pi(\varphi), LB_1(\varphi), \ldots, LB_k(\varphi) \}.$$  

Information (d) can be used to define an upper bound on the global optimum. In fact, from Corollary 2.4 the maximum consecutive delay $GUB(S)$ of a globally feasible schedule $S=\bigcup_{x=1,\ldots,k} S_x(\varphi)$ is the weight of a longest path on the border graph built with information (d). The following result holds.

**Proposition 2.5 (Optimality property)** A globally feasible schedule $S=\bigcup_{x=1,\ldots,k} S_x(\varphi)$ is an optimal solution for a given set of coordination constraints $\varphi$ if $GUB(S)=GLB(\varphi)$.

**Proof.** The proof follows from the definitions of $GUB(S)$ and $GLB(\varphi)$.

---

Figure 5: The coordinator graph for $\varphi^*$ in the example of Figure 4.
Figure 6: Time-distance diagram for $\varphi^*$ in the example of Figure 4.

Figure 5 shows the coordinator graph $G^C(\varphi^*)$ associated with an optimal set $\varphi^*$ for the example of Figure 4. Note that the two alternative pairs for this coordinator graph are implied by the initial positions of the four trains. Implied alternative arcs are depicted in dotted grey in Figure 5. The only constraints in $\varphi^*$ are of type (i), in this case $\varphi^* = \{t_{B6} \leq 5\}$, modeled with arc $(B6,0)$ with weight -5. In area $x$ this arc implies the selection of $(B6,C5)$ in the alternative pair $((B6,C5),(C4,B5))$. Thus, there is an arc $(B6,C5)$ with weight 1 also in $G^C(\varphi^*)$. In area $y$ arc $(B6,0)$ implies the selection of $(A9,D8)$ in the alternative pair $((A9,D8),(D7,A8))$, which causes a path from $A8$ to $D7$ with weight 5 in area $y$, i.e., an arc $(A8,D7)$ with weight 5 in $G^C(\varphi^*)$. Thus, $\pi(\varphi^*) = -3$ (obtained, e.g., by the path $0, A6, A8, D7, D5, n$).

Figure 7: Exchange of relevant data from the coordinator to the dispatchers and vice versa.
Figure 6 shows a time-space diagram for the coordinator graph $G_1^C(\phi')$ and the associated local solutions. The running times are depicted as line segments, while the setup times are depicted as rectangular boxes. The locally optimal solutions of the two areas are the sequence ABCD on block section 5 (with $UB_x(S_x) = LB_x(\phi') = -5$) and the sequence CADB on block section 8 (with $UB_y(S_y) = LB_y(\phi') = -3$), that are globally feasible and globally optimal, with $GUB(S) = -3$.

Proposition 2.5 suggests a branch and bound strategy to find the global optimum to the coordinator problem. Figure 7 describes the interactions between coordinator and dispatchers at each node of the branch and bound tree. The procedure is illustrated in Section 2.6.

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 additional constraints to the coordinator and/or to the dispatcher graphs.

2.6. BRANCH AND BOUND ALGORITHM

This subsection shows the pseudo-code of our algorithm to solve the coordinator problem and describes its main components. The branch and bound algorithm is based on the data exchange architecture of Figure 7. At the root node, the dispatchers exchange information $(b)$ of Section 2.5 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 $G_1^C(\varnothing)$ that is used in the subsequent computation.

A starting global upper bound $GUB$ is computed with the following starting heuristic, a generalization to $k$ areas of the iterative heuristic procedure described in [7] for two areas. Starting from $\varnothing = \varnothing$, all dispatchers $x=1,\ldots,k$ compute a feasible schedule $S_x(\varnothing)$ for their areas and send the graph compression $G_x(S_x)$ to the coordinator. For each pair of adjacent dispatchers, the procedure checks all border constraints for possible violation and takes the following coordination actions in order to remove any violation:

- If the entrance time of a train in an area is earlier than the exit time of the same train from the previous area, the entrance time is set equal to the exit time of the same train from the previous area.
- If two dispatchers select incompatible train orders at a border block section the following coordination action is undertaken to regain feasibility. If the infeasibility involves two trains running in the same direction, the train order of the previous area is imposed on the following area. If the infeasibility involves two trains running in opposite directions, the train order is obtained by giving precedence to the first train reaching the border in the dispatcher graph of a local area.
- If there is a positive cycle in the associated border graph there must be at least one train, among those involved in the cycle, for which the exit time from an area is larger than the entrance time in the next area in the local solutions of the associated dispatcher graphs. The procedure selects among these trains the earliest exit time $\tau$ scheduled by the dispatcher graphs, and then requires the entrance time of the associated train in the next area to be equal to $\tau$. In other words, the procedure sets only one additional constraint per train and area for each detected cycle.

After each coordination action a new iteration takes place in which the dispatchers compute local solutions compliant with the coordinator constraints. The starting heuristic terminates when a globally feasible solution is found or when a local infeasibility is found or when a maximum number of iterations is reached. In the two latter cases the initial $GUB$ is set to $+\infty$. In our computational experiments, we set the maximum number of iterations to 5.
since preliminary tests showed that a greater number of iterations does not improve the quality of the upper bound significantly.

A sketch of the branch and bound procedure is shown in Figure 8. The procedure starts from the root node $\varphi=\emptyset$. At the generic step of the procedure, the branch and bound nodes contain information $\varphi$ on the border constraints and are organized in an active node list $L$. Each element is labeled as:

- $\alpha$ if the dispatchers find feasible local schedules with conflicting border decisions;
- $\beta$ if the dispatchers find feasible local schedules without conflicting border decisions;
- $\gamma$ if at least one dispatcher is unable to find a locally feasible solution within the time limit.

---

**BRANCH AND BOUND ALGORITHM**

Initialize the active node list $L$ with the root node $\varphi=\emptyset$.

First round of data exchange between coordinator and dispatchers based on the initial position of trains, a starting global upper bound $GUB$ is computed by the starting heuristic.

A global lower bound $GLB(\varphi)$ for the current active node $\varphi$ is computed as in Equation (1),

- if $(GLB(\varphi) \geq GUB)$ then close search (root optimum).
- while $(L \neq \emptyset)$ and (computation time < time limit)
  
  begin
  
  if ($\exists \alpha$ nodes in $L$) then $\varphi =$ node in $L$ with highest priority and label $\alpha$,
  else if ($\exists \beta$ nodes in $L$) then $\varphi =$ node in $L$ with highest priority and label $\beta$,
  else if ($\exists \gamma$ nodes in $L$) then $\varphi =$ node in $L$ with highest priority and label $\gamma$,
  
  Remove $\varphi$ from $L$,
  
  Build the coordinator graph $G^C(\varphi)$ and select possible implications,
  
  Set an initial value $GLB(\varphi)=\pi(\varphi),$
  
  if $(GLB(\varphi) < GUB)$ and (there is no positive cycle in $G^C(\varphi)$) then
  
  begin
  
  Send the coordinator constraints $\varphi$ (border arcs, time windows) and $GUB$ to the dispatchers,
  
  Receive from the dispatchers relevant data (border arcs, $LB_s(\varphi)$ and $UB_s(S')$, longest paths),
  
  if $(GLB(\varphi) < \max_{i=1,...,k} \{ LB_s(\varphi) \})$ then $GLB(\varphi) = \max_{i=1,...,k} \{ LB_s(\varphi) \},$
  
  if (there is no feasible schedule for, at least, a dispatcher graph) then
  
  label $\varphi$ with $\gamma$ and set $GUB(\varphi)=+\infty,$
  
  else begin
  
  Build the border graph $BG(S)$,
  
  if (there is a positive cycle in $BG(S)$) then
  
  begin
  
  $GUB(\varphi) = +\infty,$
  
  if (the dispatcher graphs return conflicting border decisions) then label $\varphi$ with $\alpha$,
  
  else label $\varphi$ with $\beta$,
  
  end
  
  else begin
  
  Compute $GUB(\varphi)$ as the longest path on $BG(S),$
  
  if $(GUB(\varphi) < GUB)$ then $GUB = GUB(\varphi),$  
  
  if $(GUB \leq \min_{\varphi \in L} GLB(\varphi))$ then close search (proven optimum).
  
  Label $\varphi$ with $\beta,$
  
  end
  
  end
  
  end
  
  Execute a branch from $\varphi$ and add the resulting nodes to $L$ with the label of $\varphi,$
  
  end
  
  if (computation time < time limit) then close search (open instance) else close search (proven optimum).

---

Figure 8: Pseudo-code of the branch and bound procedure for the coordinator problem
Labels are used to guide the order of node exploration during the search. Priority is given to nodes labeled $\alpha$, then $\beta$ and finally $\gamma$. The intuition behind this choice is that good global upper bounds can be found by first exploring the conflicting border decisions. Nodes labeled $\alpha$ are explored with the FIFO (First In First Out) criterion, whereas the $\beta$ and $\gamma$ nodes are visited with a LIFO (Last In First Out) criterion.

Let $\ell^i(j)$ be the longest path from $i$ to $j$ in the coordinator graph. When a current active node $\phi$ is removed from list $L$, the coordinator applies the following rules to the coordinator graph $G^C(\phi)$ in order to enlarge the selection $\phi$ as much as possible:

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

In case it is possible to improve the current best upper bound $GUB$ starting from $\phi$, i.e., if $G^C(\phi)$ does not contain positive cycles and $\pi(\phi) < GUB$, the dispatchers are asked to solve their local problems. The dispatchers data (selected border arcs in $S'$, local lower bounds $LB_x(\phi)$ and upper bounds $UB_x(S')$, longest path weights) are then sent back to the coordinator which builds the border graph $BG(S)$. In order to compute $LB_x(\phi)$, the single machine Jackson preemptive schedule described in [11] is used unless the dispatcher $x$ is able to solve the local problem to optimality, in which case $LB_x(\phi) = UB_x(S')$. If the maximum local lower bound $\max_x\{LB_x(\phi)\}$ computed by the dispatchers is greater than $\pi(\phi)$ the value of the global lower bound $GLB(\phi)$ is updated.

If $BG(S)$ does not contain positive cycles a globally feasible schedule exists and $GUB$ is possibly updated. If $GUB > GLB(\phi)$, then $\phi$ 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:

- If $\phi$ is labeled $\alpha$, 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 $\phi \cup \{(i, j)\}$ and $\phi \cup \{(h, k)\}$ are generated and stored in $L$.
- If $\phi$ is labeled $\beta$ or $\gamma$, branch on the time windows. Choose a time window $<l, u>$ such that $(u-l)$ is minimum over all the time windows in $\phi$ and generate two nodes by dividing the time window into two parts of equal size (i.e., $<l, [(u+l)/2]>$ in the first node and $<(u+l)/2, u>$ 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 $\alpha$. Let us call $(a_1, \bar{a}_1), ..., (a_p, \bar{a}_p)$ 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 $\alpha$, in $L$ with the following constraints. For $i=1, ..., p$, the $i$-th node is described by the constraints $\{a_i, \ldots, a_i, \bar{a}_i\}$. The $(p+1)$-th node is described by the constraints $\{a_i, \ldots, a_p\}$. In this way the procedure skips the intermediate nodes with constraints $\{a_i, \ldots, a_i\}$ with $i < p$.

The branch and bound procedure stops when $L = \emptyset$ (proven optimum) or a time limit of computation is reached (open instance).

3. REAL-WORLD TEST CASE

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 [19] is adopted in order to achieve efficient inter-process communication and concurrent parallel execution.
3.1. DESCRIPTION OF THE INSTANCES

The dispatcher and coordinator procedures have been tested on a large part of the railway network in the South-East of the Netherlands (see Figure 9). 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. The two main traffic directions are served by the line between Utrecht and Arnhem (towards Germany) and the line between Utrecht and Den Bosch (from Amsterdam towards Eindhoven and the southern part of the country).

Figure 9: Infrastructure layout of the regional network considered.

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.

Figure 10 presents a graphical representation of the timetable (in terms of line frequency) for different network divisions: 1 area, 3 areas, 5 areas and 7 areas. Every solid line indicates that there are two trains per hour per direction on a specific line. Light green lines are local services and dark blue lines are intercity services. The dotted line represents one hourly international service. Finally, the thick grey lines represent the boundary of each division.
Figure 10: Hourly frequency of each line and network divisions.

Table 1 summarizes the dispatcher and coordinator graphs for the three network divisions and for the centralized approach. We analyze two time horizons of traffic prediction in order to investigate delay propagation on graphs of increasing size. Column 1 reports the number of areas and Column 2 the traffic prediction time horizons. Columns 3-6 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 7-10. In the case of 1 area the dispatcher solves the global alternative graph and there is no coordination graph.

<table>
<thead>
<tr>
<th>Network Division</th>
<th>Time Horizon</th>
<th>Dispatcher Graph</th>
<th>Coordinator Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Trains</td>
<td>[N]</td>
</tr>
<tr>
<td>1 area</td>
<td>30 min</td>
<td>99</td>
<td>3081</td>
</tr>
<tr>
<td>3 areas</td>
<td>30 min</td>
<td>40</td>
<td>1055</td>
</tr>
<tr>
<td>5 areas</td>
<td>30 min</td>
<td>29</td>
<td>656</td>
</tr>
<tr>
<td>7 areas</td>
<td>30 min</td>
<td>23</td>
<td>477</td>
</tr>
<tr>
<td>1 area</td>
<td>60 min</td>
<td>154</td>
<td>6136</td>
</tr>
<tr>
<td>3 areas</td>
<td>60 min</td>
<td>65</td>
<td>2102</td>
</tr>
<tr>
<td>5 areas</td>
<td>60 min</td>
<td>52</td>
<td>1309</td>
</tr>
<tr>
<td>7 areas</td>
<td>60 min</td>
<td>39</td>
<td>947</td>
</tr>
</tbody>
</table>
Stochastic entrance perturbations are considered in order to study delay propagation in the overall network. Realistic perturbations have been generated with the Weibull distribution described in [28], whose parameters have been designed by analyzing more than 33000 train events (arrivals, departures, dwell processes and passing times) recorded at Utrecht Central Station in April 2008. In order to analyze major disruptions, in our experiments we doubled the scale parameter of the Weibull distribution. The effect of this modification is roughly to double the variability of the initial delays, while leaving their average value virtually unchanged. For each time horizon of traffic prediction and 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.

3.2. RESULTS FOR 30-MINUTE TRAFFIC PREDICTIONS

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 [11]. For each instance, a globally feasible solution is always computed in a few seconds.

Figure 11 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 11: Percentage of optimal solutions found for different network divisions.

Regarding the solution quality, Figure 12 shows the results in terms of maximum and average consecutive delays. When an algorithm returns no solution (see the 7-area division during the first 20 seconds of computation), we impose a large failure penalty for the maximum and average consecutive delays. In general, after 20 seconds of computation the distributed approach with coordinator outperforms the centralized approach, with up to 20% delay reduction in terms of maximum consecutive delay. Even if the main objective function is the maximum consecutive delay, the average consecutive delay is also relevant. It is interesting to analyze the trade-off between the relative complexity of the dispatcher problem and the coordinator problem. 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. In our experiments, after 20 seconds of computation the approach with 5 local areas outperforms the other approaches with smaller or larger numbers of areas.
Figure 13 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. 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 fact, a key element of the branch and bound algorithm for the coordinator problem is that \(GLB(\phi)\) 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.
Table 2 gives more information on the results obtained after 600 seconds of computation. The instances are grouped into three categories: root optimal (if the coordinator problem is solved at root), proven optimal and open. For the centralized approach, we consider only proven optimal and open instances since we did not find any root optimal solution. Column 1 reports the number of areas, Columns 2-3 the instance category and its percentage over the 40 delay instances, Columns 4-5 the lower bound at root and the starting upper bound (in sec), Columns 6-7 the final lower and upper bounds (in sec), Columns 8-9 the time to compute the best solution (in sec) and the total computation time (in sec).

Table 2: Performance of the algorithms after 600 seconds of computation.

<table>
<thead>
<tr>
<th>Num. Areas</th>
<th>Instance</th>
<th>Starting</th>
<th>Final</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Proven Opt</td>
<td>%</td>
<td>LB (s)</td>
<td>UB (s)</td>
</tr>
<tr>
<td>1</td>
<td>Open Inst</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Total</td>
<td>100</td>
<td>86.1</td>
<td>435.4</td>
</tr>
<tr>
<td>3</td>
<td>Root Opt</td>
<td>45</td>
<td>225.8</td>
<td>225.8</td>
</tr>
<tr>
<td>3</td>
<td>Proven Opt</td>
<td>33</td>
<td>254.5</td>
<td>538601</td>
</tr>
<tr>
<td>3</td>
<td>Open Inst</td>
<td>22</td>
<td>175.3</td>
<td>423.2</td>
</tr>
<tr>
<td>3</td>
<td>Total</td>
<td>100</td>
<td>223.8</td>
<td>175242</td>
</tr>
<tr>
<td>5</td>
<td>Root Opt</td>
<td>40</td>
<td>219</td>
<td>219</td>
</tr>
<tr>
<td>5</td>
<td>Proven Opt</td>
<td>54</td>
<td>275.5</td>
<td>392.5</td>
</tr>
<tr>
<td>5</td>
<td>Open Inst</td>
<td>5</td>
<td>220.5</td>
<td>333.5</td>
</tr>
<tr>
<td>5</td>
<td>Total</td>
<td>100</td>
<td>250.2</td>
<td>320.2</td>
</tr>
<tr>
<td>7</td>
<td>Root Opt</td>
<td>40</td>
<td>259.8</td>
<td>259.8</td>
</tr>
<tr>
<td>7</td>
<td>Proven Opt</td>
<td>55</td>
<td>250.8</td>
<td>50294</td>
</tr>
<tr>
<td>7</td>
<td>Open Inst</td>
<td>5</td>
<td>192</td>
<td>325</td>
</tr>
<tr>
<td>7</td>
<td>Total</td>
<td>100</td>
<td>251.9</td>
<td>25280</td>
</tr>
</tbody>
</table>

From the results of Table 2, the branch and bound algorithm for 5 and 7 areas improve the starting solution to the proven optimum in more than half the cases. It is also interesting to note that the starting lower bound increases with
the number of areas, since decreasing the size of local dispatcher problems increases the probability of solving dispatcher problems to proven optimum, thus yielding higher lower bounds in Equation (1). Also for the open instances, the distributed approaches present significantly smaller gaps than the centralized approach.

3.3. RESULTS FOR 60-MINUTE TRAFFIC PREDICTIONS

We now consider the larger scheduling cases in which the traffic predictions are computed over 60-minute time horizons. This second set of experiments is relevant to assess the limits of the branch and bound algorithm for the coordinator problem. For these cases, just the problem of finding a globally feasible solution is very challenging. Figure 14 shows the percentage of feasible solutions found within 3600 seconds of computation. Here, the most important remark is that the coordination task is the bottleneck of the procedure since the centralized approach is always able to compute a feasible solution. The coordinator experiences difficulty in driving the dispatchers towards globally feasible schedules. After 300 seconds of computation, only the cases with 3 areas and the centralized approach compute feasible schedules in all the cases.

![Figure 14: Percentage of feasible schedules found.](image)

Table 3: Performance of the algorithms after 3600 seconds of computation.

<table>
<thead>
<tr>
<th>Num. Areas</th>
<th>Starting</th>
<th>Final</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LB(s)</td>
<td>LB Feas(s)</td>
<td>UB Feas(s)</td>
</tr>
<tr>
<td>1</td>
<td>102</td>
<td>102</td>
<td>724</td>
</tr>
<tr>
<td>3</td>
<td>147</td>
<td>130</td>
<td>780</td>
</tr>
<tr>
<td>5</td>
<td>226</td>
<td>190</td>
<td>628</td>
</tr>
<tr>
<td>7</td>
<td>289</td>
<td>277</td>
<td>430</td>
</tr>
</tbody>
</table>

Table 3 reports the results obtained over the 36 cases that have been solved by all the approaches after 3600 seconds of computation. For these cases, the best solution was always found within at most 2900 seconds. Column 1 reports the number of areas, Column 2 the starting lower bound for all instances (in sec), Columns 3-4 the starting lower and upper bounds for the feasible solutions computed by the starting heuristic (in sec), Column 5 the percentage of feasible initial solutions, Columns 6-8 the final lower and upper bounds (in sec) and the percentage of optimal solutions computed by the branch and bound algorithms, Columns 9-10 the time to compute the best solution (in sec) and the total computation time (in sec).

An interesting conclusion is that the approaches with network division require coordination actions that are difficult to set but provide higher quality solutions. When the number of areas increases the optimality gap decreases
significantly. In fact, only the approaches with 5 and 7 areas are able to deliver some proven optimal solutions and to reduce the optimality gap from more than 600% (one area) to 62.5% (seven areas).

4. CONCLUSIONS AND FUTURE RESEARCH

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. However, in case of huge instances the distributed approaches may fail in finding a globally feasible solution within the time limit.

To the best of our knowledge, this paper is the first attempt to optimally solve the coordinator problem. Therefore, a number of questions remain that require further investigation. We observed that the network division is important to generate feasible and optimal solutions. However, there is no clear 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.

A number of other issues remain that need further development. Train priorities, inter-train connections and other operational constraints could be included in the model, to analyze their impact on traffic management. The model could be extended to different signaling technologies, such as the moving block technology. The solution approach described in this paper could also be used in the timetable design process, to compare the performance of different timetables or even the benefit of possible infrastructure investments.

Another interesting research direction is the extension of the alternative graph formulation to other transportation and production domains. The alternative graph has been shown to be effective to the minimization of the makespan at a steelmaking-continuous casting plant [23] and to the optimization real-time aircraft operations at a Terminal Maneuvering Area [10].

Acknowledgments

We thank the Dutch infrastructure manager ProRail for providing the instances. This work is partially supported by the Dutch research program TRANSUMO “Reliable Transport Chains” and by the Italian Ministry of Research, Grant number RBIP06BWZ8, project FIRB “Advanced tracking system in intermodal freight transportation”.

References


