Delft University of Technology

Delft Center for Systems and Control

Technical report 16-013

Distributed Model Predictive Control for Railway Traffic Management*

B. Kersbergen, T. van den Boom, and B. De Schutter

To cite this work, please refer to the published version:
B. Kersbergen, T. van den Boom, and B. De Schutter, "Distributed model predictive control for railway traffic management," *Transportation Research Part C*, vol. 68, pp. 462–489, July 2016. doi:10.1016/j.trc.2016.05.006

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://dspub.eu/16-013

Distributed Model Predictive Control for Railway Traffic Management

Bart Kersbergen, Ton van den Boom, and Bart De Schutter*

March 16, 2026

Abstract

Every day small delays occur in almost all railway networks. Such small delays are often called "disturbances" in literature. In order to deal with disturbances dispatchers reschedule and reroute trains, or break connections. We call this the railway management problem. In this paper we describe how the railway management problem can be solved using centralized model predictive control (MPC) and we propose several distributed model predictive control (DMPC) methods to solve the railway management problem for entire (national) railway networks. Furthermore, we propose an optimization method to determine a good partitioning of the network in an arbitrary number of sub-networks that is used for the DMPC methods. The DMPC methods are extensively tested in a case study using a model of the Dutch railway network and the trains of the Nederlandse Spoorwegen. From the case study it is clear that the DMPC methods can solve the railway traffic management problem, with the same reduction in delays, much faster than the centralized MPC method.

1 Introduction

In many countries in the world large, complex, and very busy railway networks have been built. Especially in North and West Europe, China, and Japan the railway networks are used near their maximum capacity. As a result, very little buffer time is available to recover from delays.

Every day small delays occur in almost all railway networks. Such small delays are often called "disturbances" in literature. In order to deal with disturbances dispatchers reschedule and reroute trains, or break connections. Currently most dispatchers take these decisions based on their experience, a given set of ground rules, and a limited overview of the network situation. To be able to handle disruptions trains may need to be canceled, or they may need to be rerouted through the entire network. These changes affect the rolling stock circulation and the personnel schedules. As a result, the rolling stock circulation needs to be recomputed and adjustments to the personnel schedules need to be made. Large perturbations, such as trains breaking down and tracks being blocked, causing trains to be canceled, are often called "disruptions" in literature. In this paper we focus on railway traffic management for disturbances, and therefore we will not consider the rolling stock circulation or the personnel schedules. For an overview of those research areas, literature on disruption management, and integrated approaches combining several of these research areas the reader is referred to the survey paper of Cacchiani et al. (2014). The literature on methods to railway traffic management for disturbances can, for the most part, be split up into two groups based on the size of the problem instances that are considered: there are approaches that focus on a small part of the railway network and there are approaches that take the entire (national) network into account. We will call railway

*All authors are with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands, e-mail: {b.kersbergen, a.j.j.vandenboom, b.deschutter}@tudelft.nl

management for small parts of the railway network "local control" and railway management for the entire network "global control". Our work in the paper will focus on global control.

Most of the research on railway traffic management is on local control. There we will first look at the literature on this subject. In recent years many approaches for local railway traffic management have been proposed such as the methods of Caimi et al. (2012), Corman et al. (2010a, 2012b), D'Ariano and Pranzo (2009), D'Ariano et al. (2007), Meng and Zhou (2014), Murali et al. (2015), Pellegrini et al. (2014), Quaglietta et al. (2013) and Rodriguez (2007).

Caimi et al. (2012) developed a railway traffic management method that tries to schedule and route all trains in an area in and around a large station. A model predictive control approach with a microscopic model of the railway operations is used based on blocking times. At each point where rescheduling of a train is possible, a set of possible blocking times for different routes and departure times at platforms or arrival times at the boundaries of the area are considered for that train. As many trains as possible are then assigned a route with corresponding blocking times such that all safety and operational constraints are still respected. The objective is to optimize the passenger satisfaction, measured by punctuality and reliability. The resulting optimization problem is a binary linear programming problem.

In the work of Pellegrini et al. (2014) the authors propose a mixed integer linear programming (MILP) approach to reschedule and reroute trains. They model the infrastructure with as many details possible. They even model the track circuits a block section is build up from separately so they can model sectional release in the railway traffic management and compare it to block or route release. Optimizations can be performed subsequently over a long time horizon, and the decisions resulting from each optimization are implemented as they are produced, resulting in a so-called receding or rolling horizon. They propose two objective functions for the MILP approach, namely minimizing the maximum secondary delay and minimizing the total (secondary) delay. They test their approach for two case studies: the triangle of Gagny and the Lille-Flandres station for different prediction horizon lengths. The first case study is used to determine the difference in delay reduction for the sectional and route release approaches. The second case study is used to test the real-life applicability of the approach. They try to determine the optimal solution for the various delay scenarios and horizon lengths. In many of the cases it proves to be quite difficult to determine the optimal solution, but in most cases good solutions can be found within 3 minutes. Our work differs from this work in several ways. The goal here is to reduce the delays in small areas such as stations, whereas we aim to reduce the delays on a network wide scale. As a result we do not consider rerouting, we consider station areas a black boxes where we know the arrival and departure times, but not the routes, or the layout of the station.

Meng and Zhou (2014) developed a method for the simultaneous rerouting and rescheduling of railway traffic for an N-track network. The network is described per block section and in each block section only one train can be present at the same time. They propose a novel Lagrangian relaxation solution framework to decompose the problem into easier to solve dual subproblems and then further decompose the problem into a set of single train routing and scheduling problems that can be solved in sequence. The solutions of the dual problems can be transformed into feasible solutions for the original problem.

Murali et al. (2015) present a macroscopic modeling strategy for medium sized railway networks for freight trains. They also propose a method to reduce the complexity of the model by aggregating portions of the network into single nodes. They formulate the routing and scheduling problem as an integer programming (IP) problem and propose two methods to find (partial feasible) solutions to the IP-problem called PFSLPR and PFSRC. For PFSLPR the IP-problem is solved using LP-relaxation. The solution of the LP-relaxation is rounded of to find an integer solution giving the routes and the departure times at the origin stations. With the PFSRC approach the possible routes are limited by adding constraints that enforce a periodic route schedule. The resulting problem is a mixed integer linear programming-problem. Solving it gives the routes for the trains and with the routes the maximal matching problem can be solved to find the departure times. The authors use a genetic algorithm to improve the solutions found with the two approaches. The computation performance and quality of the solutions of the approaches is compared for a set of networks with a different number of trains. Finally the original model is compared to the aggregated model for

different levels of aggregation.

Rodriguez (2007) proposes a method for the routing and scheduling of trains through an area around the Pierrefitte-Gonesse junction north of Paris. Rodriguez uses a microscopic simulation to model the train and driver behavior and describes the routing and scheduling problem as a constraint programming problem. For the given case study the reduction of the delays ranges from 63 to 96%.

In the work of D'Ariano and Pranzo (2009), D'Ariano et al. (2007) the railway operation is also modeled as a microscopic model based on blocking times with an Alternative Graph (AG) approach. In their AG approach for every train occupying a block section a node is created in a graph. The nodes of a single train are then connected to each other through running time constraints and for every pair of trains occupying the same block section headway/separation constraints are added. If the order in which the trains can occupy the block sections can be changed with rescheduling actions, then a pair of alternative arcs defining the two orders in which the trains can occupy the block section are added to the graph. A new schedule for the railway traffic is found when for each pair of alternative arcs only one arc is chosen and no circuits of positive length are present in the graph. The graph has an extra node to which all nodes are connected and the weights of the arcs from all nodes to this extra node are chosen such that minimizing the maximum weight of all paths from the starting to the ending node corresponds to minimizing the maximum consecutive delay. To solve this problem the authors use their own branch and bound algorithm.

Corman et al. (2010a, 2012b) have extended the work of D'Ariano et al. (2007) to also consider breaking connections and locally rerouting trains. They consider the effects of breaking connections by determining the maximum consecutive delay for different sets of maintained connections. The decrease of the maximum consecutive delay is then weighed against the number of broken connections. For the local rerouting they consider a fixed set of alternative routes through station areas for the trains and determine the optimal routes, train orders, and event times of the trains using the branch and bound algorithm of D'Ariano and Pranzo (2009), D'Ariano et al. (2007).

Quaglietta et al. (2013) propose a framework to evaluate the stability of the solutions found by the railway traffic management of D'Ariano and Pranzo (2009), D'Ariano et al. (2007) in the case of dynamically changing delays in the network. To do this they develop a closed-loop framework where the railway traffic management algorithm is coupled to a microscopic simulation tool called EGTRAIN(Quaglietta, 2011). In the framework the optimizations are performed consecutively with a fixed time window in between them, and the new schedules resulting from each optimization are implemented after each optimization, resulting in receding or rolling horizon. They test this for the corridor Utrecht-Den Bosch of the Dutch railway network. From their case study they show that larger prediction horizons result in more delay reduction but also result in less stable solutions: there are more changes in the schedule between two consecutive optimizations. For their case study increasing the prediction horizon past 30 minutes does not yield a higher delay reduction.

In most cases the methods for local control use microscopic models. By using microscopic models they can model the routes, the signaling and safety system, and the speed profiles of the trains. In some cases they can even change the routes and adjust the speed profiles of the trains. Because of the use of microscopic models they must consider relatively small networks to be able to find solutions quickly. Therefore these methods are not directly applicable for global railway traffic management and we should look at different methods.

Several researchers extended their work to, or developed new methods for larger networks, such as Corman et al. (2010b, 2012a, 2014), Kanai et al. (2011), Kecman et al. (2013), Kersbergen et al. (2014a), Törnquist (2007), Törnquist and Persson (2007), and Törnquist-Krasemann (2012).

Corman et al. (2010b, 2012a) and Corman et al. (2014) extend their previous work to multiple areas using a supervisory controller that coordinates between the areas. For smaller instances and a limited number of areas the proposed bi-level approach works well, but once the prediction horizon becomes larger and the number of areas increases a good feasible solution can not always be found. In this paper we propose a flat distributed approach (single level), where the distributed controllers coordinate with each other directly and

ensure feasibility without the need of a supervisory controller. As we mentioned before another difference is that we model the network at a macroscopic level, whereas the models used in Corman et al. (2010b, 2012a) and Corman et al. (2014) use a microscopic model. Furthermore we use a general MILP approach and not a AG approach with a custom-made branch and bound algorithm optimized for minimizing the maximum consecutive delay.

Törnquist and Persson (2007) propose a railway traffic management method for large network instances, covering a part of the Swedish railway network in the south of the country. Specifically they consider a network with multiple parallel tracks that can be used between stations. They assume that there are no restrictions on the tracks the trains can use. They propose a mixed integer programming (MIP) problem to schedule the trains and to divide them over the tracks such that the delays are minimized. For some instances the MIP solver cannot find a feasible solution within the available time. To solve this problem Törnquist(Törnquist, 2007) proposes a heuristic method to solve the problems. She implements a receding horizon to be able to deal with long time horizons such as an entire day. She considers three different cost functions: Minimize total final delay, minimize total accumulated delay, and minimize total delay cost. With the heuristic approach most scenarios can be solved very quickly, but some still take a very long time (some more than 1 hour). Törnquist-Krasemann (2012) extends the model and solution procedures to also consider the available routes through the stations limiting the number of available tracks for each train.

Kecman et al. (2013) use the same AG approach as Corman et al. (2010b, 2012a,b, 2014), D'Ariano and Pranzo (2009), D'Ariano et al. (2007) but now for macroscopic models with different levels of details. The AG is specifically designed such that the maximum consecutive delay can be minimized using the branch and bound algorithm of D'Ariano et al. (2007). This branch and bound algorithm is specifically designed to minimize the weight of the longest path and the AG is adjusted such that the weight of the longest path corresponds to the maximum consecutive delays.

Kanai et al. (2011) propose a tabu search method to determine the connections of the trains for the entire network. The goal is to minimize the passenger disutility and to achieve this goal various linear and non-linear objective functions are proposed. By simulating the railway traffic and the passenger behavior they are able to determine the passenger disutility for a given set of connections. They test their tabu search on a case study consisting of a part of the Japanese railway consisting of 41 stations and 40 trains. For this test case the process time of the algorithm was about 6 minutes in all scenarios using a PC with an Intel Core2Duo CPU.

Kersbergen et al. (2014a) propose a macroscopic model of the railway traffic that can be described as a switching max-plus-linear system. Due to the macroscopic model they are able to model the entire Dutch Network, although in that paper they do not consider all trains. They write the online railway traffic management problem as a MILP problem and solve it using various MILP solvers.

The methods in these papers all consider larger networks, but most still do not consider a complete (national) railway network, or only consider a subset of all the trains in the network. Even for the railway networks they currently consider computing a good solution to the railway traffic management problem in a short time remains a challenge. Besides these papers many more papers have been published on the subject of local and in some cases global railway traffic management and several review papers on this subject have recently been published: see e.g. Cacchiani et al. (2014), Corman and Meng (2015), and Fang et al. (2015). The authors of these review papers conclude that much research has been done in the recent years on railway traffic management, but most of it has been on small networks or parts of a network. There is a need for railway traffic management methods for large-scale networks and possible directions are the use of macroscopic models and multi-level or distributed control.

We will therefore propose our own control methods to solve the global railway traffic management problem in this paper. We assume that at a local level another controller is active determines the routes and possibly arrival and departure times at local points in the network to ensure the feasibility of the global solution, or in case this is not possible sends feedback to the global controller such that the global problem is updated and can be solved again. To solve the global railway traffic management problem we will look at model predictive

control (MPC) approaches. MPC is a control methodology that, at discrete time instants, determines the control inputs for the system that minimize a cost function based on a prediction of the evolution of the state of the system under control. The prediction is done over a predefined prediction window using a model of the system under control. In almost all of the papers on railway traffic management some form of model predictive control (MPC) is used.

MPC can be characterized by five components:

1) A model of the system (and of the disturbances).
   The model is used to predict the effects of the control inputs on the evolution of the system over a given prediction window and to determine the control actions that minimize or maximize a given performance index. For railway traffic management the model is used to predict the future arrival and departure times based on the dispatching actions, current delays, infrastructure, the train dynamics, and in some cases the routes of trains through the stations. In most cases no model is used to predict the future delays; instead, only the known delays are used.

2) A cost function.
   The cost function is a function of the predicted state evolution, a reference signal and the control inputs. It is used as a measure to evaluate the quality of the control actions and is chosen by the designer of the controller. For railway traffic management the cost function is usually a measure of the delays in the network, such as the maximum train/passenger delay or the sum of (passenger) delays.

3) Constraints.
   One of the differentiating characteristics of MPC is that it can directly handle constraints. These constraints can be on the input, the output, or the state of the system, e.g. to limit the rate of change in the output, or to limit the range or shape of the input. Examples of operational constraints are: a maximum on the arrival time of a train at a certain station, or a minimum velocity (or maximum running time) that should be maintained.

4) Optimization.
   At each time instant the controller receives information about the system, usually the state or the output, and uses that information and a model of the system under control to predict the effects of the control inputs on the evolution of the state (or output) of the system. The evolution of the state is only predicted over a finite window determined by the prediction horizon. Furthermore the control inputs are also only determined over a finite window determined by the control horizon. For online railway traffic management the optimization determines the dispatching actions that reduce the delays in the future (arrival and) departure times of the trains based on the current situation in the network by determining the effects of the current delays and the dispatching actions on those arrival and departure times.

5) A receding horizon.
   Once the controller has determined the control input(s) at each time instant it only implements the first part of the control inputs. It then waits till the next time instant, receives updated information from the system and recomputes the control inputs for the optimization problem at that time instant. At each time instant the optimization is based on a prediction done over a finite window and since this window starts at the current time instant, it shifts at every time instant. It is called a receding horizon because of the moving window and the implementation of only the first parts of the control inputs.

MPC for linear models was first described and successfully applied in the late 1970s and early 1980s by Richalet et al. (1978) and Cutler and Ramaker (1980) in the chemical process industry. In the next decades MPC was applied successfully many more times and extended to include non-linear models (see García et al. (1989), Morari and Lee (1999), Qin and Badgwell (2003) and the references therein). In the early 2000s a further extension of MPC to max-plus-linear discrete-event systems was made by De Schutter and van den Boom (2001, 2003).

The aim of this paper is to propose a method that can be used for global railway traffic management. The main challenge is the size of the resulting optimization problem and the time it requires to be solved. Most of the literature on railway traffic management only considers local control and for those papers that consider larger networks most still do no consider a complete (national) network, but only large parts of it. Even for those partial networks the computation time remains a challenge. Therefore we propose to look at distributed model predictive control (DMPC) (Camponogara et al., 2002, Dunbar, 2007, Farina and Scattolini, 2012, Richards and How, 2007) to reduce the computation time. With DMPC there is no supervisory controller: instead the model predictive controllers solving the dispatching problems for the subnetworks communicate and coordinate with each other to reach a global feasible solution in a shorter time than the global model predictive controller. For an overview of DMPC methods the interested reader is referred to Maestre and Negenborn (2013).

To improve the work of Kersbergen et al. (2014b, 2015) we introduce a new method based on mixed integer quadratic programming (MIQP) to determine the partitions of the railway network. The network partitioning is directly used to determine the parts of the network that each of the distributed model predictive controllers optimize. A good partition of the railway network should results in lower computation times of the DMPC approaches and a better result. Therefore it is important to determine a good partition of the railway network. Furthermore, we determine partitions of the network for 2, 3, 4, 6, and 8 parts to find the best number of parts the network should be split into. We test all of these partitions in an extensive case study where the controllers are tested in closed-loop simulation (with the receding horizon principle implemented) with delays randomly added to the simulation model.

This paper is organized as follows. In Section 2 we explain how MPC is applied to solve the railway traffic management problem. In Section 4 we explain the distributed model predictive control approaches and in Section 3 we describe the method that we use to determine the partitioning of the network. In Section 5 we give an extensive case study on the performance of the different approaches to railway traffic management considered in this paper. We draw conclusions and give recommendations for future work in Section 5.

2 MPC for on-line railway traffic management

The components of MPC for the application of on-line railway traffic management will be explained in detail in the following subsections. First we will describe the model used in the model predictive controller. This model can be used to predict the future arrival and departure times of the trains.

2.1 Modeling of railway traffic and network

For the model used in the model predictive controller we describe the railway traffic and network as a discrete-event system. The events of the system are the arrivals and departures of the trains at stations and junctions and crossings outside the station areas. The resulting model is a macroscopic model where the stations are modeled as single nodes with sufficient capacity to accommodate all trains that arrive at the stations. The station layout and routes through the station are therefore not modeled. The tracks between stations are modeled as single links. In this paper we will only describe those elements of the model that are necessary to understand how the railway traffic is modeled. For a complete description of the model the reader is referred to Kersbergen et al. (2014a).

The events (arrivals and departures) are connected to each other through several constraints that model the signaling system, the movement of the trains through the network, and operational conditions such as scheduled departure and arrival times and possible transfers between trains. We define a train run as a train departing from one station, traversing a track, and arriving at the next station. Each train run has its own index and the indices of the associated arrival and departure events correspond to the index of the train run,

then for train run i the departure time is given by d_i and the arrival time is given by a_i. There are six types of constraints connecting the events of the train run to other events:

- Running time constraints
  A *running time constraint* describes the traversal of a train over the track and relates the arrival time of a train run to its departure time:

$$a_i \geq d_i + \tau_{\mathrm{r},i}, \tag{1}$$

  where $\tau_{\mathrm{r},i}$ is the minimum required time for the train of train run i to traverse the track.

- Continuity constraints
  A *continuity constraint* connects two trains of the same line to each other, e.g. a 'physical' train driving from one station to the next and then continuing on to a third station. This also includes trains turning or changing lines at their end station:

$$d_i \geq a_{p_i} + \tau_{\mathrm{d},i}, \tag{2}$$

  where p_i and i are two consecutive train runs of the same 'physical' train, $\tau_{\mathrm{d},i}$ is the minimum time the train remains at the station, junction, or crossing.

- Headway constraints
  Headway constraints define the order in which trains traverse tracks and they indirectly define the minimum distance between trains. This is done by relating the arrival and departure times of one train to the arrival and departure times of the other trains traversing the same track. The headway times can be chosen such that, as long as there are no unexpected delays on the tracks, none of the trains run into a yellow or red signal and have to break, or they may be based on regulations set by the industry.

  For two 'physical' trains with train runs i and j respectively traversing the track in the same direction we have

$$d_i \geq d_j + \tau_{\mathrm{h,d},i,j} \tag{3}$$
$$a_i \geq a_j + \tau_{\mathrm{h,a},i,j}, \tag{4}$$

  where $\tau_{\mathrm{h,d},i,j}$ is the minimum safe headway time between departure times d_j and d_i, and $\tau_{\mathrm{h,a},i,j}$ is the minimum safe headway time between arrival times a_j and a_i.

  In case the order of departures of two trains on the same track can be changed at the station the headway constraints need to be adjusted to reflect that possibility. This can be done by replacing the previous headway constraints with:

$$d_i \geq d_j + \tau_{\mathrm{h,d},i,j} + u_{\mathrm{h},i,j}\beta \tag{5}$$
$$a_i \geq a_j + \tau_{\mathrm{h,a},i,j} + u_{\mathrm{h},i,j}\beta \tag{6}$$
$$d_j \geq d_i + \tau_{\mathrm{h,d},j,i} + (1 - u_{\mathrm{h},i,j})\beta \tag{7}$$
$$a_j \geq a_i + \tau_{\mathrm{h,a},j,i} + (1 - u_{\mathrm{h},i,j})\beta, \tag{8}$$

  where $u_{\mathrm{h},i,j} \in \{0, 1\}$ is a binary variable and $\beta \ll 0$ is a large negative number. Notice that by setting the binary variable to 0 a large negative value is added to the last two constraints rendering them ineffective (for all (relevant) values of the continuous variables the constraints are satisfied). By setting the binary variable to 1 the first two constraints are rendered ineffective and the last two constraints determine the order of the trains.

  If two 'physical' trains with train runs i and l respectively traverse the same track in the opposite direction the headway or separation constraint becomes:

$$d_i \geq a_l + \tau_{\mathrm{s},i,l}, \tag{9}$$

where $\tau_{\mathrm{s},i,l}$ is the minimum safe separation time between the arrival of the train of train run l and the departure of the train of train run i.

When it is possible to change the order of the trains traversing the track in opposite direction this headway constraint should be replaced by:

$$d_i \geq a_l + \tau_{\mathrm{s},i,l} + u_{\mathrm{s},i,l}\beta \tag{10}$$
$$d_l \geq a_i + \tau_{\mathrm{s},l,i} + (1 - u_{\mathrm{s},i,l})\beta, \tag{11}$$

where $u_{\mathrm{s},i,l} \in \{0, 1\}$ is a binary variable and $\beta \ll 0$ is a large negative number.

- Timetable constraints
  Since the passenger railways operate according to a timetable, none of the trains are allowed to depart before their scheduled departure times and in some cases they may not arrive before their scheduled arrival times either. This requirement can be modeled by adding *timetable* constraints:

$$d_i \geq r_{\mathrm{d},i} \tag{12}$$
$$a_i \geq r_{\mathrm{a},i}, \tag{13}$$

  where $r_{\mathrm{d},i}$ and $r_{\mathrm{a},i}$ are the scheduled departure and arrival time respectively for train run i. Timetable constraints for the arrival times may not always be present.

- Connection constraints
  At some stations passengers may transfer to another train. Transfers that are guaranteed, are modeled by *connection* constraints. Connection constraints ensure that passengers can change trains at stations by defining a relation between the departure time of one train and the arrival time of the train from which the passengers transfer. In case a transfer from the train of train run m to the train of train run i is guaranteed a connection constraint must be defined that ensures this transfer:

$$d_i \geq a_m + \tau_{\mathrm{c},i,m}, \tag{14}$$

  where $\tau_{\mathrm{c},i,m}$ is the time the passengers have to transfer from the train of train run m to the train of train run i.

  If connection constraints can be broken this can be modeled by

$$d_i \geq a_m + \tau_{\mathrm{c},i,m} + u_{\mathrm{c},i,m}\beta, \tag{15}$$

  where $u_{\mathrm{c},i,m} \in \{\mathrm{c}, 0, 1\}$ is a binary variable and $\beta \ll 0$ is a large negative number.

- Coupling constraints:
  At some stations two 'physical' trains arrive and must be coupled such that they can continue as one 'physical' train. We do not model the coupled train as one 'physical' train, but as two 'physical' trains that arrive and depart at the same time. To ensure that they arrive and depart at the same time we define *coupling* constraints. Consider the train of train run i and let the train of train run o be the train to which train of train run i should be coupled to. For these trains the coupling constraints are:

$$d_i \geq d_o \tag{16}$$
$$d_o \geq d_i \tag{17}$$
$$a_i \geq a_o \tag{18}$$
$$a_o \geq a_i, \tag{19}$$

Let a and d contain all arrival and departure times respectively and let $x = \begin{bmatrix} d^\top & a^\top \end{bmatrix}^\top$ be the vector containing all event times.

Then all of these constraints can be described by one of the following general forms:

$$x_i \geq \eta_i \tag{20}$$
$$x_i \geq x_j + \eta_{i,j} \tag{21}$$
$$x_i \geq x_j + \eta_{i,j} + u_{i,j}\beta \tag{22}$$
$$x_i \geq x_j + \eta_{i,j} + (1 - u_{i,j})\beta, \tag{23}$$

where $u_{i,j}$ is one of the three different types of binary variables $u_{\mathrm{h},i,j}$, $u_{\mathrm{s},i,j}$, or $u_{\mathrm{c},i,j}$, and $\eta_i, \eta_{i,j} \in \mathbb{R}$ are constant values, that are either a process times, planned arrival or departure times, or just zero.

We assume that the trains arrive and depart as soon as all constraints are satisfied. With this model we can predict the future arrival and departure times and the effects of the binary variables on these times for all trains over a whole day. For the model predictive controller at each time instant $t(\kappa)$ we limit this to a given prediction and control horizon. The prediction and control horizon at time instant $t(\kappa)$ are explained next.

For the model predictive controller the model of the railway traffic is built up for a very large time window, much larger than the prediction horizon, and at each time instant only the part of the model needed for the optimization at that time is used. Let x then be the vector containing all event times for the entire time window and let u be the vector containing all binary variables of the model.

2.2 Prediction and control horizon

Let us denote the time instants by $t(\kappa)$, for $\kappa = 0, 1, \ldots$. The time between time instants $t(\kappa)$ can vary or be constant. We will use a fixed time step between time instants resulting in the following equation: $t(\kappa + 1) = t(\kappa) + \tau_{\mathrm{S}}$, where τ_{S} is the sampling time. To be able to describe the control problem at time instant $t(\kappa)$ we first need to define the prediction and control horizon. After that we explain how the events and control inputs that are in the prediction and control window can be determined for each time instant $t(\kappa)$ and how that information is used to determine the part of the model that is needed for the optimization at $t(\kappa)$.

The prediction horizon determines the length of the prediction window and therefore the events for which the event time must be determined. The prediction horizon determines the number of time steps in the prediction window and is denoted by p. The prediction window is then defined to be the time window $(t(\kappa), t(\kappa) + p\tau_{\mathrm{S}}]$.

The control horizon determines the length of the control window and therefore for which events the binary variables can be adjusted. The control horizon determines the number of time steps in the control window and is denoted by c, with $c \leq p$. The control window is then defined to be the time window $(t(\kappa), t(\kappa) + c\tau_{\mathrm{S}}]$.

2.2.1 Events and control variables in the prediction and control horizon

Let $\mathcal{X}(\kappa)$ be a vector containing all event time variables of x for which the event times are scheduled or expected to occur in the prediction window at time instant $t(\kappa)$. These events have not yet occurred and are predicted to occur within the time interval $(t(\kappa), t(\kappa) + p\tau_{\mathrm{S}}]$.

Not all binary variables can be changed at each time instant. To determine the control variables that can be changed at $t(\kappa)$ consider the constraints in (1)–(19). Each constraint relates two events to each other. Let $\mathcal{U}(\kappa)$ be a vector containing all binary variables that can be changed by the optimization problem at time instant $t(\kappa)$. A control variable is in $\mathcal{U}(\kappa)$ if the value of the right-hand side of a constraint depends on that control variable and if at least one of the two events of the constraint is in the control window and the other is in the control or prediction window.

All events, and their associated control variables that occurred before $t(\kappa)$ are assumed to be in the past and their event times are assumed to be known. The control variables that are only associated to the events that are predicted at $t(\kappa)$ to occur within the time interval $(t(\kappa), t(\kappa) + c\tau_{\mathrm{S}}]$ are set to their nominal value, which is zero[1].

2.2.2 Modeling constraints

The model of the railway traffic is built up for an entire day. For the optimization at time instant $t(\kappa)$ only those constraints that are needed to determine the event times in $\mathcal{X}(\kappa)$ have to be considered. The constraints for which the left-hand side contains an event time in $\mathcal{X}(\kappa)$, and the right-hand side contains an event time in $\mathcal{X}(\kappa)$ or an event time that is in the past are the constraints that are needed at time instant $t(\kappa)$.

The constraints can be rewritten in matrix form by moving the event times and binary variables to the left-hand side and the process times and constants to the right-hand side.

For example the constraints (5)–(8) can be rewritten to:

$$\begin{aligned} d_i - d_j - u_{\mathrm{h},i,j}\beta &\geq \tau_{\mathrm{h,d},i,j} \\ a_i - a_j - u_{\mathrm{h},i,j}\beta &\geq \tau_{\mathrm{h,a},i,j} \\ d_j - d_i + u_{\mathrm{h},i,j}\beta &\geq \tau_{\mathrm{h,d},j,i} + \beta \\ a_j - a_i + u_{\mathrm{h},i,j}\beta &\geq \tau_{\mathrm{h,a},j,i} + \beta, \end{aligned}$$

and are given in matrix form by:

$$\begin{bmatrix} -1 & 1 & 0 & 0 & \beta \\ 0 & 0 & -1 & 1 & \beta \\ 1 & -1 & 0 & 0 & -\beta \\ 0 & 0 & 1 & -1 & -\beta \end{bmatrix} \begin{bmatrix} d_i \\ d_j \\ a_i \\ a_j \\ u_{\mathrm{h},i,j} \end{bmatrix} \leq \begin{bmatrix} -\tau_{\mathrm{h,d},i,j} \\ -\tau_{\mathrm{h,a},i,j} \\ -(\tau_{\mathrm{h,d},j,i} + \beta) \\ -(\tau_{\mathrm{h,a},j,i} + \beta) \end{bmatrix}.$$

2.3 Cost function

Before the possible cost functions are proposed, first the definition for delay should be given:

Definition 1. *Delay*
For any event x_i with a scheduled event time r_i the delay is defined as the deviation from its scheduled event time r_i:

$$x_i^d = x_i - r_i, \tag{24}$$

where x_i^d is the delay of event x_i.

Since all of these events have a scheduled event time, they also have a timetable constraint, therefore $x_i \geq r_i$ and $x_i^{\mathrm{d}}(k) \geq 0$. Define $\iota(\kappa)$ as the vector containing the scheduled arrival or departure times of the events in the prediction horizon at $t(\kappa)$. Define $\mathcal{X}^{\mathrm{d}}(\kappa)$ as the delays of the events in $\mathcal{X}(\kappa)$. Finally define $\mathbf{1}^{1\times m}$ as a 1 by m vector containing only ones and $\mathbf{0}^{1\times m}$ as a 1 by m vector containing only zeroes. With these definition the cost function can be defined.

For railway traffic management the goal is almost always to minimize the delays; so it makes sense that the performance criterion reflects this. Obvious choices for the performance criterion would be the minimization

[1]When the control variables are zero the trains run over the tracks in the nominal order, all connections are maintained, and all trains run on the tracks they were originally planned on; only the arrival and departure times are adjusted to avoid conflicts.

of the sum of all delays, or the sum of arrival delays, minimization of the maximum delay, or minimizing the passenger delays if detailed passenger information is available. The sum of all delays would translate to the following cost function of the optimization:

$$\begin{aligned} J(\kappa) &= \mathbf{1}^{1\times n_{\mathcal{X}}(\kappa)} \cdot \mathcal{X}^{\mathrm{d}}(\kappa) + \varrho \mathbf{1}^{1\times n_{\mathcal{U}}(\kappa)} \cdot \mathcal{U}(\kappa) \\ &= \mathbf{1}^{1\times n_{\mathcal{X}}(\kappa)} \cdot (\mathcal{X}(\kappa) - \iota(\kappa)) + \varrho \mathbf{1}^{1\times n_{\mathcal{U}}(\kappa)} \cdot \mathcal{U}(\kappa) \\ &= \mathbf{1}^{1\times n_{\mathcal{X}}(\kappa)} \cdot \mathcal{X}(\kappa) + \varrho \mathbf{1}^{1\times n_{\mathcal{U}}(\kappa)} \cdot \mathcal{U}(\kappa) - \mathbf{1}^{1\times n_{\mathcal{X}}(\kappa)} \cdot \iota(\kappa), \end{aligned}$$

where ρ is a small positive value used to ensure that the minimum number of changes are made to the schedule when multiple solution result in the same value for the sum of delays, $n_{\mathcal{X}}(\kappa)$ is the number of event times in $\mathcal{X}(\kappa)$, and $n_{\mathcal{U}}$ is the number of binary variables in $\mathcal{U}(\kappa)$. The preference for solutions with minimal changes compared to the schedule is based on the idea that at some point in the future all trains should run according to the schedule again and that means that any change made to the schedule must also be reversed at some point, therefore we prefer solutions with less changes if the delays are the same for the solutions. Note that for a given time step κ, $\mathbf{1}^{1\times n_{\mathcal{X}}(\kappa)} \cdot \iota(\kappa)$ is a constant value and as the goal is to minimize the cost function, this constant term does not influence the solution and can thus be removed from the cost function. This means that minimizing the sum of delays is the same as minimizing the sum of event times.

For the sum of arrival delays this results in the cost function:

$$J(\kappa) = [\mathbf{1}^{1\times n_a(\kappa)}\ \mathbf{0}^{1\times n_d(\kappa)}] \cdot \begin{bmatrix} \mathcal{X}_{\mathrm{a}}(\kappa) \\ \mathcal{X}_{\mathrm{d}}(\kappa) \end{bmatrix} + \varrho \mathbf{1}^{1\times n_{\mathcal{U}}(\kappa)} \cdot \mathcal{U}(\kappa),$$

where $\mathbf{0}^{1\times n_d(\kappa)}$ is a vector containing only zeros of length $n_{\mathrm{d}}(\kappa)$, $n_{\mathrm{d}}(\kappa)$ is the number of departure times in $\mathcal{X}(\kappa)$, $n_{\mathrm{a}}(\kappa)$ is the number of arrival times in $\mathcal{X}(\kappa)$, $\mathcal{X}_{\mathrm{a}}(\kappa)$ is the vector containing the arrival times, and $\mathcal{X}_{\mathrm{d}}(\kappa)$ is the vector containing the departure times. For the sum of departure delays the zero and one vector in the cost function should be switched.

Another possibility is to minimize the maximum (arrival/departure) delay. For this a new continuous variable $\mathcal{X}_{\max}$ needs to be added to the optimization problem that is the maximum of all (arrival/departure) delays. This can be achieved by adding a set of constraints for the maximum of all delays:

$$\mathcal{X}_{\max} \geq \mathcal{X}_i(\kappa) - \iota_i(\kappa) \quad \text{for all } \mathcal{X}_i(\kappa) \in \mathcal{X}(\kappa).$$

For the arrival delays the set of constraints would be

$$\mathcal{X}_{\max} \geq \mathcal{X}_{\mathrm{a},i}(\kappa) - \iota_{\mathrm{a},i}(\kappa) \quad \text{for all } \mathcal{X}_{\mathrm{a},i}(\kappa) \in \mathcal{X}(\kappa).$$

For the departure delays the set of constraints would be:

$$\mathcal{X}_{\max} \geq \mathcal{X}_{\mathrm{d},i}(\kappa) - \iota_{\mathrm{a},i}(\kappa) \quad \text{for all } \mathcal{X}_{\mathrm{d},i}(\kappa) \in \mathcal{X}(\kappa).$$

The cost function would then be:

$$J(\kappa) = \mathcal{X}_{\max} + \varrho \mathbf{1}^{1\times n_{\mathcal{U}}(\kappa)} \cdot \mathcal{U}(\kappa).$$

2.4 Optimization problem

With the events, control variables, and constraints that are required for the optimization at time instant $t(\kappa)$ determined, and with the cost function defined, the optimization problem can be written as a mixed integer linear programming (MILP) problem:

$$\min_{z(\kappa)} c^{\top}(\kappa)\, z(\kappa) \tag{25}$$

$$\text{s.t. } A(\kappa)\, z(\kappa) \leq b(\kappa), \tag{26}$$

where

$$z(\kappa) = \begin{bmatrix} \mathcal{X}(\kappa) \\ \mathcal{U}(\kappa) \end{bmatrix}.$$

Equation (25) contains the cost function that needs to be minimized, where $c(\kappa)$ is a weighting vector, and (26) contains the mixed-integer-linear constraints in matrix form.

Now that the optimization problem at time instant $t(\kappa)$ of the MPC approach has been defined we can look at distributed model predictive control approaches.

3 Model-based partitioning

For the DMPC methods we will present in the next section we assume that the constraint matrix of the centralized MPC problem can be reordered such that the matrix can be structured in the following way:

$$\begin{aligned} &\min_{z(\kappa)} \begin{bmatrix} \tilde{c}_1^\top(\kappa) & \tilde{c}_2^\top(\kappa) & \dots & \tilde{c}_{n_{\text{sub}}}^\top(\kappa) \end{bmatrix} \begin{bmatrix} \tilde{z}_1^\top(\kappa) & \tilde{z}_2^\top(\kappa) & \dots & \tilde{z}_{n_{\text{sub}}}^\top(\kappa) \end{bmatrix}^\top \\ &\text{s.t.} \begin{bmatrix} A_{1,1}(\kappa) & A_{1,2}(\kappa) & \dots & A_{1,n_{\text{sub}}}(\kappa) \\ A_{2,1}(\kappa) & \ddots & & A_{2,n_{\text{sub}}}(\kappa) \\ \vdots & & \ddots & \vdots \\ A_{n_{\text{sub}},1}(\kappa) & A_{n_{\text{sub}},2}(\kappa) & \dots & A_{n_{\text{sub}},n_{\text{sub}}}(\kappa) \end{bmatrix} \begin{bmatrix} \tilde{z}_1(\kappa) \\ \tilde{z}_2(\kappa) \\ \vdots \\ \tilde{z}_{n_{\text{sub}}}(\kappa) \end{bmatrix} \leq \begin{bmatrix} \tilde{b}_1(\kappa) \\ \tilde{b}_2(\kappa) \\ \vdots \\ \tilde{b}_{n_{\text{sub}}}(\kappa) \end{bmatrix}. \end{aligned}$$

This structure can then be used to define the subproblems of the DMPC problems as we will show in the next section. In this section we present a mixed integer quadratic (MIQP) optimization problem that can determine this reordering or partitioning of the constraint matrix for a given number of desired parts. This partitioning is done off-line based on the constraints matrix of the centralized MPC problem. In the proposed approach the off-line partitioning method is described next. We first reorder the rows and columns of the constraint matrix A in (25) and reorder the variables in vector z such that the structure of A gets as close as possible to a block-diagonal structure. The number of blocks depends on the choice of the number of subsystems which is denoted by n_{sub} and should be chosen by the designer before the start of the reordering.

This reordering is based on the following goals:

- The constraints that cannot be placed in the block-diagonal structure should only depend on continuous variables.
- Each diagonal block has its own unique set of binary and continuous variables it depends on, there is no overlap between those sets, and the union of the sets contains all the binary and continuous variables.
- The number of constraints that cannot be placed in the block-diagonal structure should be minimized.
- The size of the blocks should be of the same order of magnitude.
- The difference in the number of binary variables and continuous variables each block depends on should be minimized.

If running time constraints are outside one of the diagonal blocks in the constraint matrix, then the headway constraints between the arrival and departures of the trains, which depend on the same binary variables, will be in two different blocks, and as a result those two blocks depend on the same binary variables, which means the second goal would not be achieved. Therefore running time constraints should be in the block-diagonal structure to ensure the first two goals. Furthermore headway, separation, and breakable connection constraints all depend on binary variables, and therefore should not be outside the block-diagonal structure

to ensure the first two goals. Only continuity, coupling, and unbreakable connection constraints remain. By only allowing continuity and unbreakable connection constraints to be outside the block-diagonal structure, the first two goals can be achieved.

For the railway model this means that the constraints describing the trains traversing a track, the headway, and separation constraints between the trains on that track should all be in the same block. Since coupling constraints describe relations between the same events as headway constraints (arrivals or departures of trains on the same track) they too should be in the same block.

The steps to reorder the constraint matrix such that it has n_{sub} diagonal blocks are as follows:

1) Group the variables and constraints per track, resulting in n_{T} sets of constraints and variables, where n_{T} is the number of tracks in the model. The continuity constraints are added to the group in which the variable of the departure time of the continuity constraint is in.

2) Merge all sets that are connected via a breakable connection constraint, resulting in n_{T2} remaining sets[2]. Denote these sets of constraints as CON_i for $i = 1, \ldots, n_{\text{T2}}$.

3) Solve a mixed integer quadratic programming (MIQP) problem that minimizes the sum of the maximum difference in the number of constraints of the n_{sub} subproblems and minimizes the number of constraints connecting the n_{sub} subproblems. The values of the binary variables determine for each of the n_{T2} sets to which of the n_{sub} subproblem they are assigned to.

The MIQP problem can be set up as follows:

- Define n_{sub} continuous variables, denoted by S_i for $i = 1, \ldots, n_{\text{sub}}$ that represent the number of constraints each subproblem has.
- Define one continuous variable $S_{\max}$ that is the maximum difference between the values S_i for $i = 1, \ldots, n_{\text{sub}}$.

  In order for $S_{\max}$ to be equal to the maximum difference between the values S_i for $i = 1, \ldots, n_{\text{sub}}$ the following set of constraints is used:

  $$S_{\max} \geq S_i - S_j \quad \text{for } i \in \{1, \ldots, n_{\text{sub}}\},\ j \in \{1, \ldots, n_{\text{sub}}\} \setminus \{i\}.$$

- For each constraint set CON_j, $j = 1, \ldots, n_{\text{T2}}$ define n_{sub} binary variables denoted by $\upsilon_{j,i}$, for $i = 1, \ldots, n_{\text{sub}}$. If binary variable $\upsilon_{j,i} = 1$, then the set CON_j is part of block i of the reordered constraint matrix.
- Define the cost function as

  $$J = \varrho S_{\max} - \sum_{j=1}^{n_{\text{T2}}} \sum_{k=1}^{n_{\text{T2}}} \sum_{i=1}^{n_{\text{sub}}} \upsilon_{j,i} Q_{j,k} \upsilon_{k,i},$$

  where ϱ is a tuning parameter that determines the importance of minimizing $S_{\max}$. The element $Q_{j,k}$ has the value equal to the number of constraints connecting set CON_k to set CON_j. For each combination of two sets CON_j and CON_k the cost function reduces in value equal to $Q_{j,k}$ if and only if both sets are in the same subproblem. The cost function thus reduces in value for every constraint connecting two sets if both sets are in the same subproblem. This corresponds to the number of connecting constraints that are put in one of the diagonal blocks due to the partitioning. The cost function is thus a combination of a weighted $S_{\max}$ (the maximum difference in the number of constraints between subproblems) and the sum of connecting constraints that are no longer outside the diagonal blocks.

[2]The number of subsystems n_{sub} can therefore at most be n_{T2}

- To ensure each constraint set is assigned to exactly one block of the constraint matrix the following set of constraints is used:
$$\sum_{i=1}^{n_{\text{sub}}} v_{j,i} = 1 \quad \text{for } j = 1, \ldots, n_{\text{T2}}.$$
- The number of constraints each block has is determined by the following constraints:
$$S_i = \sum_{j=1}^{n_{\text{T2}}} v_{j,i} s_j \quad \text{for } i \in \{1, \ldots, n_{\text{sub}}\},$$
where s_j is the number of constraints of constraint set CON_j.

With this optimization we can determine for the event time variables of each track to which diagonal block they belong and reorder the constraint matrix into n_{sub} diagonal blocks with nearly empty non-diagonal blocks.

To illustrate the reordering and partitioning procedure consider the following example:

- A network consisting of 5 stations connected in a straight line via 4 tracks.
- The trains only drive in one direction over the four tracks.
- There are 2 trains in total driving over all four tracks.
- The trains do not drive according to a schedule (there are no timetable constraints)
- The goal is to partition the network into two parts.

Let train 1 be modeled by $a_1, a_2, a_3, a_4, d_1, d_2, d_3, d_4$, which are the subsequent arrival and departure events at the 5 stations. There is no arrival event at the first station and no departure event at the last station. The running time constraints for train 1 are given by

$$\begin{aligned} a_1 &\geq d_1 + \tau_{\text{r},1} & a_2 &\geq d_2 + \tau_{\text{r},2} \\ a_3 &\geq d_3 + \tau_{\text{r},3} & a_4 &\geq d_4 + \tau_{\text{r},4}. \end{aligned}$$

The continuity constraints are given by

$$\begin{aligned} d_2 &\geq a_1 + \tau_{\text{d},2} & d_3 &\geq a_2 + \tau_{\text{d},3} \\ d_4 &\geq a_3 + \tau_{\text{d},4}. \end{aligned}$$

Let train 2 be modeled by $a_5, a_6, a_7, a_8, d_5, d_6, d_7, d_8$, which are the subsequent arrival and departure events at the 5 stations. There is no arrival event at the first station and no departure event at the last station. The running time constraints for train 1 are given by

$$\begin{aligned} a_5 &\geq d_5 + \tau_{\text{r},5} & a_6 &\geq d_6 + \tau_{\text{r},6} \\ a_7 &\geq d_7 + \tau_{\text{r},7} & a_8 &\geq d_8 + \tau_{\text{r},8}. \end{aligned}$$

The continuity constraints are given by

$$\begin{aligned} d_6 &\geq a_5 + \tau_{\text{d},6} & d_7 &\geq a_6 + \tau_{\text{d},7} \\ d_8 &\geq a_7 + \tau_{\text{d},6}. \end{aligned}$$

The headway constraints between the two trains are given by

$$\begin{aligned} d_5 \geq & d_1 + \tau_{\mathrm{h,d},5,1} + u_{\mathrm{h},5,1}\beta & d_1 \geq & d_5 + \tau_{\mathrm{h,d},1,5} + (1 - u_{\mathrm{h},5,1})\beta \\ a_5 \geq & a_1 + \tau_{\mathrm{h,a},5,1} + u_{\mathrm{h},5,1}\beta & a_1 \geq & a_5 + \tau_{\mathrm{h,a},1,5} + (1 - u_{\mathrm{h},5,1})\beta \\ d_6 \geq & d_2 + \tau_{\mathrm{h,d},6,2} + u_{\mathrm{h},6,2}\beta & d_2 \geq & d_6 + \tau_{\mathrm{h,d},2,6} + (1 - u_{\mathrm{h},6,2})\beta \\ a_6 \geq & a_2 + \tau_{\mathrm{h,a},6,2} + u_{\mathrm{h},6,2}\beta & a_2 \geq & a_6 + \tau_{\mathrm{h,a},2,6} + (1 - u_{\mathrm{h},2,6})\beta \\ d_7 \geq & d_3 + \tau_{\mathrm{h,d},7,3} + u_{\mathrm{h},7,3}\beta & d_3 \geq & d_7 + \tau_{\mathrm{h,d},3,7} + (1 - u_{\mathrm{h},7,3})\beta \\ a_7 \geq & a_3 + \tau_{\mathrm{h,a},7,3} + u_{\mathrm{h},7,3}\beta & a_3 \geq & a_7 + \tau_{\mathrm{h,a},3,7} + (1 - u_{\mathrm{h},7,3})\beta \\ d_8 \geq & d_4 + \tau_{\mathrm{h,d},8,4} + u_{\mathrm{h},8,4}\beta & d_4 \geq & d_8 + \tau_{\mathrm{h,d},4,8} + (1 - u_{\mathrm{h},8,4})\beta \\ a_8 \geq & a_4 + \tau_{\mathrm{h,a},8,4} + u_{\mathrm{h},8,4}\beta & a_4 \geq & a_8 + \tau_{\mathrm{h,a},4,8} + (1 - u_{\mathrm{h},8,4})\beta. \end{aligned}$$

There are no connection constraints, timetable constraints, or coupling constraints in this example.

The first step of the reordering and partitioning process is to order the events per track. There are 4 tracks, so that results in four sets of constraints. The variables for the four sets are: $\{a_1, a_5, d_1, d_5, u_{\mathrm{h},5,1}\}$, $\{a_2, a_6, d_2, d_6, u_{\mathrm{h},6,2}\}$, $\{a_3, a_7, d_3, d_7, u_{\mathrm{h},7,3}\}$, and $\{a_4, a_8, d_4, d_8, u_{\mathrm{h},8,4}\}$, resulting in the following set of constraints for track 1:

$$\begin{aligned} a_1 \geq & d_1 + \tau_{\mathrm{r},1} & a_5 \geq & d_5 + \tau_{\mathrm{r},5} \\ d_5 \geq & d_1 + \tau_{\mathrm{h,d},5,1} + u_{\mathrm{h},5,1}\beta & d_1 \geq & d_5 + \tau_{\mathrm{h,d},1,5} + (1 - u_{\mathrm{h},5,1})\beta \\ a_5 \geq & a_1 + \tau_{\mathrm{h,a},5,1} + u_{\mathrm{h},5,1}\beta & a_1 \geq & a_5 + \tau_{\mathrm{h,a},1,5} + (1 - u_{\mathrm{h},5,1})\beta, \end{aligned}$$

for track 2:

$$\begin{aligned} a_2 \geq & d_2 + \tau_{\mathrm{r},2} & a_6 \geq & d_6 + \tau_{\mathrm{r},6} \\ d_6 \geq & d_2 + \tau_{\mathrm{h,d},6,2} + u_{\mathrm{h},6,2}\beta & d_2 \geq & d_6 + \tau_{\mathrm{h,d},2,6} + (1 - u_{\mathrm{h},6,2})\beta \\ a_6 \geq & a_2 + \tau_{\mathrm{h,a},6,2} + u_{\mathrm{h},6,2}\beta & a_2 \geq & a_6 + \tau_{\mathrm{h,a},2,6} + (1 - u_{\mathrm{h},2,6})\beta \\ d_6 \geq & a_5 + \tau_{\mathrm{d},6}, \end{aligned}$$

for track 3:

$$\begin{aligned} a_3 \geq & d_3 + \tau_{\mathrm{r},3} & a_7 \geq & d_7 + \tau_{\mathrm{r},7} \\ d_7 \geq & d_3 + \tau_{\mathrm{h,d},7,3} + u_{\mathrm{h},7,3}\beta & d_3 \geq & d_7 + \tau_{\mathrm{h,d},3,7} + (1 - u_{\mathrm{h},7,3})\beta \\ a_7 \geq & a_3 + \tau_{\mathrm{h,a},7,3} + u_{\mathrm{h},7,3}\beta & a_3 \geq & a_7 + \tau_{\mathrm{h,a},3,7} + (1 - u_{\mathrm{h},7,3})\beta \\ d_7 \geq & a_6 + \tau_{\mathrm{d},7}, \end{aligned}$$

and for track 4:

$$\begin{aligned} a_4 \geq & d_4 + \tau_{\mathrm{r},4} & a_8 \geq & d_8 + \tau_{\mathrm{r},8} \\ d_8 \geq & d_4 + \tau_{\mathrm{h,d},8,4} + u_{\mathrm{h},8,4}\beta & d_4 \geq & d_8 + \tau_{\mathrm{h,d},4,8} + (1 - u_{\mathrm{h},8,4})\beta \\ a_8 \geq & a_4 + \tau_{\mathrm{h,a},8,4} + u_{\mathrm{h},8,4}\beta & a_4 \geq & a_8 + \tau_{\mathrm{h,a},4,8} + (1 - u_{\mathrm{h},8,4})\beta \\ d_8 \geq & a_7 + \tau_{\mathrm{d},6}. \end{aligned}$$

Since we have no connection constraints step 2 of the reordering process can be skipped and these sets can be described by CON_1, CON_2, CON_3, and CON_4. Now we can set up the MIQP problem to reduce the number of sets to two.

- First define 2 variables S_1 and S_2 that represent the number of variables each resulting set has.
- Define the variable $S_{\max}$ and define the constraints such that it is the maximum difference between S_1 and S_2:

$$S_{\max} \geq S_1 - S_2 \qquad\qquad S_{\max} \geq S_2 - S_1.$$

- For each constraint set CON_j, $j = 1, 2, 3, 4$ define 2 binary variables denoted by $\upsilon_{j,i}$, for $i = 1, 2$, resulting in 8 binary variables $\upsilon_{1,1}$, $\upsilon_{1,2}$, $\upsilon_{2,1}$, $\upsilon_{2,2}$, $\upsilon_{3,1}$, $\upsilon_{3,2}$, $\upsilon_{4,1}$, and $\upsilon_{4,2}$.
- Define the cost function as

$$J = \varrho S_{\max} - \sum_{j=1}^{4}\sum_{k=1}^{4}\sum_{i=1}^{2} \upsilon_{j,i} Q_{j,k} \upsilon_{k,i},$$

  where ϱ is a tuning parameter that determines the importance of minimizing $S_{\max}$. In this example we chose $\rho = 0.5$ and the matrix Q is given by

$$Q = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

  since there are 2 continuity constraints connecting set 1 to set 2, 2 continuity constraints connecting set 2 to set 3, and 12 continuity constraints connecting set 3 to set 4.
- To ensure each constraint set is assigned to exactly one block of the constraint matrix the following set of constraints is used:

$$\begin{aligned} \upsilon_{1,1} + \upsilon_{1,2} &= 1 & \upsilon_{2,1} + \upsilon_{2,2} &= 1 \\ \upsilon_{3,1} + \upsilon_{3,2} &= 1 & \upsilon_{4,1} + \upsilon_{4,2} &= 1 \end{aligned}$$

- The number of constraints each block has is determined by the following constraints:

$$\begin{aligned} S_1 &= \upsilon_{1,1} s_1 + \upsilon_{2,1} s_2 + \upsilon_{3,1} s_3 + \upsilon_{4,1} s_4 \\ S_2 &= \upsilon_{1,2} s_1 + \upsilon_{2,2} s_2 + \upsilon_{3,2} s_3 + \upsilon_{4,2} s_4, \end{aligned}$$

  where $s_1 = 6$, $s_1 = 8$, $s_1 = 8$, and $s_1 = 8$ are the number of constraints per set.

Now the MIQP is fully described for the example and can be solved. The optimal solution is found as

$$\begin{aligned} \upsilon_{1,1} &= 1 & \upsilon_{1,2} &= 0 \\ \upsilon_{2,1} &= 1 & \upsilon_{2,2} &= 0 \\ \upsilon_{3,1} &= 0 & \upsilon_{3,2} &= 1 \\ \upsilon_{4,1} &= 0 & \upsilon_{4,2} &= 1 \\ S_1 &= 14 & S_2 &= 16 \\ S_{\max} &= 2 & J &= -3 \end{aligned}$$

This means that the sets of track 1 and track 2 should be combined and the sets for track 3 and track 4 should be combined resulting in the following two sets of constraints:

$$\begin{aligned} a_1 &\geq d_1 + \tau_{\text{r},1} & a_5 &\geq d_5 + \tau_{\text{r},5} \\ d_5 &\geq d_1 + \tau_{\text{h,d},5,1} + u_{\text{h},5,1}\beta & d_1 &\geq d_5 + \tau_{\text{h,d},1,5} + (1 - u_{\text{h},5,1})\beta \\ a_5 &\geq a_1 + \tau_{\text{h,a},5,1} + u_{\text{h},5,1}\beta & a_1 &\geq a_5 + \tau_{\text{h,a},1,5} + (1 - u_{\text{h},5,1})\beta \\ a_2 &\geq d_2 + \tau_{\text{r},2} & a_6 &\geq d_6 + \tau_{\text{r},6} \\ d_6 &\geq d_2 + \tau_{\text{h,d},6,2} + u_{\text{h},6,2}\beta & d_2 &\geq d_6 + \tau_{\text{h,d},2,6} + (1 - u_{\text{h},6,2})\beta \\ a_6 &\geq a_2 + \tau_{\text{h,a},6,2} + u_{\text{h},6,2}\beta & a_2 &\geq a_6 + \tau_{\text{h,a},2,6} + (1 - u_{\text{h},2,6})\beta \\ d_6 &\geq a_5 + \tau_{\text{d},6} & d_2 &\geq a_1 + \tau_{\text{d},2}, \end{aligned}$$

and:

$$
\begin{aligned}
\boldsymbol{d_7 \geq a_6 + \tau_{\mathrm{d},7}} & \qquad \boldsymbol{d_3 \geq a_2 + \tau_{\mathrm{d},3}} \\
d_8 \geq & a_7 + \tau_{\mathrm{d},6} & d_4 \geq & a_3 + \tau_{\mathrm{d},4} \\
a_3 \geq & d_3 + \tau_{\mathrm{r},3} & a_7 \geq & d_7 + \tau_{\mathrm{r},7} \\
d_7 \geq & d_3 + \tau_{\mathrm{h,d},7,3} + u_{\mathrm{h},7,3}\beta & d_3 \geq & d_7 + \tau_{\mathrm{h,d},3,7} + (1 - u_{\mathrm{h},7,3})\beta \\
a_7 \geq & a_3 + \tau_{\mathrm{h,a},7,3} + u_{\mathrm{h},7,3}\beta & a_3 \geq & a_7 + \tau_{\mathrm{h,a},3,7} + (1 - u_{\mathrm{h},7,3})\beta \\
a_4 \geq & d_4 + \tau_{\mathrm{r},4} & a_8 \geq & d_8 + \tau_{\mathrm{r},8} \\
d_8 \geq & d_4 + \tau_{\mathrm{h,d},8,4} + u_{\mathrm{h},8,4}\beta & d_4 \geq & d_8 + \tau_{\mathrm{h,d},4,8} + (1 - u_{\mathrm{h},8,4})\beta \\
a_8 \geq & a_4 + \tau_{\mathrm{h,a},8,4} + u_{\mathrm{h},8,4}\beta & a_4 \geq & a_8 + \tau_{\mathrm{h,a},4,8} + (1 - u_{\mathrm{h},8,4})\beta,
\end{aligned}
$$

with only two constraints, denoted in bold, connecting the two sets.

4 Distributed Model Predictive Control

For very large and complex systems the optimization problem resulting from a centralized MPC approach may not be solvable within the time available, or the solution returned may be far from optimal. A solution for this problem can be to use distributed model predictive control (DMPC) (Camponogara et al., 2002, Dunbar, 2007, Farina and Scattolini, 2012, Richards and How, 2007). Most DMPC approaches have been developed for continuous or discrete-time systems (Maestre and Negenborn, 2013). Since our model is a discrete-event model, and these DMPC approaches make use of continuous or discrete time stability and robustness theories to ensure convergence, these DMPC approaches cannot be directly applied to our problem. Instead, we introduce our own DMPC methods for discrete event systems.

In DMPC the system is partitioned into a number of subsystems and each subsystem is controlled by its own model predictive controller. Each controller therefore has to solve its own optimization or subproblem. In the case of railway traffic management each subsystem consists of a subset of all stations and tracks and the trains on that part of the network and the controller of the subsystem can only control that part of the network. In that part of the network the local controller can take any dispatching action. In the rest of the network the other controllers decide the schedule.

The advantage of partitioning the system into subsystems is that the optimization problem that must be solved to find the control inputs for the subsystems can in general be solved much faster since it is smaller and less complex than the optimization problem used to determine the control inputs for the centralized MPC method. The downside is that the obtained control inputs may be suboptimal for the complete system. In Section 3 we will explain the partitioning method, but first we will discuss the DMPC methods.

In the previous section we have shown that the constraint matrix can be reordered such that the optimization problem at time instant $t(\kappa)$ of the MPC problem can be written as

$$
\min_{z(\kappa)} \begin{bmatrix} \tilde{c}_1^\top(\kappa) & \tilde{c}_2^\top(\kappa) & \dots & \tilde{c}_{n_{\mathrm{sub}}}^\top(\kappa) \end{bmatrix} \begin{bmatrix} \tilde{z}_1^\top(\kappa) & \tilde{z}_2^\top(\kappa) & \dots & \tilde{z}_{n_{\mathrm{sub}}}^\top(\kappa) \end{bmatrix}^\top \tag{27}
$$

$$
\text{s.t.} \begin{bmatrix} A_{1,1}(\kappa) & A_{1,2}(\kappa) & \dots & A_{1,n_{\mathrm{sub}}}(\kappa) \\ A_{2,1}(\kappa) & \ddots & & A_{2,n_{\mathrm{sub}}}(\kappa) \\ \vdots & & \ddots & \vdots \\ A_{n_{\mathrm{sub}},1}(\kappa) & A_{n_{\mathrm{sub}},2}(\kappa) & \dots & A_{n_{\mathrm{sub}},n_{\mathrm{sub}}}(\kappa) \end{bmatrix} \begin{bmatrix} \tilde{z}_1(\kappa) \\ \tilde{z}_2(\kappa) \\ \vdots \\ \tilde{z}_{n_{\mathrm{sub}}}(\kappa) \end{bmatrix} \leq \begin{bmatrix} \tilde{b}_1(\kappa) \\ \tilde{b}_2(\kappa) \\ \vdots \\ \tilde{b}_{n_{\mathrm{sub}}}(\kappa) \end{bmatrix}, \tag{28}
$$

where $\tilde{c}_i(\kappa)$, $\tilde{z}_i(\kappa)$, $\tilde{b}_i(\kappa)$ for $i \in \{1, \dots, n_{\mathrm{sub}}\}$ are vectors of appropriate size, $A_{i,j}(\kappa)$ for $i \in \{1, \dots, n_{\mathrm{sub}}\}$, $j \in$

$\{1, \ldots, n_{\text{sub}}\}$ are matrices of appropriate size, and

$$\begin{aligned} \tilde{z}_i(\kappa) &= \begin{bmatrix} \tilde{\mathcal{X}}_i^\top(\kappa) & \tilde{\mathcal{U}}_i^\top(\kappa) \end{bmatrix}^\top \\ A_{i,i}(\kappa) &= \begin{bmatrix} A_{i,i,x}(\kappa) & A_{i,i,\mathcal{U}}(\kappa) \end{bmatrix} \\ A_{i,j}(\kappa) &= \begin{bmatrix} A_{i,j,x}(\kappa) & \mathbf{0}(\kappa) \end{bmatrix}, \end{aligned}$$

for $i \in \{1, \ldots, n_{\text{sub}}\}$ and $j \in \{1, \ldots, n_{\text{sub}}\} \setminus \{i\}$, and where $\tilde{\mathcal{X}}_i(\kappa)$ and $\tilde{\mathcal{U}}_i(\kappa)$ are vectors of appropriate size. The matrices $A_{i,i}(\kappa)$ are split up into a part $A_{i,i,x}(\kappa)$ that is multiplied by $\tilde{\mathcal{X}}_i(\kappa)$, and a part $A_{i,i,\mathcal{U}}(\kappa)$ that is multiplied by $\tilde{\mathcal{U}}_i(\kappa)$ in (28). The matrices $A_{i,j}(\kappa)$ are also split up into two parts. One part $A_{i,j,x}(\kappa)$ is multiplied by $\tilde{\mathcal{X}}_j(\kappa)$ in (28). The part that is multiplied by $\tilde{\mathcal{U}}_j(\kappa)$ in (28) is guaranteed to be a zero matrix. In Section 3 we show how we can reorder the constraint matrix such that it gets the assumed structure.

4.1 Distributed method 1

Based on the reordered constraint matrix two methods for solving the problem in a distributed manner are developed. The resulting structure of matrix shows that there are only a few constraints connecting each of the blocks, so there is very little interaction between the variables of each block. Therefore it makes sense to develop DMPC methods based on this block structure.

For the first method each subproblem models the arrival and departures of all trains in the network, but it can only reorder the trains in one part of the network. Each subproblem optimizes the centralized cost function while considering all constraints of the centralized problem, but for subproblem i at time instant $t(\kappa)$ only $\mathcal{X}(\kappa)$ and $\tilde{\mathcal{U}}_i(\kappa)$ can be used to optimize the cost function. By using the same cost function as the centralized problem, each subproblem considers the effects of its dispatching actions on the entire network. By limiting the binary variables that can be changed to $\mathcal{U}_i(\kappa)$ the subproblem is computationally less complex to solve than the centralized problem. As a result the MILP problem can be solved much faster. The subproblems are then solved in sequence. When solving a subproblem the binary variables of the other subproblems are fixed to the value found when their respective subproblems were solved previously. Since all subproblems share the same cost function, the cost function of the centralized problem is reduced every time a subproblem is solved. This continues until none of the subproblems can improve the cost function anymore. Note that a feasible initial solution can always be found since a feasible solution is the solution where no dispatching actions are taken; more specifically, all binary variables will be zero, and the trains would just continue to drive as regular and no order would be changed or connections broken. Another possibility would be to use a heuristic method such as first-come, first-served to find an initial solution.

To summarize the method:

The overall problem is split into n_{sub} subproblems. The i-th subproblem can be written as[3]:

$$\begin{aligned} \min_{\mathcal{X}(\kappa), \tilde{\mathcal{U}}_i(\kappa)} \quad & c^\top(\kappa)\, z(\kappa) \\ \text{s.t.} \quad & A(\kappa)\, z(\kappa) \leq b(\kappa) \\ & \tilde{\mathcal{U}}_j(\kappa) = \hat{\mathcal{U}}_j(\kappa) \text{ for } j \neq i. \end{aligned}$$

where $\hat{\mathcal{U}}_j(\kappa)$ is the vector containing the binary variables of subproblem j found in a previous iteration. The steps to determine the solution are:

1) Define the iteration counter $l = 0$. Define an initial estimate for the solution of the centralized MPC $z(\kappa)$ denoted by $\hat{z}^l(\kappa)$.

[3]Recall that $z(\kappa)$ is split up into n_{sub} sub-vectors $\tilde{z}_i(\kappa)$, and $\tilde{\mathcal{U}}_i(\kappa)$ are the binary variables in $\tilde{z}_i(\kappa)$ and are part of $z(\kappa)$.

2) Use $\hat{z}^l(\kappa)$ as an initial solution for subproblem i:

$$\begin{aligned} \min_{\mathcal{X}(\kappa),\tilde{\mathcal{U}}_i(\kappa)} \quad & c^\top(\kappa)\, z(\kappa) \\ \text{s.t.} \quad & A(\kappa)\, z(\kappa) \leq b(\kappa) \\ & \tilde{\mathcal{U}}_j(\kappa) = \hat{\mathcal{U}}_j^l(\kappa) \text{ for } j \neq i. \end{aligned}$$

and solve it. Increase the iteration counter by one: $l = l + 1$ and denote the solution as $\hat{z}^l(\kappa)$.

3) Repeat step 2 for the other subproblems.

4) Repeat steps 2 and 3 until $||\hat{z}^l(\kappa) - \hat{z}^{l-n_{\text{sub}}}(\kappa)|| < \Delta$, where Δ is a very small value.

Since a feasible initial estimate can always be found, and all subproblems consider the centralized problem, but can only change a limited number of binary variables, a feasible solution for the centralized problem will always be found in step 2. Furthermore every feasible solution found can and will be used as a starting solution for the next subproblem. Therefore, every subproblem either improves the found solution or cannot find a better solution than the current solution. If no better solution than the current solution is found, the current solution is used for the next subproblem. Once no subproblem can improve the solution the algorithm stops. We call this method "DMPC method 1".

Recall the example in Section 3 of the two trains traversing the four tracks. For DMPC method 1 the constraints of subproblem 1 can be written as:

$$\begin{aligned}
d_1 - a_1 &\leq -\tau_{\text{r},1} & d_5 - a_5 &\leq -\tau_{\text{r},5} \\
d_1 - d_5 + u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,d},5,1} & d_5 - d_1 - u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,d},1,5} - \beta \\
a_1 - a_5 + u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,a},5,1} & a_5 - a_1 - u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,a},1,5} - \beta \\
d_2 - a_2 &\leq -\tau_{\text{r},2} & d_6 - a_6 &\leq -\tau_{\text{r},6} \\
d_2 - d_6 + u_{\text{h},6,2}\beta &\leq -\tau_{\text{h,d},6,2} & d_6 - d_2 - u_{\text{h},6,2}\beta &\leq -\tau_{\text{h,d},2,6} - \beta \\
a_2 - a_6 + u_{\text{h},6,2}\beta &\leq -\tau_{\text{h,a},6,2} & a_6 - a_2 - u_{\text{h},2,6}\beta &\leq -\tau_{\text{h,a},2,6} - \beta \\
a_5 - d_6 &\leq -\tau_{\text{d},6} & a_6 - d_7 &\leq -\tau_{\text{d},7} \\
a_7 - d_8 &\leq -\tau_{\text{d},6} & a_1 - d_2 &\leq -\tau_{\text{d},2} \\
a_2 - d_3 &\leq -\tau_{\text{d},3} & a_3 - d_4 &\leq -\tau_{\text{d},4} \\
d_3 - a_3 &\leq -\tau_{\text{r},3} & d_7 - a_7 &\leq -\tau_{\text{r},7} \\
d_3 - d_7 + u_{\text{h},7,3}\beta &\leq -\tau_{\text{h,d},7,3} & d_7 - d_3 - u_{\text{h},7,3}\beta &\leq -\tau_{\text{h,d},3,7} - \beta \\
a_3 - a_7 + u_{\text{h},7,3}\beta &\leq -\tau_{\text{h,a},7,3} & a_7 - a_3 - u_{\text{h},7,3}\beta &\leq -\tau_{\text{h,a},3,7} - \beta \\
d_4 - a_4 &\leq -\tau_{\text{r},4} & d_8 - a_8 &\leq -\tau_{\text{r},8} \\
d_4 - d_8 + u_{\text{h},8,4} &\leq -\tau_{\text{h,d},8,4} & d_8 - d_4 - u_{\text{h},8,4} &\leq -\tau_{\text{h,d},4,8} - \beta \\
a_4 - a_8 + u_{\text{h},8,4} &\leq -\tau_{\text{h,a},8,4} & a_8 - a_4 - u_{\text{h},8,4} &\leq -\tau_{\text{h,a},4,8} - \beta \\
\boldsymbol{u_{\text{h},8,4}} &= \boldsymbol{u^l_{\text{h},8,4}} \\
\boldsymbol{u_{\text{h},7,3}} &= \boldsymbol{u^l_{\text{h},7,3}},
\end{aligned}$$

where $u^l_{\text{h},7,3}$, $u^l_{\text{h},8,4}$ are the values of the binary variables determined in the l-th iteration of the DMPC method. The variables that can be adjusted by the optimization are $\{a_1, a_2, a_3, a_4, d_1, d_2, d_3, d_4, u_{\text{h},5,1}, u_{\text{h},6,2}, a_5, a_6, a_7, a_8, d_5, d_6, d_7, d_8\}$. The only variables it cannot change are $u_{\text{h},7,3}$ and $u_{\text{h},8,4}$, these are determined by subproblem 2 and are considered constant during the optimization of subproblem 1 as is shown in the last two equality constraints (shown in bold). The constraints for

subproblem 2 are written as:

$$
\begin{aligned}
d_1 - a_1 &\leq -\tau_{\mathrm{r},1} & \qquad d_5 - a_5 &\leq -\tau_{\mathrm{r},5} \\
d_1 - d_5 + u_{\mathrm{h},5,1}\beta &\leq -\tau_{\mathrm{h,d},5,1} & \qquad d_5 - d_1 - u_{\mathrm{h},5,1}\beta &\leq -\tau_{\mathrm{h,d},1,5} - \beta \\
a_1 - a_5 + u_{\mathrm{h},5,1}\beta &\leq -\tau_{\mathrm{h,a},5,1} & \qquad a_5 - a_1 - u_{\mathrm{h},5,1}\beta &\leq -\tau_{\mathrm{h,a},1,5} - \beta \\
d_2 - a_2 &\leq -\tau_{\mathrm{r},2} & \qquad d_6 - a_6 &\leq -\tau_{\mathrm{r},6} \\
d_2 - d_6 + u_{\mathrm{h},6,2}\beta &\leq -\tau_{\mathrm{h,d},6,2} & \qquad d_6 - d_2 - u_{\mathrm{h},6,2}\beta &\leq -\tau_{\mathrm{h,d},2,6} - \beta \\
a_2 - a_6 + u_{\mathrm{h},6,2}\beta &\leq -\tau_{\mathrm{h,a},6,2} & \qquad a_6 - a_2 - u_{\mathrm{h},2,6}\beta &\leq -\tau_{\mathrm{h,a},2,6} - \beta \\
a_5 - d_6 &\leq -\tau_{\mathrm{d},6} & \qquad a_6 - d_7 &\leq -\tau_{\mathrm{d},7} \\
a_7 - d_8 &\leq -\tau_{\mathrm{d},6} & \qquad a_1 - d_2 &\leq -\tau_{\mathrm{d},2} \\
a_2 - d_3 &\leq -\tau_{\mathrm{d},3} & \qquad a_3 - d_4 &\leq -\tau_{\mathrm{d},4} \\
d_3 - a_3 &\leq -\tau_{\mathrm{r},3} & \qquad d_7 - a_7 &\leq -\tau_{\mathrm{r},7} \\
d_3 - d_7 + u_{\mathrm{h},7,3}\beta &\leq -\tau_{\mathrm{h,d},7,3} & \qquad d_7 - d_3 - u_{\mathrm{h},7,3}\beta &\leq -\tau_{\mathrm{h,d},3,7} - \beta \\
a_3 - a_7 + u_{\mathrm{h},7,3}\beta &\leq -\tau_{\mathrm{h,a},7,3} & \qquad a_7 - a_3 - u_{\mathrm{h},7,3}\beta &\leq -\tau_{\mathrm{h,a},3,7} - \beta \\
d_4 - a_4 &\leq -\tau_{\mathrm{r},4} & \qquad d_8 - a_8 &\leq -\tau_{\mathrm{r},8} \\
d_4 - d_8 + u_{\mathrm{h},8,4} &\leq -\tau_{\mathrm{h,d},8,4} & \qquad d_8 - d_4 - u_{\mathrm{h},8,4} &\leq -\tau_{\mathrm{h,d},4,8} - \beta \\
a_4 - a_8 + u_{\mathrm{h},8,4} &\leq -\tau_{\mathrm{h,a},8,4} & \qquad a_8 - a_4 - u_{\mathrm{h},8,4} &\leq -\tau_{\mathrm{h,a},4,8} - \beta \\
\boldsymbol{u_{\mathrm{h},5,1}} &= \boldsymbol{u^{l}_{\mathrm{h},5,1}} \\
\boldsymbol{u_{\mathrm{h},6,2}} &= \boldsymbol{u^{l}_{\mathrm{h},6,2}},
\end{aligned}
$$

where $u^l_{\mathrm{h},5,1}$, $u^l_{\mathrm{h},6,2}$ are the values of the binary variables determined in the l-th iteration of the DMPC method. The variables that can be adjusted by the optimization are $\{a_1, a_2, a_3, a_4, d_1, d_2, d_3, d_4, a_5, a_6, a_7, a_8, d_5, d_6, d_7, d_8, u_{\mathrm{h},7,3}, u_{\mathrm{h},8,4}\}$. The only variables it cannot change are $u_{\mathrm{h},5,1}$ and $u_{\mathrm{h},6,2}$, these are determined by subproblem 1 and are considered constant during the optimization of subproblem 2, as shown in the last two equality constraints (shown in bold).

Clearly both subproblems consider all constraints of the global MPC, only a few constraints are added such that the variables the subproblem cannot adjust are constant and have a value. Because each subproblem has all constraints it always find a feasible global solution and the consequences for the entire network of reordering trains in a part of the network are modeled.

You could say subproblem 1 can only change the order of the trains on tracks 1 and 2, the order of the trains on tracks 3 and 4 are predetermined by an outside factor (subproblem 2). It can however change all arrival and departure times, as long as it respects the specified order of the trains on tracks 3 and 4.

For subproblem 2 the opposite is true, it can only change the order of the trains on tracks 3 and 4. The order of the trains on track 1 and 2 are determined by an outside factor (subproblem 1). Subproblem 2 can also change all arrival and departure times, as long as it respects the specified order of the trains on tracks 1 and 2.

4.2 Distributed method 2

Since the number of constraints of each subproblem for DMPC method 1 is equal to the number of constraints for the central MPC problem, for very large systems the speed-up in computation time may still not enough. For very large systems increasing the number of subproblem is not a solution either, since the number of constraints remains the same. Therefore another DMPC approach is proposed.

For the second DMPC method each subproblem only models the arrivals and departures of the trains in the part of the network it can control, it does not consider the effects of its control actions on the arrival and

departure times of the trains outside its area of control. Therefore each subproblem only considers a part of the MPC problem in (28). It only optimizes the continuous and binary variables of one of the diagonal blocks. Each subproblem can then be written as:

$$\min_{\tilde{z}_i(\kappa)} \tilde{c}_i^\top(\kappa)\, \tilde{z}_i(\kappa) \tag{29}$$

$$\text{s.t. } A_{i,i}(\kappa)\tilde{z}_i(\kappa) \leq b_i(\kappa) - \sum_{j\in\{1,\ldots,n_{\text{sub}}\}\setminus\{i\}} A_{i,j}(\kappa)\tilde{z}_j(\kappa), \tag{30}$$

where $\tilde{z}_j(\kappa)$, for $j \in \{1,\ldots,n_{\text{sub}}\} \setminus \{i\}$ is fixed in this subproblem and determined by the other local model predictive controllers.

The steps to determine a feasible centralized solution are:

1) Define an initial estimate for the solution of the centralized MPC $z(\kappa)$ denoted by $\hat{z}(\kappa)$, define the iteration counter $l = 0$, and define $\hat{\tilde{z}}_i^l(\kappa) = \hat{z}_i(\kappa)$ for $i = 1,\ldots,n_{\text{sub}}$.
2) For subproblem i, given by (29)–(30), assume $\tilde{z}_j(\kappa)$ for $j \in \{1,\ldots,n_{\text{sub}}\} \setminus \{i\}$ is known and equal to $\hat{\tilde{z}}_j^{l+1}(\kappa)$ and solve subproblem i for $i = 1$. Denote the solution as $\hat{\tilde{z}}_i^l(\kappa)$.
3) Repeat step 2 for $i = 2,\ldots,n_{\text{sub}}$.
4) Increase the iteration counter by one: $l = l + 1$ and repeat steps 2 and 3 until $|\hat{\tilde{z}}_i^l(\kappa) - \hat{\tilde{z}}_i^{l-1}(\kappa)| < \Delta$ for all subproblems ($i = 1,\ldots,n_{\text{sub}}$), where Δ is a very small value. If there is no convergence within $\text{iter}_{\max}$ iterations then lock the binary variables $\tilde{\mathcal{U}}(\kappa)$ to their last determined values and determine the corresponding continuous variables $\tilde{\mathcal{X}}(\kappa)$.

A feasible solution can always be found in step 2, since taking no dispatching actions will always result in a feasible solution; the delays however may be much higher in that case compared to the optimal solution.

The advantage of this method is that the subproblems are even smaller and can be solved faster. However since all subproblems only consider a part of the network, the subproblems have no way of taking into account the effects of their control actions on the other subproblems. It is therefore likely that the solutions found will be worse than those found with DMPC method 1. We will simply call this "DMPC method 2".

Recall the example in Section 3 of the two trains traversing the four tracks. For DMPC method 2 the constraints of subproblem 1 can be written as:

$$\begin{aligned}
d_1 - a_1 &\leq -\tau_{\text{r},1} & d_5 - a_5 &\leq -\tau_{\text{r},5}\\
d_1 - d_5 + u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,d},5,1} & d_5 - d_1 - u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,d},1,5} - \beta\\
a_1 - a_5 + u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,a},5,1} & a_5 - a_1 - u_{\text{h},5,1}\beta &\leq -\tau_{\text{h,a},1,5} - \beta\\
d_2 - a_2 &\leq -\tau_{\text{r},2} & d_6 - a_6 &\leq -\tau_{\text{r},6}\\
d_2 - d_6 + u_{\text{h},6,2}\beta &\leq -\tau_{\text{h,d},6,2} & d_6 - d_2 - u_{\text{h},6,2}\beta &\leq -\tau_{\text{h,d},2,6} - \beta\\
a_2 - a_6 + u_{\text{h},6,2}\beta &\leq -\tau_{\text{h,a},6,2} & a_6 - a_2 - u_{\text{h},2,6}\beta &\leq -\tau_{\text{h,a},2,6} - \beta\\
a_5 - d_6 &\leq -\tau_{\text{d},6} & a_1 - d_2 &\leq -\tau_{\text{d},2}
\end{aligned}$$

where all the variables that can be adjusted by the optimization are on the left-hand side of the equations. These variables are: $\{a_1, a_2, a_5, a_6, d_1, d_2, d_5, d_6, u_{\text{h},5,1}, u_{\text{h},6,2}\}$. The variables it cannot change are $\{a_3, a_4, d_3, d_4, a_7, a_8, d_7, d_8, u_{\text{h},7,3}, u_{\text{h},8,4}\}$, these are determined by subproblem 2. The constraints for sub-

problem 2 are written as:

$$
\begin{aligned}
\boldsymbol{-d_7 \leq -\tau_{\mathrm{d},7} - a_6} \qquad & \qquad a_7 - d_8 \leq -\tau_{\mathrm{d},6} \\
\boldsymbol{-d_3 \leq -\tau_{\mathrm{d},3} - a_2} \qquad & \qquad a_3 - d_4 \leq -\tau_{\mathrm{d},4} \\
d_3 - a_3 \leq -\tau_{\mathrm{r},3} \qquad & \qquad d_7 - a_7 \leq -\tau_{\mathrm{r},7} \\
d_3 - d_7 + u_{\mathrm{h},7,3}\beta \leq -\tau_{\mathrm{h,d},7,3} \qquad & \qquad d_7 - d_3 - u_{\mathrm{h},7,3}\beta \leq -\tau_{\mathrm{h,d},3,7} - \beta \\
a_3 - a_7 + u_{\mathrm{h},7,3}\beta \leq -\tau_{\mathrm{h,a},7,3} \qquad & \qquad a_7 - a_3 - u_{\mathrm{h},7,3}\beta \leq -\tau_{\mathrm{h,a},3,7} - \beta \\
d_4 - a_4 \leq -\tau_{\mathrm{r},4} \qquad & \qquad d_8 - a_8 \leq -\tau_{\mathrm{r},8} \\
d_4 - d_8 + u_{\mathrm{h},8,4} \leq -\tau_{\mathrm{h,d},8,4} \qquad & \qquad d_8 - d_4 - u_{\mathrm{h},8,4} \leq -\tau_{\mathrm{h,d},4,8} - \beta \\
a_4 - a_8 + u_{\mathrm{h},8,4} \leq -\tau_{\mathrm{h,a},8,4} \qquad & \qquad a_8 - a_4 - u_{\mathrm{h},8,4} \leq -\tau_{\mathrm{h,a},4,8} - \beta,
\end{aligned}
$$

where all the variables that can be adjusted by the optimization are on the left-hand side of the equations. These variables are: $\{a_3, a_4, d_3, d_4, a_7, a_8, d_7, d_8, u_{\mathrm{h},7,3}, u_{\mathrm{h},8,4}\}$. The variables it cannot change are $\{a_3, a_4, d_3, d_4, a_7, a_8, d_7, d_8, u_{\mathrm{h},7,3}, u_{\mathrm{h},8,4}\}$, these are determined by subproblem 1. The two constraints in bold show how subproblem 2 depends on the result of subproblem 1. In these two constraints a variable of subproblem 1 is in the right-hand side of the constraint and during the optimization of subproblem it is considered as a constant.

From the constraint sets it is clear that only a subset of the constraints of the global MPC problem are considered. Only the constraints that are needed to determine the variables the subproblem can change are given. The rest of the constraints are left out and the variables it cannot change are put in the right-hand side of the constraints and are considered constant. As a result each subproblem only considers the consequences of its decisions on the tracks it controls and not the rest of the network.

In this case subproblem 1 can adjust the order and the arrival and departure times of the trains on tracks 1 and 2, but has no knowledge at all about the effects on the arrival and departure times of the trains on tracks 3 and 4. Subproblem 2 can only adjust the order and the arrival and departure times of the trains on tracks 3 and 4, furthermore it depends on the arrival times of train runs 2 and 6 which it cannot influence. Also subproblem 2 has no knowledge at all about the effects on the arrival and departure times of the trains on tracks 1 and 2. In this example subproblem 2 does not affect subproblem 1, since the trains only run in one direction and there are no trains going from tracks 3 and 4 to tracks 1 and 2.

4.3 Adjusting cost functions

For the second DMPC method in this section the local model predictive controllers do not consider the delay propagation to the other parts of the network and as a result the train orders at the border of their area of control tend to be suboptimal for the total network, since a large part of the effects of the order changes are only noticed in the next area. To reduce the delay propagation from one area controlled by a local model predictive controller to another area we will improve the second method by adjusting the weights of the events of the trains leaving to another area. The weights are adjusted only at the start of the optimization at time instants $t(\kappa)$, $\kappa = 0, 1, \ldots$. The weights that need to be adjusted can be found easily. For the local model predictive controller i the indices of the weights of the MILP problem in $\tilde{c}_i(\kappa)$ that we want to adjust can be determined by finding the variables of MILP problem i that are connected to MILP problem j for $j \in \{1, \ldots, n_{\mathrm{sub}}\} \setminus \{i\}$ through a constraint.

If we look at how MILP problem i is defined in (29)–(30) it is clear that the indices of the variables of MILP problem j on which MILP problem i depends coincide with the indices of the non-zero columns in $A_{j,i}(\kappa)$. Then for MILP problem j for $j \in \{1, \ldots, n_{\mathrm{sub}}\} \setminus \{i\}$ the indices of $\tilde{c}_j(\kappa)$ that need to be adjusted therefore coincide with the indices of the non-zero columns of $A_{j,i}(\kappa)$ for $i = 1, \ldots, n_{\mathrm{sub}}$. Denote these weights by $\tilde{c}_j^{\mathrm{out}}(\kappa)$ and denote the corresponding variables as $\tilde{z}_j^{\mathrm{out}}(\kappa)$.

By increasing the weights $\tilde{c}_j^{\mathrm{out}}(\kappa)$ the local controllers are forced to focus on reducing the delays of the

corresponding events, and by doing so possibly reduce the delay propagation to the other areas, at the cost of the total delay in the local area. There are many possible choices for the values of the weights $\tilde{c}_j^{\text{out}}(\kappa)$. In this paper we will consider two different choices:

- Increase the values of weights $\tilde{c}_j^{\text{out}}(\kappa)$ with the same factor.
- Increase the values of weights $\tilde{c}_j^{\text{out}}(\kappa)$ based on the number of continuous variables the corresponding trains have in the other MILP problems.

The first option is to simply choose a factor with which to increase all weights $\tilde{c}_j^{\text{out}}(\kappa)$. Our choice is to double the weights. The reason for this is that every continuous variable corresponding to these weights is connected to at least one continuous variable in one of the other MILP problems, and the dependency between the delays of these two continuous variables is very high. We call this method "DMPC method 3".

For the second option we need to determine for every train leaving the area which events are used to model this train in the other areas. This corresponds to the continuous variables that are connected through continuity and running time constraints to the variables $\tilde{z}_j^{\text{out}}(\kappa)$. Denote $\tilde{z}_{j,k}^{\text{out}}(\kappa)$ as the k-th element of vector $\tilde{z}_j^{\text{out}}(\kappa)$ and denote the set containing the continuous variables in the other subproblems connected to $\tilde{z}_{j,k}^{\text{out}}(\kappa)$ through continuity and running time constraints by $\mathcal{Z}_{j,k}^{\text{out}}(\kappa)$. For every variable in the vector $\tilde{z}_j^{\text{out}}(\kappa)$ the weight $\tilde{c}_{j,k}^{\text{out}}(\kappa)$ is then set to:

$$\tilde{c}_{j,k}^{\text{out}}(\kappa) = \tilde{c}_{j,k}^{\text{out}}(\kappa) + \sum_{l:z_l(\kappa)\in\mathcal{Z}_{j,k}^{\text{out}}(\kappa)} c_l(\kappa), \tag{31}$$

which is the sum of the weights of the standard cost function of the continuous variables in $\mathcal{Z}_{j,k}^{\text{out}}(\kappa)$ plus the weight on $\tilde{z}_{j,k}^{\text{out}}(\kappa)$. We call this method "DMPC method 4". In the next section we will show what effects these changes to the cost function have on the quality of the solution found with the DMPC methods and what influence they have on the computation time.

4.4 Summary of the DMPC methods

We will compare the MPC and the four DMPC methods based on whether they can reorder the trains *local*ly (only in 1 part of the network) or *global*ly (the entire network), if they can reschedule the trains *local*ly (only in the part of the network where they can also reorder) or *global*ly (in the entire network, as long as they satisfy the given order of the trains if it is fixed), and whether or not the weights on the trains leaving the controlled part of the network have their *default* value, increased by a *factor*, or *relative* to the number of continuous variables. The comparison is given in Table 1

(*Place Table 1 about here*)

5 Case Study

In this section we compare the performance of the centralized MPC method to the four DMPC methods. We use a model of the railway network that is based on the Dutch railway network with the timetable of 2011[4]. The train lines, their line type, and their frequencies are shown in Appendix A. The cost function of the centralized MPC is the sum of all delays with a small weight on the binary variables: $c(\kappa)^\top z(\kappa) =$

[4]The complete timetable is too large to include in this paper and is no longer available online. Since there have been very few major changes in the timetable in the last years the reader can get a general idea of the timetable from the 2015 timetable. The timetable of 2015 can be found (in Dutch) at http://www.ns.nl/reizigers/reisinformatie/informatie/informatie-tijdens-uw-reis/download-dienstregeling-2014-2015.html.

$[\mathbf{1}^{1\times n_x(\kappa)}\ \ 0.0001\cdot\mathbf{1}^{1\times n_{\mathcal{U}}(\kappa)}]\begin{bmatrix}x(\kappa)\\ \mathcal{U}(\kappa)\end{bmatrix}$, where $\mathbf{1}^{1\times m}$ is a 1 by m vector containing only ones, $n_x(\kappa)$ is the number of continuous variables at time instant $t(\kappa)$, and $n_{\mathcal{U}}$ is the number of binary variables at time instant $t(\kappa)$. The number of trains, constraints, and continuous and binary variables for different time windows of the timetable of the complete constraint set are given in Table 2.

(*Place Table 2 about here*)

The personal computer we use has these specifications: AMD Phenom II X4 960T at 3.00GHz with 16GB memory running 64bit Windows 7 with Matlab 2013b and we have solved the MILP optimization problems with Gurobi 5.6.0.

Using the method described in Section 3 we have determined partitions of 2, 3, 4, 6, and 8 parts for the DMPC methods using a value of $\rho = 0.005$. The partitions can be seen in Figure 1 (b)-(f) and the number of constraints, and binary and continuous variables per constraint set or part are given in Table 3.

(*Place Figure 1 about here*) (*Place Table 3 about here*)

100 scenarios were generated, in each scenario the scheduled arrival and departure times for a 3 hour window were considered. Depending on the delays between 180 and 200 optimizations are done per scenario. We delay 10% of the trains with a randomly generated delay according to a Weibull (Weibull, 1951, Yuan, 2006) distribution with scale parameter 5 and shape parameter 0.8, resulting in a delay scenario comparable to an average day in The Netherlands (ProRail).

At the start there are no delays. As a result during the first 30 minutes the amount of delays increases, after that it stabilizes. Therefore, we do not consider the computation times of the optimizations for the first 30 minutes. Depending on the length of the prediction horizon the size of the optimization problems decreases near the end of the 3 hour window, and therefore we only consider the computation times up to the 130th minute, leaving us with 100 computation times for each scenario, or 10000 computation times in total, for each method, and for each partition.

The controllers have to update the schedule every minute and are operating in a closed loop such that at every time instant new information is gathered from the simulation model and new delays are "detected" (or introduced) and the controllers have to compute a new schedule at every time instant. Once the new schedule has been determined only the dispatching actions that apply to the events that will occur in the minute after the current time instant are implemented. For each scenario we have solved the rescheduling problem for the centralized MPC method and the four DMPC methods for the five different partitions for prediction horizons of length 30, 45, 60, and 75 minutes.

For methods based on MPC at each step the solution, or control input, should be determined and implemented well before the new information is received and the next step starts. In this case study we assume we get new information every minute and therefore the schedule is recomputed every minute. Because of that the computation time should be well below one minute, an acceptable maximum computation time would be 20 seconds.

5.1 Computation times

First consider the computation times of the centralized MPC method as given in Figure 2. Most of the time the new schedule is computed well within 20 seconds, but for the prediction horizons of 45, 60, and 75 minutes, some computation times are already above 60 seconds with a maximum of 158 seconds for the prediction horizon of 45 minutes. For a prediction horizon of 60 minutes this increases to 736 seconds and for a prediction horizon of 75 minutes the maximum becomes 1009 seconds. Because of the maximum computation time the implementations of the centralized MPC with a prediction horizon of 45, 60, and 75

minutes are currently not suitable for on-line railway traffic management.

(*Place Figure 2 about here*)

Next consider the computation times of DMPC method 1 as shown in Figure 3 and given in Table 4. For the prediction horizons of 30 minutes increasing the number of parts in the partition above 3 parts only increases the computation time. This is because solving the optimization problem of the centralized MPC method is already relatively easy; therefore, the computation time for each subproblem of DMPC method 1 is only slightly shorter than the centralized MPC method and with an increased number of parts in the partition the number of iterations the DMPC method needs to perform increases, resulting in an increase in computation time instead of a decrease. Since the optimization problem becomes harder to solve for larger prediction horizons, the number of parts for which the computation time still reduces is higher for larger prediction horizons. Only for prediction horizons of 60 and 75 minutes with a partitioning into 2 parts the maximum computation time goes above 60 seconds, with a maximum of 67 seconds for a prediction horizon of 60 minutes and 2 parts, and 103 seconds for a prediction horizon of 75 minutes and 2 parts. For a prediction horizon of 75 minutes the maximum computation time is above 20 seconds for all partitions. For a prediction horizon of 60 minutes only the partition with 2 parts has a maximum computation time above 20 seconds. For the prediction horizons of 30 and 45 minutes the maximum computation time is below 20 seconds for all partitions.

(*Place Figure 3 about here*) (*Place Table 4 about here*)

Since DMPC methods 2, 3, and 4 are nearly identical methods, only their cost function is different, their computation times are also similar, as shown in Figures 4-6 respectively. Compared to the computation times of DMPC method 1 there are clear differences. For DMPC methods 2, 3, and 4 increasing the number of parts in the partition decreases the computation time for all lengths of the prediction horizon tested and all partitions, except for DMPC 4, with a prediction horizon of 60 minutes and 8 parts in the partition and with a prediction horizon of 75 minutes and 4 parts in the partition. In both of these cases the maximum computation time is slightly higher than for some of the other partitions with less parts. The cause of the decrease in computation time is that each subproblem only contains of a part of the constraints of the centralized problem with no overlap in constraints of the subproblems. Thus increasing the number of subproblems reduces the number of constraints of each subproblem. This is in contrast with the subproblems of DMPC method 1, where each subproblem contains all constraints of the centralized problem. Therefore, increasing the number of subproblems does not reduce the number of the constraints of each subproblem.

For DMPC method 2 the maximum computation time is above 60 seconds in only two cases: for a prediction horizon of 60 minutes and the partition into 2 parts with a maximum of 65 seconds, and for a prediction horizon of 75 minutes and the partition into 2 parts with a maximum of 118 seconds. There is only one case where the maximum computation time is between 20 and 60 seconds: a prediction horizon of 75 minutes and the partition in 3 parts with a maximum of 24 seconds. All other combinations of partitions and prediction horizon lengths have a maximum computation time below 20 seconds and can be used for on-line railway traffic management.

(*Place Figure 4 about here*)

For DMPC method 3 the maximum computation time is above 60 seconds in only one case: A prediction horizon of 60 minutes and the partition in 2 parts with a maximum of 120 seconds. There are two cases where the computation time is between 20 and 60 seconds: A prediction horizon of 75 minutes and the partitions in 2 and 3 parts with maximum computation times of 44 and 24 seconds respectively. All other combinations of prediction horizon lengths and partitions are below 20 seconds and therefore suitable for on-line railway traffic management. It is unexpected that the maximum computation time of the prediction horizon of 75 minutes with the partition in 2 parts is lower than the maximum computation time with the prediction horizon of 60 minutes and the partition in 2 parts. The longer prediction horizon in general makes the problems harder to solve, but even though we ran 10000 simulations for each of the possible combinations of prediction horizon length and number of subproblems the maximum computation time may decrease when

increasing the prediction horizon due to the binary nature of the problems. Therefore, no strict conclusions can be drawn about the maximum computation time, which is some cases may still exceed the computation times we have found. We can only say that it is highly unlikely that the maximum computation time is higher than the maximum we have determined under the similar delay scenarios.

(*Place Figure 5 about here*)

Finally for DMPC method 4 there are again two cases where the maximum computation time is above 60 seconds: the prediction horizon of 60 seconds with the partition in 2 parts with a maximum computation time of 76 seconds and for the prediction horizon of 75 minutes with the partition in 2 parts with a maximum computation time of 70 seconds. Only for the case of the prediction horizon of 75 minutes and the partition in 4 parts the maximum computation time is between 20 and 60 seconds. For all other cases the maximum computation time is below 20 seconds.

(*Place Figure 6 about here*)

Next we will discuss the delay reduction achieved with the various (D)MPC methods proposed.

5.2 Delay reduction

In order to determine the delay reduction achieved with the various control methods we compare the total delay (in delay minutes) of the 100 scenarios combined for all methods, prediction horizon lengths, and partitions to the nominal case. In the nominal case no train orders are changed: only the arrival and departure times are changed to satisfy the headway constraints. We determine the reduction in delays compared to the nominal case for all cases in percent of the total delay (delay minutes, not the number of delayed trains) of the nominal case. For the centralized MPC approach the delay reduction is given in the bar plot in Figure 7.

(*Place Figure 7 about here*)

For the DMPC approaches the delay reduction is given in the bar plots in Figure 8 for the various prediction horizon lengths and partitions.

(*Place Figure 8 about here*)

All the data is also given in Table 5.

(*Place Table 5 about here*)

It is clear that DMPC method 4 always outperforms methods 2 and 3 in terms of delay reduction. Recall that the difference in computation time between DMPC methods 2, 3, and 4 is negligible. Therefore, we will not consider DMPC methods 2 and 3 in the rest of the discussion.

In Figure 9 the delay reduction is plotted against the maximum computation for DMPC methods 1 and 4 and the centralized MPC method. Ideally the (D)MPC methods have a very high delay reduction and only require a very short computation time. Such methods would be found in the upper left part of the figure. The methods with the highest reduction in delays for the given computation time are connected via the dashed line. When we try to minimize the computation time and maximize the delay reduction, then these are the Pareto optimal solutions.

As is clear from this figure, the DMPC methods can achieve a similar delay reduction in a much less time compared to the centralized MPC method. In most cases DMPC method 4 is even slightly faster than DMPC 1.

(*Place Figure 9 about here*)

Furthermore it is clear that by increasing the prediction horizon the delays are reduced more. The effects of increasing the prediction horizon further diminishes when the prediction horizon is longer: from 30 to 45 minutes the delay reduction for the centralized MPC is improved from 17.9% to 20.0%, from 45 to 60 minutes it is already slightly lower with an improvement to 21.6%, and from 60 to 75 minutes the delay reduction is only increased to 21.9%, an 0.3% increase. For the DMPC approaches we see similar results. An explanation for this is that the further ahead the controller predicts the future arrival and departure times, the less accurate those departure and arrival times become. As a result most of the future arrival and departure times will be changed at future time instants when new information becomes available and better predictions can be made. Furthermore, many trains reach their final destination or turning station within 75 minutes, and at the turning station the trains usually have more buffer time, so any delay they still have at that point is almost completely reduced to zero. As a result the delays do not propagate much further in time than 60 to 75 minutes.

When we only consider the delay reduction and not the computation time, the centralized MPC with a prediction horizon of 75 minutes reduces the delays the most: 21.9%. Of the DMPC methods, DMPC method 4 with a prediction horizon of 75 minutes and partitioned into 2 parts has the highest delay reduction: 21.8%. The difference in delay reduction between the two is almost negligible, but the difference in computation time is substantial; the average computation time needed for DMPC method is 3.38 times lower than for the centralized MPC method and the maximum computation time is 14.5 times lower.

When we consider a limit on the maximum computation time of 20 seconds DMPC method 1 with a prediction horizon of 60 minutes and partitioned into 4 parts and DMPC method 4 with a prediction horizon of 75 minutes and partitioned into 3 parts have the best results. DMPC method 4 has a slightly lower average computation time, but a higher maximum computation time. Both methods achieve a 21.4% reduction in delays, which is the highest of all methods with a maximum computation time below 20 seconds. In most cases DMPC 4 is slightly faster and achieves the same delay reduction and therefore DMPC method 4 is recommended.

6 Discussion and conclusions

One of the remaining challenges for on-line railway traffic management is global control, where a new schedule is determined for the entire network, instead of a small part of the network. The main challenge is the computation time needed to find a new and improved schedule. In this paper we have introduced four distributed model predictive control (DMPC) approaches to solve the problem of global on-line rescheduling of railway traffic. With the use of a distributed approach a new schedule for the entire railway network can be found much faster than with a centralized approach. To determine how the network should be split up for the distributed approach we introduced an optimization-based off-line method that partitions the centralized rescheduling problem into a set of subproblems that can be used in the DMPC approaches.

To test the proposed DMPC approaches an extensive case studies was performed. In the case study a model of the Dutch railway network containing all Nederlandse Spoorwegen (Dutch Railways) trains as described in the timetable of 2011 was used. The centralized problem was partitioned into 2, 3, 4, 6, and 8 parts for the DMPC approaches and tested for prediction horizons of 30, 45, 60, and 75 minutes. The performance of the controllers was compared for a three-hour window, where the controller was in a closed loop. Each minute a new optimization had to be solved and implemented. The total delay in this three hour window was compared for all four approaches, for all five partitions, and four prediction horizon lengths, for 100 scenarios.

With the MPC method the delay can be reduced the most, but the increased delay reduction comes at a steep price in terms of computation time, requiring up to 1000 seconds to find the optimal solution. Furthermore

it is clear that increasing the prediction horizon further than 60 minutes only yields minor improvements while the computation time does increase steadily. DMPC1 method requires much less time to find its best solution. At most 103 seconds. The decreased computation time comes at a small price in terms of delay reduction. For DMPC1 increasing the number of parts of the partition does not improve the computation time much. For the large prediction horizons the worst case computation time decreased when partitioning the network in up to 6 parts, but on average the computation time does not improve. For DMPC2-4 on average the computation time is even better than of DMPC1, and increasing the number of parts does improves the computation time even up to 8 parts. The delay reduction is a little worse than DMPC1. So for the highest delay reduction the MPC method is the best. DMPC2-4 require the least computation time to find their optimal solution. DMPC1 is slightly slower than DMPC2-4 but has a slightly higher delay reduction. In most cases either DMPC1 or DMPC4 are recommended, but if computation time is no issue at all then MPC will yield the best results.

In the case study and the rest of the paper we only considered cost functions that represent a measure for the delays of trains, but for the passengers it would be better to focus on the delays the passengers incur. To achieve this extensive information on the passengers and their behavior would be needed. With the recent introduction of smart cards to pay for the railway tickets, railway operators have gained a new source of information to estimate the number of passengers in the trains and several researchers have been working on using this data to estimate the passenger flows (Asakura et al., 2012, Kusakabe et al., 2010, van der Hurk et al., 2015). With this research the cost functions can be changed from a measure of train delays to a measure of passenger delays. As future research changing the cost function from a measure of train delays to a measure of passenger delays would be very interesting.

The methods and models we used in the method assume there are only disturbances in the network and no disruptions, but in reality disruptions happen on a nearly daily basis. We would therefore like to extend our research into disruption management as well.

Acknowledgments

This research is supported by the Dutch Technology Foundation STW, project 11025 “Model-Predictive Railway Traffic Management; A Framework for Closed-Loop Control of Large-Scale Railway Systems”. STW is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs.

References

Y. Asakura, T. Iryo, Y. Nakajima, and T. Kusakabe, 2012. Estimation of behavioural change of railway passengers using smart card data. Public Transport, 4(1),1–16.

V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf, and J. Wagenaar, 2014. An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological, 63,15–37.

G. Caimi, M. Fuchsberger, M. Laumanns, and M. Lüthi, 2012. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. Computers and Operations Research, 39(11),2578–2593.

E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, 2002. Distributed model predictive control. IEEE Control Systems, 22(1),44–52.

F. Corman and L. Meng, 2015. A review of online dynamic models and algorithms for railway traffic management. IEEE Transactions on Intelligent Transportation Systems, 16(3),1274–1284.

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

F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, 2010b. Centralized versus distributed systems to reschedule trains in two dispatching areas. Public Transport, 2(3),219–247.

F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, 2012a. Optimal inter-area coordination of train rescheduling decisions. Transportation Research Part E: Logistics and Transportation Review, 48(1), 71–88.

F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, 2012b. Bi-objective conflict detection and resolution in railway traffic management. Transportation Research Part C: Emerging Technologies, 20(1),79–94.

F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, 2014. Dispatching and coordination in multi-area railway traffic management. Computers & Operations Research, 44(1),146–160.

C. R. Cutler and B. L. Ramaker, 1980. Dynamic matrix control – a computer control algorithm. In Proceedings of the Joint Automatic Control Conference, San Francisco, California, 1980.

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

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

B. De Schutter and T. van den Boom, 2001. Model predictive control for max-plus-linear discrete event systems. Automatica, 37(7),1049–1056.

B. De Schutter and T. van den Boom, 2003. MPC for discrete-event systems with soft and hard synchronisation constraints. International Journal of Control, 76(1),82–94.

W. Dunbar, 2007. Distributed receding horizon control of dynamically coupled nonlinear systems. IEEE Transactions on Automatic Control, 52(7),1249–1263.

W. Fang, S. Yang, and X. Yao, 2015. A survey on problem models and solution approaches to rescheduling in railway networks. IEEE Transactions on Intelligent Transportation Systems, pages 1–20.

M. Farina and R. Scattolini, 2012. Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems. Automatica, 48(6),1088–1096.

C. E. García, D. M. Prett, and M. Morari, 1989. Model predictive control: Theory and practice - a survey. Automatica, 25(3),335–348.

S. Kanai, K. Shiina, S. Harada, and N. Tomii, 2011. An optimal delay management algorithm from passengers' viewpoints considering the whole railway network. Journal of Rail Transport Planning & Management, 1(1),25–37.

P. Kecman, F. Corman, A. D'Ariano, and R. M. Goverde, 2013. Rescheduling models for railway traffic management in large-scale networks. Public Transport, 5(1-2),95–123.

B. Kersbergen, J. Rudan, T. van den Boom, and B. De Schutter, 2014a. Towards railway traffic management using switching max-plus-linear systems. Discrete Event Dynamic Systems, pages 1–41.

B. Kersbergen, T. J. J. van den Boom, and B. De Schutter, 2014b. Distributed model predictive control for rescheduling of railway traffic. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC2014), pages 2732–2737, China, October 2014.

B. Kersbergen, T. J. J. van den Boom, and B. De Schutter, 2015. Improved distributed model predictive control for rescheduling of railway traffic by manipulation of the cost functions. In Proceedings of the 6th International Seminar on Railway Operations Modelling and Analysis (RailTokyo), Japan, March 2015. Paper 25.

T. Kusakabe, T. Iryo, and Y. Asakura, 2010. Estimation method for railway passengers' train choice behavior with smart card transaction data. Transportation, 37(5),731–749.

J. M. Maestre and R. R. Negenborn. Distributed Model Predictive Control Made Easy. Springer Publishing Company, 2013.

L. Meng and X. Zhou, 2014. Simultaneous train rerouting and rescheduling on an n-track network: A model reformulation with network-based cumulative flow variables. Transportation Research Part B: Methodological, 67,208–234.

M. Morari and J. H. Lee, 1999. Model predictive control: past, present and future. Computers & Chemical Engineering, 23(4–5),667–682.

P. Murali, F. Ordóñez, and M. M. Dessouky, 2015. Modeling strategies for effectively routing freight trains through complex networks. Transportation Research Part C: Emerging Technologies, pages 1–17.

P. Pellegrini, G. Marlière, and J. Rodriguez, 2014. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. Transportation Research Part B: Methodological, 59,58–80.

ProRail. Prorail jaarverslag 2014, 2014. URL `http://www.jaarverslagprorail.nl/`. (In Dutch).

S. Qin and T. A. Badgwell, 2003. A survey of industrial model predictive control technology. Control Engineering Practice, 11(7),733–764.

E. Quaglietta, 2011. A microscopic simulation model for supporting the design of railway systems: development and applications. PhD thesis, University of Naples Federico II, Naples, Italy.

E. Quaglietta, F. Corman, and R. M. P. Goverde, 2013. Impact of a stochastic and dynamic setting on the stability of railway dispatching solutions. In 16th International IEEE Conference on Intelligent Transportation Systems, pages 1035–1040, 2013.

J. Richalet, A. Rault, J. Testud, and J. Papon, 1978. Model predictive heuristic control: Applications to industrial processes. Automatica, 14(5),413–428.

A. Richards and J. P. How, 2007. Robust distributed model predictive control. International Journal of Control, 80(9),1517–1531.

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

J. Törnquist, 2007. Railway traffic disturbance management – an experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. Transportation Research Part A: Policy and Practice, 41(3),249–266.

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

J. Törnquist-Krasemann, 2012. Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. Transportation Research Part C: Emerging Technologies, 20(1),62–78.

E. van der Hurk, L. Kroon, G. Maroti, and P. Vervest, 2015. Deduction of passengers' route choices from smart card data. IEEE Transactions on Intelligent Transportation Systems, 16(1),430–440.

W. Weibull, 1951. A statistical distribution function of wide applicability. Journal of Applied Mechanics, 18,293–297.

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

A Train lines and their frequencies

The train lines and their frequencies are shown in Table 6. Each line is in both directions and the frequency is per direction.

(*Place Table 6 about here*)

List of Table captions:

1. Comparison of the properties of the proposed control methods.
2. Number of trains, constraints, and continuous and binary variables for 30, 45, 60, and 75 minutes of the timetable.
3. Number of constraints, and continuous and binary variables for one hour of the timetable per constraint set.
4. Computation time in seconds of the MPC and DMPC methods for the case study. The first row indicates the length of the prediction horizon. The number of parts the network was divided into for the DMPC methods is given in brackets after the name of the DMPC method. The average (avg), minimum (min), and maximum (max) computation times are given.
5. Delay reduction in % for the centralized MPC and the DMPC methods. The number of parts the network was divided into for the DMPC methods is given in brackets after the name of the DMPC method.
6. Train lines and their frequencies for the 2011 timetable of the Dutch Network. InterCity (IC) trains are interregional trains and Sprinters (SP) are local trains.

List of Figure captions:

1. Model of the Dutch Network (a), and the partitions used for DMPC, partitioned in 2 (b), 3 (c), 4 (d), 6 (e), and 8 (f) parts. Each part is marked with a different line style.
2. Computation time needed for the centralized model predictive control approach with a prediction horizon of 30, 45, 60, and 75 minutes.
3. Computation time needed for DMPC method 1 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".
4. Computation time needed for DMPC method 2 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".
5. Computation time needed for DMPC method 3 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".
6. Computation time needed for DMPC method 4 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".
7. Reduction of the delays in percentage compared to the nominal case for the centralized MPC approach for the case study.
8. Reduction of the delays in percentage compared to the nominal case for (a) DMPC approach 1, (b) DMPC approach 2, (c) DMPC approach 3, and (d) DMPC approach 4 for the partitions in 2, 3, 4, 6, and 8 parts.
9. Reduction of the delays in percentage plotted against the maximum computation for DMPC method 1, DMPC method 4, and the MPC method for the partitions in 2, 3, 4, 6, and 8 parts, and the prediction horizons of 30, 45, 60, and 75 minutes. The dashed line shows the Pareto frontier connecting the Pareto optimal methods.

Table 1: Comparison of the properties of the proposed control methods.

Method	Reorder	Reschedule	Border weights
MPC	Global	Global	Default
DMPC1	Local	Global	Default
DMPC2	Local	Local	Default
DMPC3	Local	Local	Factor
DMPC4	Local	Local	Relative

Table 2: Number of trains, constraints, and continuous and binary variables for 30, 45, 60, and 75 minutes of the timetable.

Horizon	30	45	60	75
Trains	163	244	326	408
Continuous variables	960	1430	1930	2401
Binary variables	559	1457	2744	4325
Constraints	9397	15275	22050	29094

Table 3: Number of constraints, and continuous and binary variables for one hour of the timetable per constraint set.

Parts									
2	Continuous variables	924	1006						
	Binary variables	1420	1324						
	Constraints	11154	10896						
3	Continuous variables	774	580	576					
	Binary variables	876	894	974					
	Constraints	7426	7016	7608					
4	Continuous variables	476	448	504	502				
	Binary variables	714	706	652	672				
	Constraints	5632	5522	5360	5536				
6	Continuous variables	416	252	312	264	358	328		
	Binary variables	470	460	492	482	406	434		
	Constraints	3972	3492	3912	3696	3454	3524		
8	Continuous variables	224	316	214	308	292	168	256	152
	Binary variables	360	362	310	338	380	314	336	344
	Constraints	2800	3050	2524	2956	3096	2364	2732	2528

Table 4: Computation time in seconds of the MPC and DMPC methods for the case study. The first row indicates the length of the prediction horizon. The number of parts the network was divided into for the DMPC methods is given in brackets after the name of the DMPC method. The average (avg), minimum (min), and maximum (max) computation times are given.

	30 min			45 min			60 min			75 min		
	avg	min	max	avg	min	max	avg	min	max	avg	min	max
MPC	0.31	0.12	4.34	1.93	0.30	158	5.60	0.48	736	11.8	0.75	1009
DMPC1 (2)	0.41	0.19	1.79	1.24	0.45	11.8	2.88	0.90	67.4	5.50	1.16	103
DMPC1 (3)	0.53	0.30	1.34	1.46	0.62	5.05	2.78	1.06	18.5	4.73	1.22	52.4
DMPC1 (4)	0.66	0.35	1.39	1.65	0.81	4.71	3.01	1.18	11.3	5.22	1.57	47.5
DMPC1 (6)	0.89	0.52	1.72	2.15	1.18	5.44	4.17	1.99	14.5	6.20	2.34	26.8
DMPC1 (8)	1.12	0.69	2.85	2.54	1.36	6.92	4.80	2.42	15.1	7.32	2.86	32.0
DMPC2 (2)	0.23	0.09	1.06	0.86	0.23	12.4	1.99	0.38	64.9	3.47	0.54	118
DMPC2 (3)	0.20	0.08	0.56	0.70	0.22	3.20	1.43	0.37	12.2	2.24	0.44	28.1
DMPC2 (4)	0.19	0.08	0.46	0.53	0.18	2.40	1.03	0.31	5.57	1.70	0.35	13.0
DMPC2 (6)	0.17	0.08	0.40	0.45	0.17	1.63	0.78	0.25	4.95	1.20	0.34	7.54
DMPC2 (8)	0.16	0.08	0.38	0.38	0.15	1.05	0.64	0.23	2.38	0.98	0.25	5.84
DMPC3 (2)	0.23	0.08	1.02	0.85	0.23	16.5	1.99	0.46	120	3.42	0.54	43.9
DMPC3 (3)	0.20	0.08	0.93	0.70	0.22	3.24	1.47	0.37	13.0	2.32	0.44	24.5
DMPC3 (4)	0.19	0.08	0.58	0.53	0.18	2.03	1.06	0.31	5.59	1.75	0.40	11.3
DMPC3 (6)	0.17	0.08	0.56	0.45	0.17	1.79	0.79	0.25	3.41	1.22	0.31	9.24
DMPC3 (8)	0.16	0.08	0.36	0.37	0.15	1.44	0.64	0.23	3.36	0.97	0.24	7.37
DMPC4 (2)	0.23	0.08	1.17	0.86	0.23	12.6	2.06	0.46	75.6	3.49	0.54	69.6
DMPC4 (3)	0.20	0.08	0.90	0.67	0.22	4.89	1.47	0.37	10.6	2.39	0.45	19.0
DMPC4 (4)	0.19	0.08	0.49	0.53	0.18	2.74	1.09	0.28	6.07	1.82	0.32	21.7
DMPC4 (6)	0.17	0.08	0.57	0.44	0.16	2.58	0.79	0.25	5.48	1.25	0.33	10.5
DMPC4 (8)	0.16	0.08	0.34	0.37	0.15	1.06	0.63	0.23	9.82	0.94	0.24	5.81

Table 5: Delay reduction in % for the centralized MPC and the DMPC methods. The number of parts the network was divided into for the DMPC methods is given in brackets after the name of the DMPC method.

	30 min	45 min	60 min	75 min
MPC	17.9%	20.0%	21.6%	21.9%
DMPC 1 (2)	17.9%	20.2%	21.5%	21.7%
DMPC 1 (3)	17.7%	19.9%	21.2%	21.4%
DMPC 1 (4)	17.9%	20.0%	21.4%	21.6%
DMPC 1 (6)	17.6%	19.6%	21.0%	21.1%
DMPC 1 (8)	18.0%	20.0%	21.2%	21.4%
DMPC 2 (2)	17.4%	19.2%	20.6%	20.8%
DMPC 2 (3)	17.1%	17.8%	20.1%	20.2%
DMPC 2 (4)	16.1%	17.4%	18.6%	18.5%
DMPC 2 (6)	16.1%	16.7%	18.9%	19.0%
DMPC 2 (8)	16.3%	17.2%	19.2%	19.2%
DMPC 3 (2)	17.8%	19.9%	21.2%	21.4%
DMPC 3 (3)	17.7%	18.5%	20.7%	20.8%
DMPC 3 (4)	17.2%	18.9%	20.0%	20.0%
DMPC 3 (6)	17.0%	17.8%	19.8%	19.8%
DMPC 3 (8)	16.9%	18.2%	20.1%	20.2%
DMPC 4 (2)	17.9%	20.2%	21.6%	21.8%
DMPC 4 (3)	17.8%	19.9%	21.3%	21.4%
DMPC 4 (4)	17.8%	20.1%	21.2%	21.4%
DMPC 4 (6)	17.7%	19.6%	21.1%	21.1%
DMPC 4 (8)	17.7%	19.9%	21.0%	21.0%

Table 6: Train lines and their frequencies for the 2011 timetable of the Dutch Network. InterCity (IC) trains are interregional trains and Sprinters (SP) are local trains.

Train line	**Train type**	**Frequency**
Alkmaar - Maastricht CS	IC	every 30 minutes
Arnhem - Ede-Wageningen	SP	every 60 minutes
Arnhem - Zutphen	SP	every 30 minutes
Amsterdam CS - Almere Oostvaarders	SP	every 30 minutes
Amsterdam CS - Amersfoort	SP	every 30 minutes
Amsterdam CS - Breda	SP	every 30 minutes
Amsterdam CS - Den Haag CS	IC	every 30 minutes
Amsterdam CS - Dordrecht	IC	every 60 minutes
Amsterdam CS - Haarlem	SP	every 30 minutes
Amsterdam CS - Hoofddorp	SP	every 30 minutes
Amsterdam CS - Lelystad Centrum	IC	every 30 minutes
Amsterdam CS - Rotterdam CS	IC	every 60 minutes
Amsterdam CS - Roosendaal	IC	every 60 minutes
Amsterdam CS - Uitgeest (Haarlem)	SP	every 30 minutes
Amsterdam CS - Uitgeest (Zaandam)	SP	every 30 minutes
Amsterdam CS - Vlissingen	IC	every 60 minutes
Apeldoorn - Enschede	SP	every 30 minutes
Breukelen - Rhenen	SP	every 30 minutes
Den Haag CS - Breda	SP	every 30 minutes
Den Haag CS - Enschede	IC	every 60 minutes
Den Haag CS - Groningen	IC	every 60 minutes
Den Haag CS - Gouda Goverwelle	SP	every 30 minutes
Den Haag CS - Haarlem	SP	every 30 minutes
Den Haag CS - Hoorn	SP	every 30 minutes
Den Haag CS - Lelystad Centrum	IC	every 30 minutes
Den Haag CS - Lelystad Centrum	SP	every 30 minutes
Den Haag CS - Utrecht CS	IC	every 30 minutes
Den Haag CS - Utrecht CS	SP	every 30 minutes
Den Haag CS - Venlo	IC	every 30 minutes
Den Haag CS - Roosendaal	SP	every 30 minutes
Den Helder - Nijmegen	IC	every 30 minutes
Enkhuizen - Amersfoort	IC	every 30 minutes
Eindhoven - Weert	SP	every 30 minutes
Eindhoven - Tilburg	SP	every 30 minutes
Groningen - Zwolle	IC	every 60 minutes
's Hertogenbosch - Nijmegen	SP	every 30 minutes
's Hertogenbosch - Deurne	SP	every 30 minutes
Hoorn - Hoofddorp	SP	every 30 minutes
Roermond - Maastricht Randwyck	SP	every 30 minutes
Roosendaal - Vlissingen	IC	every 60 minutes
Roosendaal - Zwolle	SP	every 30 minutes
Rotterdam CS - Amersfoort	IC	every 30 minutes
Rotterdam CS - Deventer	IC	every 60 minutes
Rotterdam CS - Leeuwarden	IC	every 60 minutes
Rotterdam CS - Uitgeest	SP	every 30 minutes
Schiphol - Groningen	IC	every 60 minutes
Schiphol - Eindhoven	IC	every 30 minutes
Schiphol - Enschede	IC	every 60 minutes

Continued on next page

Table 6 – continued from previous page

Train line	Train type	Frequency
Schiphol - Leeuwarden	IC	every 60 minutes
Schiphol - Nijmegen	IC	every 30 minutes
Sittard - Heerlen	IC	every 30 minutes
Sittard - Heerlen	SP	every 30 minutes
Utrecht CS - Almere Centrum	IC	every 30 minutes
Utrecht CS - Breda	SP	every 30 minutes
Utrecht CS - Breukelen	SP	every 30 minutes
Utrecht CS - Hoofddorp	SP	every 30 minutes
Utrecht CS - Tiel	SP	every 30 minutes
Utrecht CS - Zwolle	SP	every 30 minutes

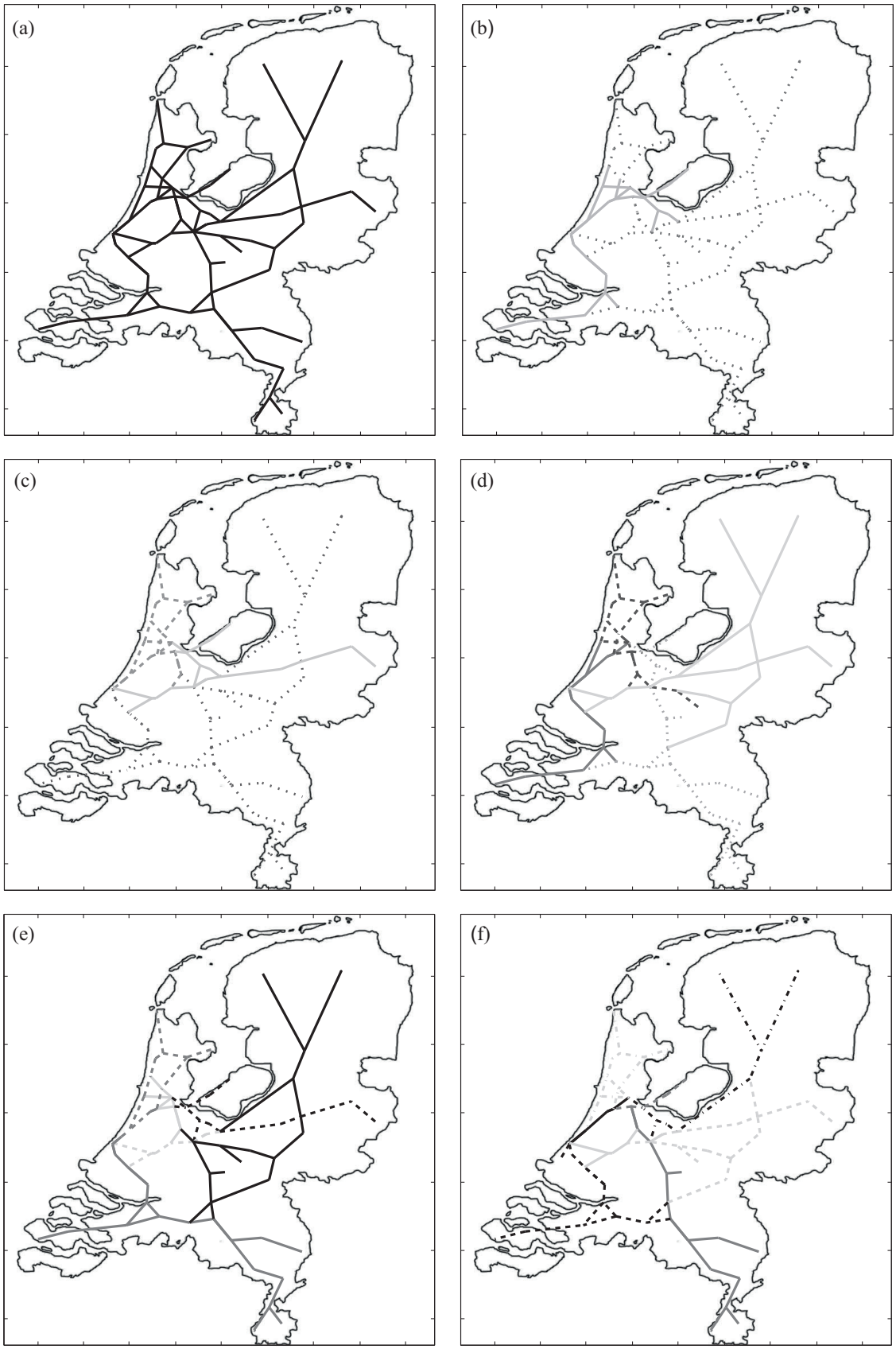

Figure 1: Model of the Dutch Network (a), and the partitions used for DMPC, partitioned in 2 (b), 3 (c), 4 (d), 6 (e), and 8 (f) parts. Each part is marked with a different line style.

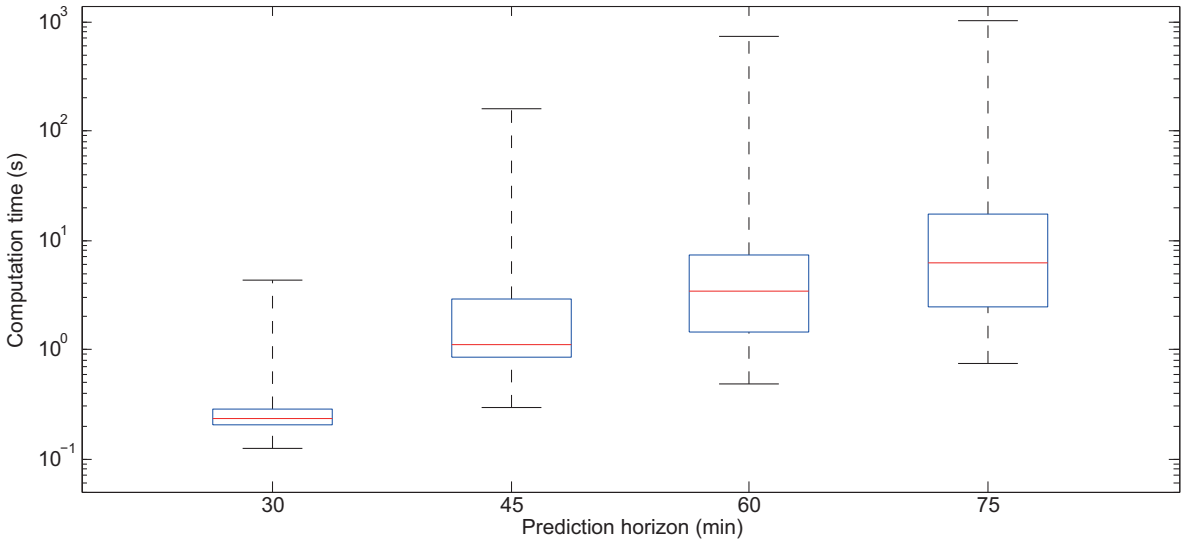

Figure 2: Computation time needed for the centralized model predictive control approach with a prediction horizon of 30, 45, 60, and 75 minutes.

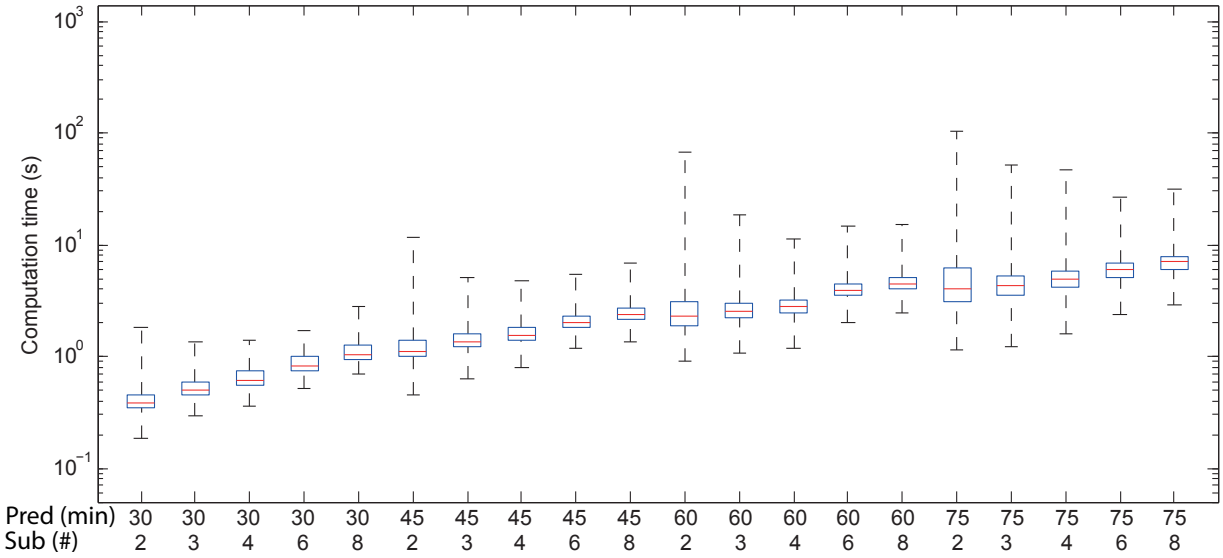

Figure 3: Computation time needed for DMPC method 1 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".

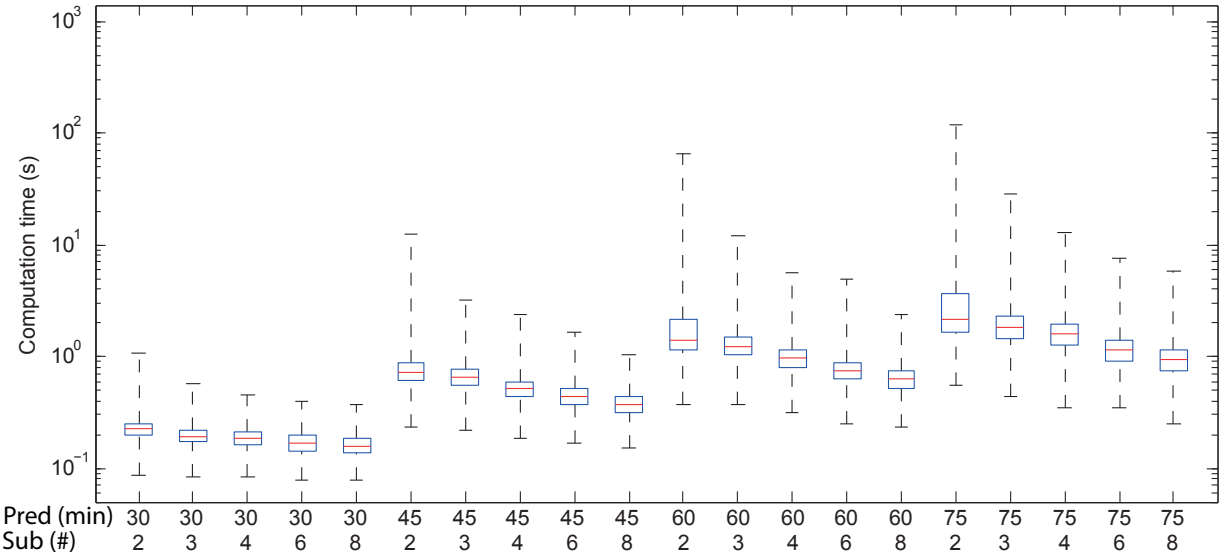

Figure 4: Computation time needed for DMPC method 2 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".

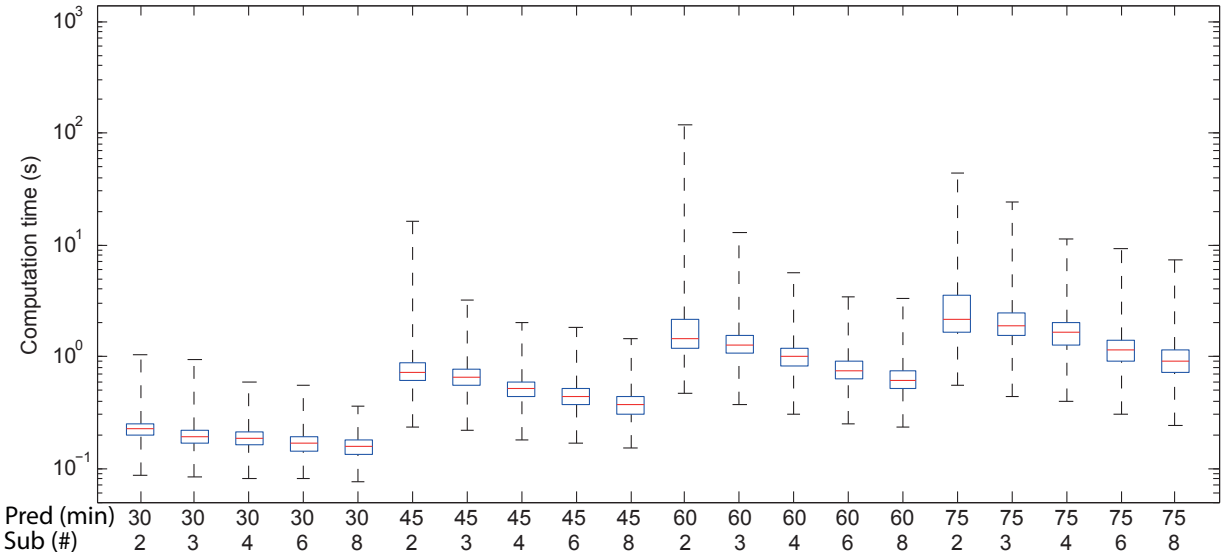

Figure 5: Computation time needed for DMPC method 3 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".

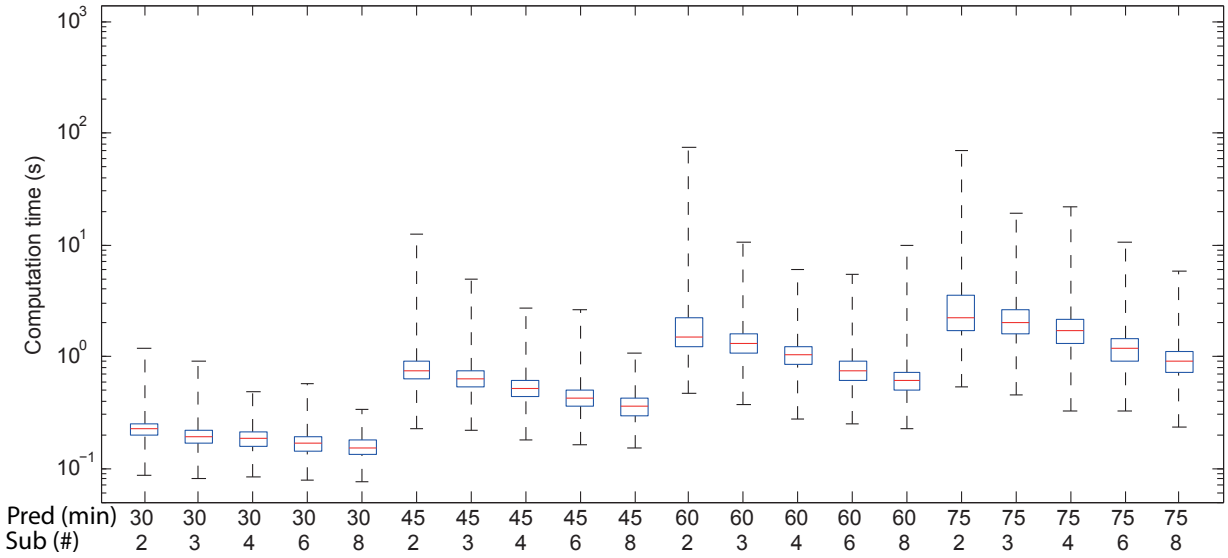

Figure 6: Computation time needed for DMPC method 4 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts. The number of minutes of the prediction horizon are given by "Pred (min)" below the figure and the number of parts is given by "Sub (#)".

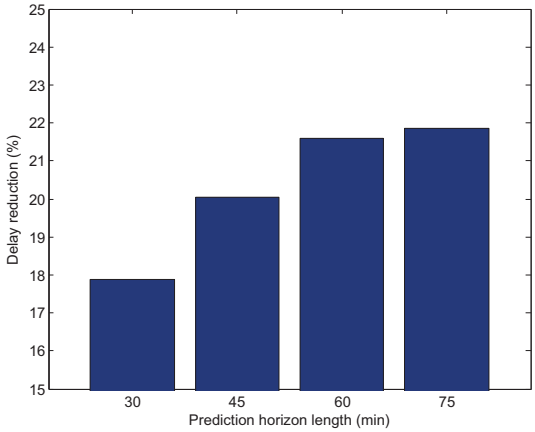

Figure 7: Reduction of the delays in percentage compared to the nominal case for the centralized MPC approach for the case study.

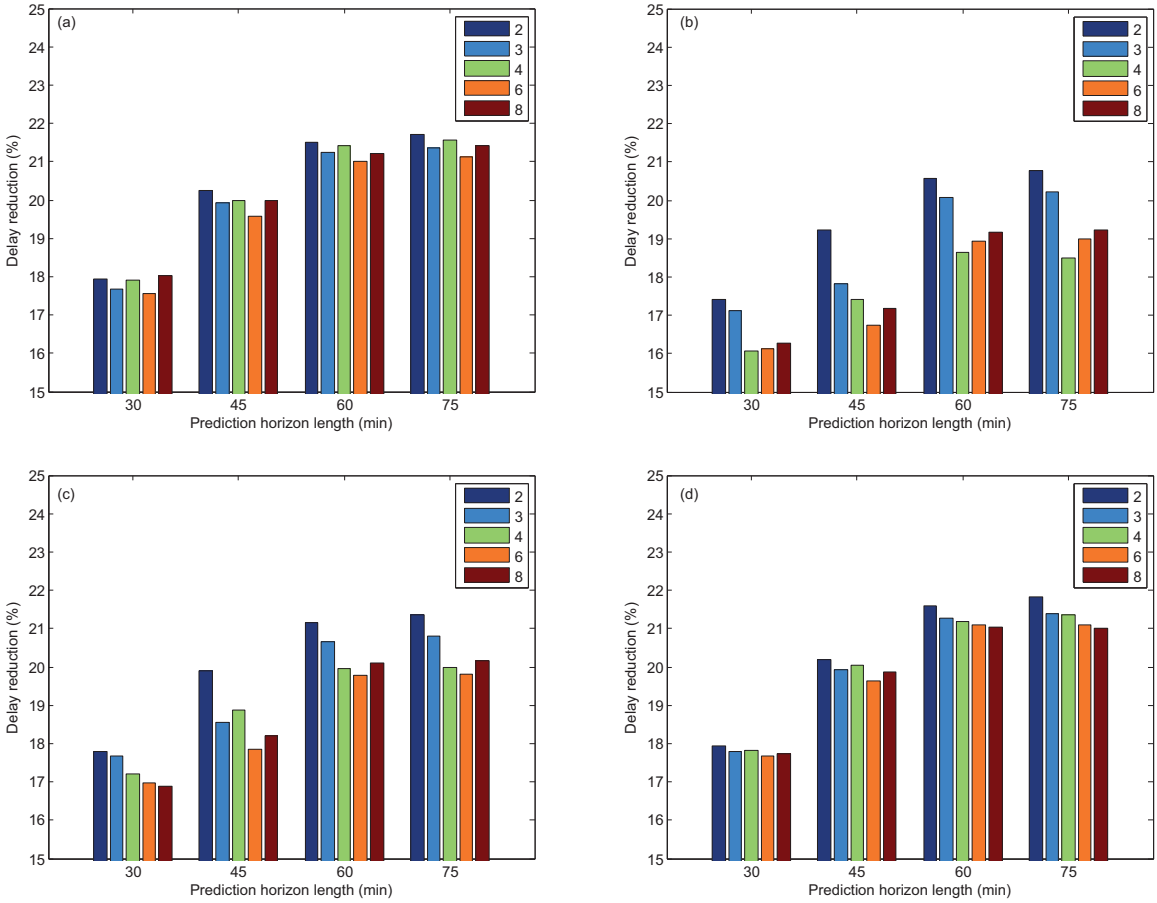

Figure 8: Reduction of the delays in percentage compared to the nominal case for (a) DMPC approach 1, (b) DMPC approach 2, (c) DMPC approach 3, and (d) DMPC approach 4 for the partitions in 2, 3, 4, 6, and 8 parts.

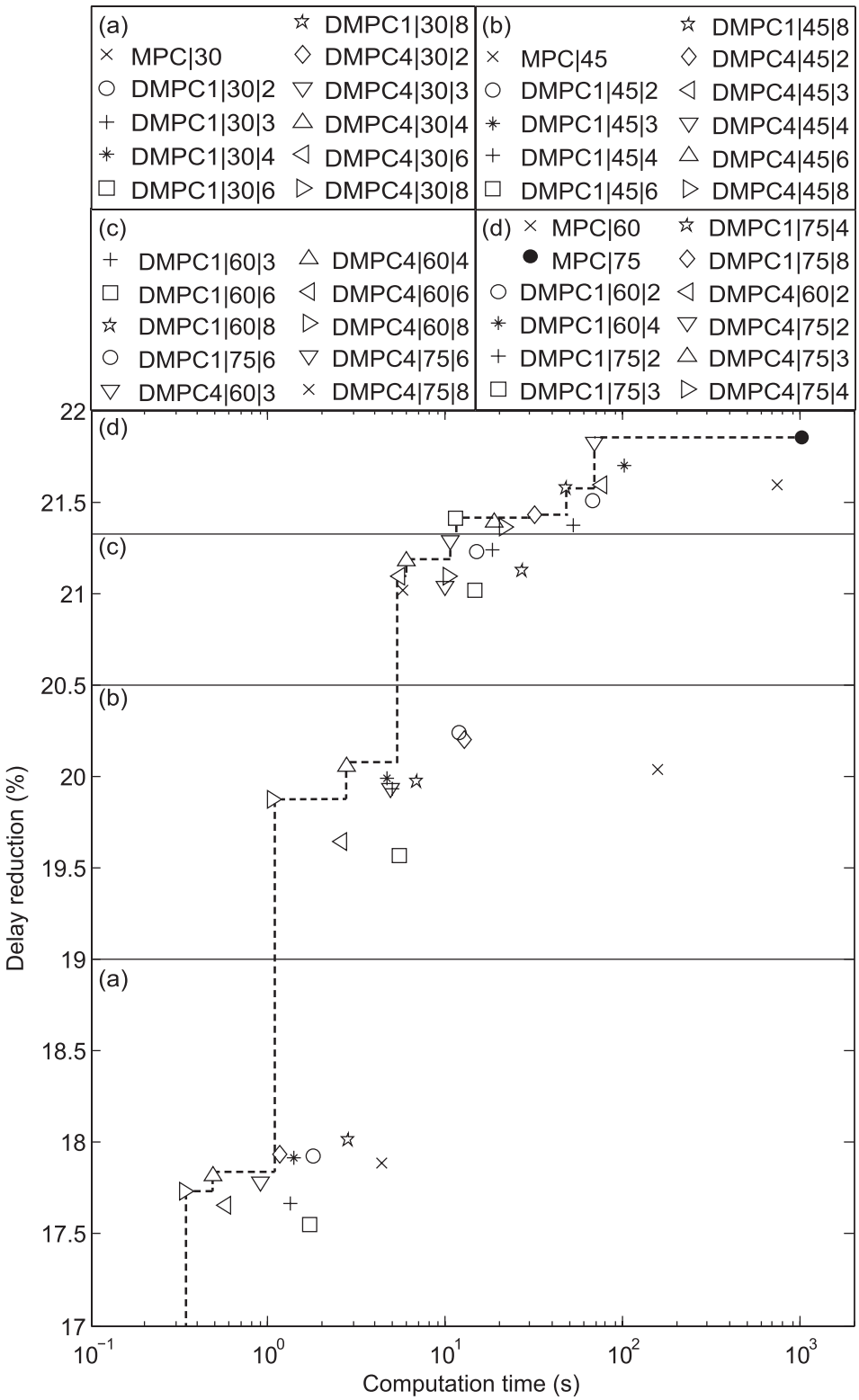

Figure 9: Reduction of the delays in percentage plotted against the maximum computation for DMPC method 1, DMPC method 4, and the MPC method for the partitions in 2, 3, 4, 6, and 8 parts, and the prediction horizons of 30, 45, 60, and 75 minutes. The dashed line shows the Pareto frontier connecting the Pareto optimal methods.