Delft University of Technology

Delft Center for Systems and Control

Technical report 14-036

Distributed model predictive control for rescheduling of railway traffic*

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

If you want to cite this report, please use the following reference instead:

B. Kersbergen, T. van den Boom, and B. De Schutter, "Distributed model predictive control for rescheduling of railway traffic," *Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC 2014)*, Qingdao, China, pp. 2732–2737, Oct. 2014.

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://pub.deschutter.info/abs/14_036.html`

Distributed Model Predictive Control for Rescheduling of Railway Traffic

Bart Kersbergen[1], Ton van den Boom[1] and Bart De Schutter[1]

Abstract— **In this paper we introduce two distributed model predictive control (DMPC) methods for the rescheduling of railway traffic. In each step of the DMPC approach dispatching actions are determined that reduce the amount of delay in the network as much as possible by solving a mixed integer linear programming (MILP) problem. The constraints of the MILP are based on a model of the railway traffic and network and the possible dispatching actions.**

In the first method each subproblem consists of the complete constraint matrix and the solver tries to minimize the centralized cost function, but can only change a limited number of binary variables (which correspond to the dispatching actions). By limiting the number of binary variables each subproblem is easier to solve than the centralized problem. For the second method each subproblem consists of only a part of the problem and the solver minimizes a local cost function, and it can only change the binary variables for that part of the problem. This reduces the complexity of the subproblems even further, but the solver can not determine the effects of the binary variables on the solution quality of the other subproblems.

Both methods significantly reduce the time needed to determine the dispatching actions. The average time needed to compute the solution is 11.56 times shorter when using method 1 and 39.11 times shorter when using method 2. The solution found is on average only 0.63 % less optimal for method 1 and 1.27 % less optimal for method 2.

I. Introduction

In many countries in the world large, complex, and very busy railway networks have been built. Especially in North-West Europe, China, and Japan the railway networks are used near the maximum capacity. As a result, very little buffer and slack time is available to recover from delays. Every day small delays occur in almost all railway networks, but in these heavily used railway networks small delays may cause many extra delays in large parts of the network because of the small buffer and slack times. In order to effectively deal with these delays dispatchers reschedule trains, break connections, or even cancel trains. Currently most dispatchers take these decisions based on their experience, a given set of ground rules, and a limited overview of the network situation.

In recent years many researchers have been developing dispatcher support systems [1], [2], [3], [4], [5], [6] to help the dispatchers determine which actions to take. Most of these approaches model a part of the network (from one station or track up to several railway stations and the tracks between them) as a microscopic model. The microscopic model allows for very accurate modeling of the railway network but results in a large and complex model. In [7] it was shown that for rescheduling a macroscopic model can be sufficiently detailed.

In many of the dispatching support systems model predictive control (MPC)[8] is used to determine the optimal dispatching actions based on a prediction of the evolution of the state of the system under control, using a model of that system. Recently the use of MPC has been extended from discrete-time systems to discrete-event systems [9] opening a whole new set of systems to be controlled with MPC.

For very complex or large systems it is not always feasible to design one centralized controller to control the entire system. It is sometimes better to split the system into several smaller parts and to control these parts individually. For such systems MPC has been extended to distributed model predictive control (DMPC) [10], [11]. Another approach is multi-level control as proposed in [2], [3], where multiple local controllers are connected to each and a supervisory controller ensures the border constraints between two local controllers are satisfied. With DMPC there is no supervisory controller and the local controllers have to coordinate with each other to ensure the border constraints are satisfied.

We build on the work of [12], [13], [14], [15] where the railway network and traffic is modeled as a discrete-event system and the dispatching problem is solved using a centralized model predictive controller. We extend on that work by proposing two DMPC methods to determine the dispatching actions for the system. With these DMPC methods we can reduce the time needed to solve the dispatching actions compared to the previously used MPC method.

In Section II the railway traffic and network model is described. In Section III we explain how the model predictive controller is constructed. The two DMPC methods are explained in IV. The performance of the two DMPC methods is compared to the centralized MPC approach for a case study of the Dutch Railway Network in Section V. Finally in Section VI we discus the result, draw conclusions, and discuss our future research plans.

II. Railway Traffic and Network Model

We model the railway traffic and network as a discrete-event system. The events of the system are the arrivals and departures of trains at stations and junctions. Because the railway traffic and network models we consider are very large we have to make a trade-off between the level of detail we model and the size of the model. Therefore we model the railway traffic and network on a macroscopic level. The stations and their interlocking areas are modeled as single points in the network. We assume stations have sufficient capacity to handle all trains that arrive at the station and

[1]B. Kersbergen, T.J.J. van den Boom and B. De Schutter are with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands.

that rerouting through station areas is done locally and is without deadlocks. Each track connecting stations and/or junctions to each other is modeled as a single element. The signaling system to ensure a safe distance between trains is not modeled explicitly either. Instead a safe distance is ensured by constraints on the arrival and departure times of the trains. The simplification in the modeling of the station areas and the tracks significantly reduces the size of the model, but also decreases the accuracy of the model. The model however is still sufficiently accurate for the purpose of rescheduling of railway traffic. It has been shown in [7] that a macroscopic model can be sufficient for the purpose of rescheduling. In this paper the basics of the railway traffic model we will use are described. For a complete description of the model, the interested reader is referred to [12], [13], [14].

The events are connected to each other through several constraints. There are five types of constraints, where each type models a different part of the railway system. The types of constraints are:

- continuity constraints that connect the arrival and departure events of a single train at a station,
- running time constraints that describe the train moving from one station to another,
- headway and separation constraints that ensure a safe distance between trains on the same track,
- connection constraints that ensure passengers can transfer from one train to another,
- timetable constraints that ensure trains do not depart too early.

The first four types of constraints have the same general form:

$$x_i \geq x_j + \tau_{ij}, \tag{1}$$

where $x_i, x_j \in \mathbb{R}$ are the event times and $\tau_{ij} \in \mathbb{R}$ is the minimum process time (dwell, running, headway, separation, or connection time). The timetable constraints have the following form:

$$x_i \geq r_i, \tag{2}$$

where $r_i \in \mathbb{R}$ is the scheduled departure (or arrival) time. The trains depart and arrive as soon as all constraints are satisfied.

Since the headway constraints define a minimum time difference between the events of two trains on the same track, they also define the order in which the trains traverse the tracks. A different order of trains requires a different set of headway constraints. Changing the order of the trains must therefore be done by activating and deactivating sets of headway constraints, which can be done by adding control variables to the headway constraints. Constraints with control variables have one of the following two general forms:

$$x_i \geq x_j + \tau_{ij} + u_{ij}\beta \tag{3}$$

$$x_i \geq x_j + \tau_{ij} + (1 - u_{ij})\beta, \tag{4}$$

where $u_{ij} \in \{0, 1\}$ is the binary control variable and $\beta \ll 0$ is a large negative value. If the binary variable is zero, the first constraint coincides with (1), while the second constraint does not influence the value of x_i, since a large negative value is added to the constraint, and as a result the second constraint is always satisfied. If the binary variable is one, the first constraint does not influence the value of x_i, because of the large negative value added to it, while the second constraint coincides with (1). Control variables can also be added to connection constraints allowing us to break or keep connections.

The event time x_i is then in general determined by an equation of the form:

$$\begin{aligned} x_i = \max \big(r_i, & \max_{x_j \in X_i^1} (x_j + \tau_{ij}), \\ & \max_{x_k \in X_i^2} (x_k + \tau_{ik} + u_{ik}\beta), \\ & \max_{x_l \in X_i^3} (x_l + \tau_{il} + (1 - u_{il})\beta) \big) \end{aligned} \tag{5}$$

where X_i^1 is the set of events for which the constraints on x_i have the form of (1), X_i^2 is the set of events for which the constraints on x_i have the form of (3), and X_i^3 is the set of events for which the constraints on x_i have the form of (4).

III. Model Predictive Control

A model predictive controller determines the optimized control inputs at discrete-time instants $t(k)$ for $k = 0, 1, \ldots$. At each time instant $t(k)$ the optimized control inputs, computed at time instant $t(k-1)$, that are in $[t(k), t(k+1))$, are implemented. Furthermore the controller receives information about the current state of the system that is to be controlled. Together with a model of the system this information is used to calculate new optimized control inputs based on the effects of the control inputs have on the evolution of the state of the model. At $t(k+1)$ the whole process is repeated. The model predictive controller therefore only has the time between two time instants to determine the optimized control inputs.

When we talk about the model predictive controller in the rest of this paper we only consider determining the control inputs for a single step, and we will not go into detail on how the information from the system is gathered or how the control inputs are updated when new information is obtained. In [16] a tool is described that gathers the system information and uses historical data to predict the future process times and future conflicts between trains. In [17], [8] it is explained how the control inputs are implemented and updated step after step.

To determine the optimal dispatching actions a mixed integer linear programming (MILP) problem is constructed and solved. In [13], [15] it is explained how this MILP can be determined. The MILP problem has the following structure:

$$\begin{aligned} & \min_z c\, z \\ & \text{s.t. } A\, z \leq b \end{aligned}$$

with

$$z = \begin{bmatrix} x^\top & u^\top \end{bmatrix}^\top$$
$$c = \begin{bmatrix} c_x & c_u \end{bmatrix}$$
$$A = \begin{bmatrix} A_x & A_u \end{bmatrix}$$

where x are the continuous variables (in our case the arrival and departure times of the trains), u are the binary variables (in our case the dispatching actions), c_x and c_u are the weights for the continuous and binary variables respectively, A_x is the part of matrix A that is multiplied by x, and A_u is the part of matrix A that is multiplied by u. The inequality $A\,z \leq b$ contains the constraints from Section II. The weights c_x and c_u are chosen such that the cost function is a measure for the total delay in the network. In the case study we will give the exact cost function.

For large railway networks finding the optimal solution to the MILP problem may take a long time, since solving MILP problems is considered NP-hard [18] and the worst case time needed to solve it is generally assumed to increase exponentially with the size of the problem. Because of that it may take more time to solve the problem than there is time available for determining the control inputs for the next step. We have therefore developed two distributed model predictive control methods.

IV. Distributed Model Predictive Control

In DMPC the system is divided into a number of subsystems and each subsystem is controlled by its own model predictive controller. The controllers determine the optimal control inputs for the subsystems while interacting with the controllers of the other subproblems to ensure global feasibility and good global performance.

The advantage of dividing the system into subsystems is that finding the control inputs for the subsystems is in general much easier than finding the control inputs using the centralized MPC approach. The downside is that the control inputs found may be suboptimal for the complete system.

In this section we propose two DMPC methods to find the dispatching actions for the whole network.

Before the two DMPC methods are explained we will first reorder the rows and columns of the constraint matrix A and reorder the variables in vector z, such that the structure of A gets as close as possible to a block diagonal structure with n blocks, where n is the number of subsystems. The constraint matrix has the following structure after reordering:

$$\min_z \begin{bmatrix} \tilde{c}_1 & \tilde{c}_2 & \dots & \tilde{c}_n \end{bmatrix} \begin{bmatrix} \tilde{z}_1^\top & \tilde{z}_2^\top & \dots & \tilde{z}_n^\top \end{bmatrix}^\top \tag{6}$$

$$\text{s.t.} \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & \ddots & & A_{2,n} \\ \vdots & & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{bmatrix} \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \vdots \\ \tilde{z}_n \end{bmatrix} \leq \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix} \tag{7}$$

where $\tilde{c}_i$, $\tilde{z}_i$, $\tilde{b}_i$ for $i \in \{1, \dots, n\}$ are vectors of appropriate size, $A_{i,j}$ for $i \in \{1, \dots, n\}$, $j \in \{1, \dots, n\}$ are matrices of appropriate size, and

$$\tilde{z}_i = \begin{bmatrix} \tilde{x}_i^\top & \tilde{u}_i^\top \end{bmatrix}^\top$$
$$A_{i,i} = \begin{bmatrix} A_{i,i,x} & A_{i,i,u} \end{bmatrix}$$
$$A_{i,j} = \begin{bmatrix} A_{i,j,x} & \mathbf{0} \end{bmatrix}$$

for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, n\}/\{i\}$, and where $\tilde{x}_i$ and $\tilde{u}_i$ are vectors of appropriate size.

This reordering is based on the following goals:

- The constraints that have an element in $A_{i,j}$ should only depend on continuous variables.
- The number of elements in $A_{i,j}$ should be minimized.
- The difference in the number of elements in the diagonal matrices $A_{i,i}$, for $i \in \{1, \dots, n\}$, should be minimized.
- The difference in the number of binary and continuous variables between the vectors $\tilde{z}_i$ and $\tilde{z}_j$, for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, n\}/\{i\}$ should be minimized.

The first goal ensures the structure of the non-diagonal matrices $A_{i,j}$. The second goal ensures that the interaction between the continuous variables in two different vectors $\tilde{z}_i$ and $\tilde{z}_j$ is minimized. The third and fourth goal ensures that the difference in the number of constraints, binary, and continuous variables for the subproblems resulting from this reordering is minimized.

The first goal can be achieved by grouping the variables of constraints of the form (3) and (4) in the same vector $\tilde{z}_i$. Since headway, separation, and breakable connection constraints are of the form (3) or (4), the continuous and binary variables of these constraints must be grouped in the same $\tilde{z}_i$. That only leaves running, dwell, and unbreakable connection constraints. Since the headway constraints between the departures of two trains on the same track and the arrivals of two trains depend on the same variable, the continuous (and binary) variables of both the arrival and departure headway constraints must be grouped in the same $\tilde{z}_i$. As a result the continuous variables of the running time constraints are already in the same $\tilde{z}_i$. That leaves only the variables of continuity and unbreakable connection constraints to be in different vectors $\tilde{z}_i$ and $\tilde{z}_j$.

If we look at the railway model this means that the continuous variables of 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 $\tilde{z}_i$. The first step should thus be to group the variables and constraints per track, resulting in T vectors of variables and T diagonal matrices and $T^2 - T$ non-diagonal matrices, where T is the number of tracks in the model. If a breakable connection constraint exists between the variables of two vectors $\tilde{z}_i$ and $\tilde{z}_j$, the two vectors should be combined into $\tilde{z}_k$, also resulting in a new matrix $A_{k,k}$:

$$\tilde{z}_k = \begin{bmatrix} \tilde{z}_i^\top & \tilde{z}_j^\top \end{bmatrix}^\top$$
$$A_{k,k} = \begin{bmatrix} A_{i,i} & A_{i,j} \\ A_{j,i} & A_{j,j} \end{bmatrix}$$

The other three goals are achieved simultaneous by further reducing the number of vectors $\tilde{z}_i$ to n by combining them.

By combining the vectors the total number of elements in all the non-diagonal matrices $A_{i,j}$ is reduced, but the remaining matrices will be larger. The vectors, for which the continuous variables are connected to each other through the largest number of constraints, are combined first, while ensuring that none of the final n vectors and resulting diagonal matrices is much smaller or larger than the others.

The grouping of the variables for the reordering can be determined before hand since the general problem structure does not change for different scenarios. As a result reordering the constraint matrix can be done very quickly while rescheduling.

A. Distributed Method 1

Based on this reordered constraint matrix two methods for solving the problem in a distributed manner are now developed. For the first method each subproblem optimizes the centralized cost function while considering all constraints of the centralized problem, but for subproblem i all continuous variables, denoted by x, and the binary variables in $\tilde{z}_i$, denoted by $\tilde{u}_i$, can be used to optimize the cost function. The other binary variables in z are determined by the other subproblems and cannot be changed by subproblem i. As a result the number of binary variables that can be changed is greatly reduced. A direct consequence of this is that the MILP problem will be much easier to solve. The subproblems are then solved in sequence. A feasible initial solution can always be found. Indeed, the simplest feasible solution is the solution when the train orders do not change and no connections are broken, then all binary variables will be zero, and only the arrival and departure times are updated in order to avoid conflicts and deadlocks on open tracks. The method can be summarized as follows:

Each subproblem can be written as:

$$\min_{x,\tilde{u}_i} c\,z \tag{8}$$

$$\text{s.t. } A\,z \leq b \tag{9}$$

where $u/\{u_i\}$ is determined by the other subproblems.

The steps to determine the solution are:

1) Define an initial estimate for z denoted by $\hat{z}$
2) Use $\hat{z}$ as an initial solution for subproblem i in (8) and (9), for $i = 1$, and solve it. Denote the solution as $\hat{z}^i$
3) Update the estimate: $\hat{z} = \hat{z}^i$
4) Repeat steps 2 and 3 for $i = 2, \ldots, n$.
5) Repeat steps 2, 3, and 4 until $||\hat{z} - \hat{z}^i|| < \Delta$ for $i = 1, \ldots, n$, where $\Delta < 0.0001$.

The initial solution can be determined by a heuristic method, or can be the result found by setting all binary variables to zero, which can be determined very fast.

Since a feasible initial solution 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 is 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 the stopping criteria is reached, the algorithm stops.

B. Distributed Method 2

For the second DMPC method each subproblem only considers a part of the MPC problem in (7). Subproblem i only optimizes the binary and continuous variables in $\tilde{z}_i$. It does not consider or model the effects of its choices on the other subproblems. The other binary and continuous variables are determined by the other subproblems and cannot be changed by subproblem i. Each subproblem can be written as:

$$\min_{\tilde{z}_i} \tilde{c}_i\,\tilde{z}_i \tag{10}$$

$$\text{s.t. } A_{i,i}\tilde{z}_i \leq b_i - \sum_{j \in \{1,\ldots,n\}/\{i\}} A_{i,j}\tilde{z}_j \tag{11}$$

where $\tilde{z}_j$, for $j \in \{1, \ldots, n\}/\{i\}$ are determined by the other subproblems.

The steps to determine a near optimal feasible centralized solution are:

1) Define an initial estimate for z denoted by $\hat{z}$
2) For subproblem i assume $\tilde{z}_j$ for $j \in \{1, \ldots, n\}/\{i\}$ is known and equal to $\hat{\tilde{z}}_j$ and solve subproblem i for $i = 1$. Denote the solution as $\hat{\tilde{z}}_i^i$.
3) Update the estimate $\hat{\tilde{z}}_i = \hat{\tilde{z}}_i^i$.
4) Repeat steps 2 and 3 for $i = 2, \ldots, n$.
5) Repeat steps 2, 3, and 4 until $||\hat{\tilde{z}}_i - \hat{\tilde{z}}_i^i|| < \Delta$ for $i = 1, \ldots, n$, where $\Delta < 0.0001$.

A feasible solution can always be found in step 2, since setting all binary variables to zero will always result in a feasible solution, the delays however may be much higher in that case.

The advantage of this method is that the subproblems are even smaller and easier to solve. 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 less optimal than those found with method 1. The case study of the next section will show that the algorithm converges for all scenarios we have ran. In our future work we will prove that, under certain conditions method 2 always converges .

V. Case Study

We have tested the two proposed DMPC methods for a case study based on the Dutch railway network with the timetable of 2011. The largest part of the network is considered, the tracks from Leiden via Gouda to Woerden (and back) has been left out, the tracks from Baarn to Den Dolder (and back), as well as the tracks from Almelo to Zwolle (and back), and the tracks that are not electrified. But the trains on these parts of the network are mostly unconnected to the rest of the network. All types of passenger trains are included. The only dispatching action in this case study is the reordering of trains.

The two DMPC methods are tested for 1000 scenarios, where we randomly delayed 10% of the trains with random delays based on a Weibull distribution with scale parameter 8 and shape parameter 0.8. During one scenario the MPC controller has to determine the optimal dispatching actions of one cycle of the model, which corresponds to one hour of the timetable. The resulting MILP problem is split up into four smaller subproblems, based on the method explained in Section IV. The structure of the constraint matrix is shown in Figure 1. The case study was run on an AMD quad core CPU running at 4GHz with 16GB memory on Windows 7 running Matlab® R2013b, and using Gurobi 5.6.0[19] to solve the MILP problem.

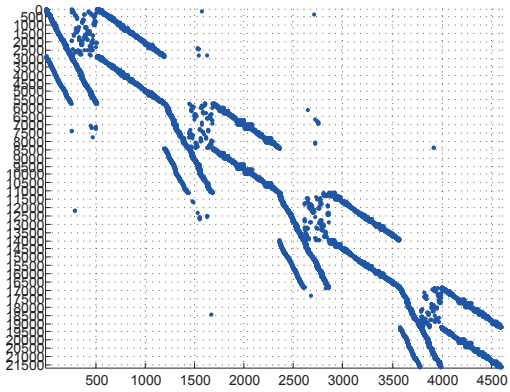

Fig. 1. Structure of the constraint matrix A. Matrix A has 21658 rows and 4602 columns. The dots correspond to non-zero elements in the matrix.

The cost function for all problems is the sum of delays: $cz = [\mathbb{1}^{1\times n_x}\ 0.0001 \cdot \mathbb{1}^{1\times n_u}] \begin{bmatrix} x \\ u \end{bmatrix}$, where $\mathbb{1}^{1\times m}$ is a 1 by m vector containing only ones, n_x is the number of continuous variables, and n_u is the number of binary variables. We added a small weight to the dispatching actions to ensure that if multiple solutions result in the same sum of delays the solution with the least number of dispatching actions is chosen.

The total number of constraints is 21658. Of these 21658 constraints only 48 connect the four subproblems. The number of constraints, binary, and continuous variables of the centralized problem and the subproblems are given in Table I for DMPC method 1 and in Table II for DMPC method 2.

TABLE I

SPECIFICATIONS OF THE MILP PROBLEMS FOR DMPC METHOD 1

Problem	Number of constraints	Continuous variables	Binary variables
Centralized	21658	1930	2672
1	21658	1930	683
2	21658	1930	673
3	21658	1930	711
4	21658	1930	605

We will compare the solution found using the DMPC methods to the solution found with the centralized optimization problem. The increase in delays compared to the optimal solution is determined for all scenarios, as well as the time needed to find the solution.

TABLE II

SPECIFICATIONS OF THE MILP PROBLEMS FOR DMPC METHOD 2

Problem	Number of constraints	Continuous variables	Binary variables
Centralized	21658	1930	2672
1	5724	512	683
2	5410	490	673
3	5704	504	711
4	4820	424	605

In Figures 2 are box plots[1] that show the increase in delays, in minutes (a) and in percentage (b), of the solution found with the DMPC methods compared to the centralized solution for the 1000 scenarios. In Figure 3 are box plots illustrating the time needed to find the optimal solution for the DMPC methods and the MPC method. In Table III the results are shown numerically.

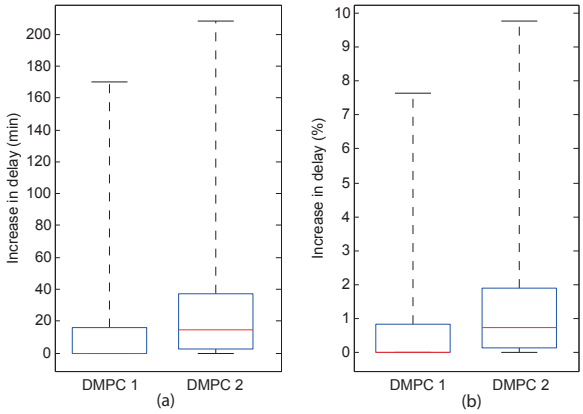

Fig. 2. Box plots[1] of the increase in delays (minutes in (a) and percentage in (b)) as a result of using DMPC method 1 and method 2.

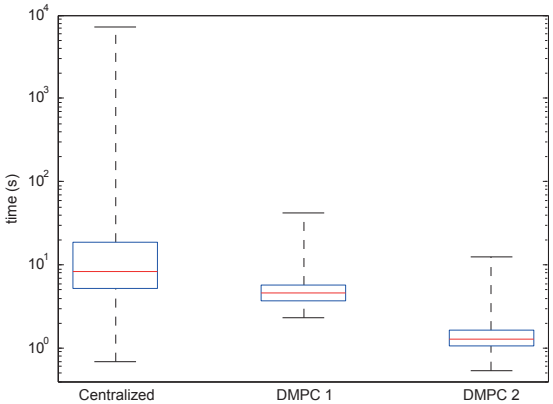

Fig. 3. Box plots[1] of the time needed to solve problem using the centralized MPC, DMPC method 1, and DMPC method 2 on an AMD quad core CPU running at 4GHz with 16GB memory.

The average computation time of DMPC method 1 is 11.56 times shorter than the computation time of the centralized approach, the maximum computation time needed is 171

[1]Box plots divide the results into four equally sized parts: the 25% of the results with the lowest values are indicated by the lower vertical dashed line and bottom horizontal solid line. The 25% of the results with the highest values are represented by the upper vertical dashed line and top horizontal solid line. The other 50% is shown in the (blue) rectangle between the two dashed vertical lines, where the median of the results is presented by the (red) horizontal line splitting the box in two.

TABLE III

RESULTS FOR THE CENTRALIZED AND DISTRIBUTED METHODS

	Average increase of the sum of delays	Maximum increase of the sum of delays	Average computation time (s)	Maximum computation time (s)
MPC	N/A	N/A	60.23	7121
DMPC 1	12.7 min 0.63%	168 min 7.64%	5.21	41.63
DMPC 2	25.1 min 1.27%	207 min 9.75%	1.54	12.44

times shorter. Of the 1000 scenarios the distributed method found a solution, that was at most Δ less optimal than the optimal solution for the centralized problem in 535 cases, where $\Delta < 0.0001$. On average the second DMPC method is 39.11 times faster than the centralized approach and the maximum time needed is 572 times shorter. Of the 1000 scenarios the distributed method found a solution, that was at most ε less optimal than the optimal solution for the centralized problem in 201 cases, where $\varepsilon < 0.0001$. The downside of the second approach is that the solutions found result in a bit more delay than the solutions found with the first method. On average the increase in delays is doubled and the maximum increase in delays is 1.28 times higher. However, the second method is on average 3.40 times faster than the first method and the maximum time needed to solve the problem is 3.34 times lower.

Both proposed DMPC methods find very good solutions, while the computation time is a lot lower than the computation time of the centralized approach.

VI. DISCUSSION

To be able to implement the system in reality we need to be able to show that the controller will be able to find good feasible solutions within a short time frame for any scenario. With the two distributed methods introduced in this paper we have shown that this is possible.

With method 1 each subproblem considers the global problem but each subproblem can only modify a limited number of binary variables. With method 2 each subproblem only considers a part of the global problem and does not consider the rest of the problem.

Whether method 1 or 2 should be used depends on the requirements on the computation time and how far from the global optimum the found solutions can be. Method 1 gives better solutions than method 2, but method 2 is about 3.4 times faster. If global optimality is a hard requirement then the centralized approach should be used.

The next step of our research will be to try and prove that method 2 always converges and the stopping criteria is reached for every possible scenario. So far we have shown that for all of the 1000 scenarios that we ran it converged. Based on this it seems likely that it indeed converges for any scenario and it should therefore be possible to prove convergence.

As we have mentioned the subproblems in method 2 do not consider the effects of their control actions on the other subproblems. In our future work we will therefore also develop methods to include a measure of the effects of the control actions on the other subproblems in to the subproblems of method 2, and by doing so improving the solution that the method finds.

REFERENCES

[1] M. M. Dessouky, Q. Lu, J. Zhao, and R. C. Leachman, "An exact solution procedure to determine the optimal dispatching times for complex rail networks," *IIE Transactions*, vol. 38, no. 2, pp. 141–152, 2006.

[2] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "Centralized versus distributed systems to reschedule trains in two dispatching areas," *Public Transport*, vol. 2, pp. 219–247, 2010.

[3] ——, "Dispatching and coordination in multi-area railway traffic management," *Computers & Operations Research*, vol. 44, no. 1, pp. 146 – 160, 2014.

[4] J. Törnquist Krasemann, "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances," *Transportation Research Part C: Emerging Technologies*, vol. 20, no. 1, pp. 62–78, 2012.

[5] G. Caimi, M. Fuchsberger, M. Laumanns, and M. Lüthi, "A model predictive control approach for discrete-time rescheduling in complex central railway station areas," *Computers and Operations Research*, vol. 39, no. 11, pp. 2578–2593, 2012.

[6] L. Meng and X. Zhou, "Simultaneous train rerouting and rescheduling on an n-track network: A model reformulation with network-based cumulative flow variables," *Transportation Research Part B: Methodological*, vol. 67, no. 0, pp. 208–234, 2014.

[7] P. Kecman, F. Corman, A. D'Ariano, and R. M. Goverde, "Rescheduling models for railway traffic management in large-scale networks," *Public Transport*, vol. 5, no. 1-2, pp. 95–123, 2013.

[8] C. E. Garcí, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice - a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[9] T. J. J. van den Boom and B. De Schutter, "Modelling and control of discrete event systems using switching max-plus-linear systems," *Control Engineering Practice*, vol. 14, no. 10, pp. 1199–1211, Oct. 2006.

[10] M. Farina and R. Scattolini, "Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems," *Automatica*, vol. 48, no. 6, pp. 1088–1096, 2012.

[11] W. Dunbar, "Distributed receding horizon control of dynamically coupled nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 7, pp. 1249–1263, July 2007.

[12] T. J. J. van den Boom, B. Kersbergen, and B. De Schutter, "Structured modeling, analysis, and control of complex railway operations," in *Proceedings of the 51st IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2012, pp. 7366–7371.

[13] B. Kersbergen, T. J. J. van den Boom, and B. De Schutter, "On implicit versus explicit max-plus modeling for the rescheduling of trains," in *Proceedings of the 5th International Seminar on Railway Operations Modelling and Analysis (RailCopenhagen)*, Copenhagen, Denmark, May 2013, pp. 467–481.

[14] ——, "Reducing the time needed to solve the global rescheduling problem for railway networks," in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC2013)*, The Hague, The Netherlands, Oct. 2013, pp. 791–796.

[15] J. Rudan, B. Kersbergen, T. van den Boom, and K. Hangos, "Performance analysis of milp based model predictive control algorithms for dynamic railway scheduling," in *Proceedings of the European Control Conference 2013 (ECC)*, July 2013, pp. 4562–4567.

[16] P. Kecman and R. Goverde, "Adaptive, data-driven, online prediction of train event times," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC2013)*, Oct 2013, pp. 803–808.

[17] B. De Schutter and T. van den Boom, "Model predictive control for max-plus-linear discrete event systems," *Automatica*, vol. 37, no. 7, pp. 1049–1056, 2001.

[18] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982.

[19] "Gurobi optimizer reference manual," 2014. [Online]. Available: http://www.gurobi.com