Technical report 18-011

The interaction between scheduling and control of semi-cyclic hybrid systems*

T.J.J. van den Boom, H. de Bruijn, B. De Schutter, and L. Özkan

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/18_011.html

The interaction between scheduling and control of semi-cyclic hybrid systems

Ton J.J. van den Boom [*] Hilco de Bruijn [*] Bart De Schutter [*]
Leyla Özkan [**]

[*] *Delft University of Technology, Delft Center for Systems and Control, Delft, The Netherlands (e-mail: a.j.j.vandenboom@tudelft.nl)*
[**] *Eindhoven University of Technology, Control Systems Group, Eindhoven, The Netherlands*

Abstract: In this paper a new iterative approach is proposed for the design of a combined real-time scheduling and control algorithm that can be applied to industrial systems that are described by a hybrid model with a (semi-)cyclic behavior. Traditionally scheduling and control problems are considered in a sequential way. First the scheduling problem is solved and subsequently the control problem. This may result in inconsistent solutions such that the system may not operate adequately and does not reach the desired operational targets.
In our approach scheduling is done with model predictive control using a switching max-plus linear model of the discrete event part of the system. The interface with a reference generator determines whether the computed reference signal will lead to a feasible response. Furthermore, it estimates the duration of the operations in the system based on the actual state, and communicates that with the scheduler. In an iterative procedure the optimal and feasible schedule can be computed. In a case study the railway traffic on a single track is considered, showing that updating the schedule results in feasible local speed profiles for the trains and less delay in the overall system in case of a delay.

Keywords: Scheduling algorithms, Control, Hybrid Systems, Switching Max-Plus linear systems, Iterative methods

1. INTRODUCTION

For complex dynamical systems with a semi-cyclic behavior, such as manufacturing operation, large cyber-physical systems, and transportation systems, the real-time operation involves a large number of scheduling decisions distributed over several layers of automation hierarchy. This paper is concerned with the development of new methodologies and algorithms for the smart online interaction between scheduling and control layers.

The major disadvantage of traditional scheduling, planning and control techniques is that the problems are considered separately and solved in a sequential top-down way: First a schedule is computed, subsequently the schedule is translated into a number of reference signals that need to be tracked by the controllers.

To avoid these issues, our aim in this paper is to design a combined real-time scheduling and control algorithm that can be applied to industrial systems that are described by a hybrid model with a (semi-)cyclic behavior (such as production systems (Mutsaers et al., 2012), cyber-physical systems (Alirezaei et al., 2012), traffic networks (Kersbergen et al., 2016), queuing systems and array processors). Semi-cyclic behavior occurs when cycles deviate from each other because of different ordering, different routing or just different parameters. We will go through scheduling and control levels sequentially at regular intervals. This means that we need to monitor the process status continuously and establish a smart interaction between the scheduler and the controllers. This interaction is done via a smart interface.

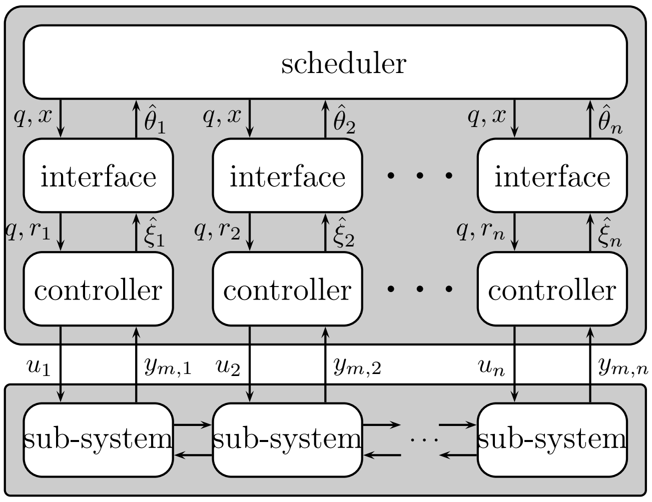

Figure 1: Process with interacting scheduler and controller.

The overall system is depicted in Figure 1 and it consists of four layers:

(1) Process with interacting sub-systems. Each sub-system i has an input u_i and an output y_i.

(2) Control: Given the global discrete state q the ith model-based controller will compute a control signal u_i such that the sub-system i follows the given reference signal r_i. The measured state $\hat{\xi}_i$ is sent to the interface.
(3) Scheduling: The real-time scheduling strategy aims at providing the mode (represented by the discrete state q) and the vector x_i with begin and end times of all tasks based on the vector $\hat{\theta}_i$ of estimated operation times of sub-system i. In this paper we propose to use a switching max-plus linear (SMPL) approach to solve the scheduling problem.
(4) Interface: A smart interface between scheduler and control layers. The aim of the interface is to communicate the proposed schedule to the local controller, compute an appropriate reference signal r_i for the ith sub-system, and to extract process variables $\hat{\theta}_i$ efficiently from the dynamical hybrid model's data and communicate them to the scheduler.

In this paper we propose an iterative sequential approach. The scheduling problem and the control problem are solved sequentially in an iterative procedure. For the scheduling part we will use switching max-plus linear systems. Max-plus algebra has been used more often in literature to solve scheduling problems. In Yurdakul and Odrey (2004) an algorithm has been developed to solve a steady-state schedule problem by transforming their original non-linear model into a linear one. In Bouquard et al. (2006) the single machine, two-machines and three-machines flow shop scheduling problems have been solved. More recently, a mathematical formulation for cyclic flow-shops using max-plus algebra has been presented in Nambiar and Judd (2011), and a method to solve the cyclic job shop scheduling problem was proposed in Houssin (2011). In van den Boom et al. (2013) a methodology has been derived to systematically construct synchronization controllers for multiple cyclic discrete-event systems modeled in the max-plus framework.

2. EVENT-DRIVEN AND TIME-DRIVEN MODELS

Consider the hybrid system of Figure 1 consisting of a scheduler, interfaces, controllers and subsystems. The overall system can be represented by the hybrid automaton $H = (\mathcal{Q}, \Xi, f, \mathrm{Init}, \mathrm{Inv}, \mathcal{E}, \mathcal{R})$, where $\mathcal{Q}$ is a finite set of discrete states, Ξ is the continuous state space, f is a vector field, Init is the set of initial states, Inv describes the invariants of the locations, $\mathcal{E}$ is the transition relation, and $\mathcal{R}$ is the reset map. The state of the hybrid system is given by $(q, \xi) \in \mathcal{Q} \times \Xi$, and the evolution of the state is given by

$$\dot{\xi}(t) = f(q(t), \xi(t)) \ , \quad t \in \mathbb{R} \tag{1}$$

Invariants and guards describe when a transition will take place (the guard will enable a transition, the invariant forces a transition). The reset map specifies how new continuous states are related to previous continuous states for a particular transition.

We assume the overall system can be split into N subsystems, each with its own continuous state ξ_i, its own local discrete state $q_{\mathrm{loc},i}$ and with one common discrete state q_{com}, leading to the local system descriptions:

$$\dot{\xi}_i(t) = f_i(q_{\mathrm{com}}(t), q_{\mathrm{loc},i}(t), \xi_i(t)) \ , \quad \text{for } i = 1, \ldots, N \tag{2}$$

This means that we assume the interaction between the sub-systems is only in the discrete event domain, i.e. the coupling of begin and end times of the different operations in the sub-systems.

In (semi-)cyclic systems we assume that all operations will repeat in a (semi-)cyclic manner. Let $t_{j,\mathrm{b}}$ be the time instant that the hybrid system switches to discrete state q_j. The state will evolve according to (1) for a fixed q_j until the invariant and guard decide that a transition to another discrete state will take place. Let $t_{j,\mathrm{e}}$ be the end-time in discrete state q_j at which a new transition will take place. We can say that the duration of the operation in discrete state q_j is equal to $\theta_j = t_{j,\mathrm{e}} - t_{j,\mathrm{b}}$.

Based on the estimated values $\hat{\theta}$ of the operations in the system the scheduler will construct an optimal schedule for the overall system, described by the discrete state q.

Scheduling with SMPL systems

Now we arrive at the scheduling part of the system, in which the starting times $t_{j,\mathrm{b}}(k)$ and finishing times $t_{j,\mathrm{e}}(k)$ of the system's operations for cycle k are synchronized. This means that a next operation can only start if one or more other operations are finished.

Let all processing times in the system for the cycle k be collected in a vector

$$\theta(k) = [\, \theta_1(k) \ \theta_2(k) \ \cdots \ \theta_{N_\theta}(k) \,]^T$$

and define a vector

$$x(k) = [\, t_{1,\mathrm{b}}(k) \ t_{1,\mathrm{e}}(k) \ \cdots \ t_{n,\mathrm{b}}(k) \ t_{n,\mathrm{e}}(k) \,]^T$$

where $k \in \mathbb{Z}^+$ is the cycle counter.

These synchronizations can then be written in the form:

$$x_j(k) = \max_{p,\mu}(x_p(k-\mu) + \tau_{jp\mu})$$

where $\tau_{jp\mu}$ is the minimal delay time between event p in cycle $k - \mu$ and event j in cycle k. If there is no synchronization between event p in cycle $k - \mu$ and event j in cycle k we choose $\tau_{jp\mu} = -\infty$.

Now we introduce some notation from max-plus algebra (Baccelli et al., 1992). Define $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$ where $\varepsilon = -\infty$. The max-plus-algebraic addition ($\oplus$) and multiplication ($\otimes$) are defined as follows:

$$x \oplus y = \max(x, y) \quad , \quad x \otimes y = x + y$$

for any $x, y \in \mathbb{R}_\varepsilon$, and

$$[A \oplus B]_{i,j} = a_{i,j} \oplus b_{i,j} = \max(a_{i,j}, b_{i,j})$$

$$[A \otimes C]_{i,j} = \bigoplus_{k=1}^{n} a_{i,k} \otimes c_{k,j} = \max_{k=1,\ldots,n}(a_{i,k} + c_{k,j})$$

for matrices $A, B \in \mathbb{R}_\varepsilon^{m \times n}$ and $C \in \mathbb{R}_\varepsilon^{n \times p}$.

If we collect all values $\tau_{jp\mu}$ in a matrix A^μ with $[A^\mu]_{jp} = \tau_{jp\mu}$ the relation between the events becomes (van den Boom et al., 2013):

$$x(k) = \bigoplus_{\mu=1}^{\bar{\mu}} A^\mu(\theta) \otimes x(k-\mu)$$

where we write $A^\mu(\theta(k))$ because the variables $\tau_{jp\mu}$ are the entries of the vector θ (with $N_\theta = n^2\bar{\mu}$).

When we do scheduling of the system, the synchronization between different events may change (e.g. because of rerouting or reordering of the events). In van den Boom et al. (2013) we have shown that by introducing a binary scheduling variable $u(k) \in \{0, \varepsilon\}$ and the adjoint

$$\bar{u}(k) = \begin{cases} 0 & \text{if } u(k) = \varepsilon \\ \varepsilon & \text{if } u(k) = 0 \end{cases}$$

we can introduce the scheduling matrix $A^\mu(\theta(k))$ as follows

$$A^\mu(u(k), \theta(k), k) = \bigoplus_{\ell=1}^{L} u_\ell(k) \otimes \mathbf{A}^\mu_\ell(\theta(k), k) \;\oplus \bigoplus_{\ell=1}^{L} \bar{u}_\ell(k) \otimes \bar{\mathbf{A}}^\mu_\ell(\theta(k), k)$$

for some fixed matrices $\mathbf{A}^\mu_\ell(\theta(k), k)$ and $\bar{\mathbf{A}}^\mu_\ell(\theta(k), k)$ with L the number of max-plus binary decision variables. The synchronization between the events can be described by the so-called switching max-plus linear (SMPL) system description (van den Boom and De Schutter, 2006):

$$x(k) = \bigoplus_{\mu=1}^{\bar{\mu}} A^\mu(u(k), \theta(k), k) \otimes x(k-\mu) \qquad (3)$$

The event state $q(k)$ will be encoded by the max-plus binary variables $u(k)$.

In this paper we will use a model predictive control strategy for the scheduling part. With a receding horizon principle (Maciejowski, 2002; van den Boom et al., 2013) the schedule for the complete task is not calculated at once, but in several iterations. In every iteration the schedule is calculated for only the jobs in the nearest future, where only these few future jobs and the necessary past jobs are taken into account, instead of all jobs in the scheduling task. The scheduling task may contain many jobs. The computation time of the optimal solution increases as the number of scheduling variables increases. A too long computation time can cancel out the time gained by optimizing the schedule, or even deteriorate the total solution. The negative impact of the computation time can be avoided by using the receding horizon principle, which is one of the main characteristics of MPC. We aim for predictive operational scheduling, which means that based on observations of the system's behavior we can reschedule (reroute, resynchronize, and reorder) the jobs of the system to optimize the performance. The optimal schedule is computed by minimizing a performance index $J(k)$ over a prediction horizon N_p. This performance index $J(k)$ is usually given by

$$J(k) = \sum_{j=0}^{N_\mathrm{p}-1} \sum_{i=1}^{n} \sigma_{j,i} \max\Big(x_i(k+j) - d_i(k+j)\ ,\ 0 \Big) + \max_{i=1,\ldots,n} \alpha\, x_i(k+N_\mathrm{p}) + \sum_{l=1}^{L_\mathrm{tot}} \rho_{j,l}\, \eta_l(k+j). \qquad (4)$$

where

$$\eta_l(k+j) = \begin{cases} 0 & \text{for } u_l(k+j) = \varepsilon \\ 1 & \text{for } u_l(k+j) = 0 \end{cases} \qquad (5)$$

is a conventional binary variable, and d is a due date signal. Further α, $\sigma_{j,i}$, and $\rho_{j,l}$ are weighting scalars. The first term is related to the weighted sum of delays with respect to a due date d, the second term of (4) is the makespan over the prediction horizon (that is the total production length over the next N_p jobs), and the third term denotes the penalty for all changes in ordering or synchronization during cycle $k+j$, and

Often we don't have a due date and we like to minimize the global makespan, i.e. the total length of the schedule. Let N_tot be the number of job cycles to be scheduled. Then the aim will be to minimize $\max_i x_i(k + N_\mathrm{tot})$. If N_tot is very large it is usually better to choose a prediction horizon $N_\mathrm{p} \ll N_\mathrm{tot}$, and the criterion will be to minimize (4) where $\alpha = 1$ and $0 \le \sigma_i \ll 1$, $i = 1, \ldots, N_\mathrm{p} - 1$. A major advantage of a small prediction horizon N_p is that the computational complexity of the optimization problem is drastically reduced. In other cases we like to minimize the sum of delays with respect to a due date signal d. We then have $\alpha = 0$ and $\sigma_i = 1$, $\forall\, i$.

The model predictive control problem of minimizing (4) can be recast as an MILP problem (van den Boom et al. (2013)).

3. INTERACTION

The overall system, including the system, control layers and scheduler is a dynamical hybrid system. In our approach each sub-system has its own controller and there is a continuous information exchange between the control and scheduling level. The interface consists of two parts, namely the trajectory generator and the event-observer/predictor.

The trajectory generator (from scheduler to control layer)

The trajectory generator takes care of the communication between events and time driven parts of the system. and it translates high-level output (vector x with the start time and end time of the jobs) into a reference trajectory r that

- satisfies the start time and end time of the operation as given in the high-level output.
- is feasible for the dynamic system it is designed for.
- is feasible with respect to operational constraints.

If there is no feasible trajectory that finishes the operation within the allocated time, the trajectory generator will compute a minimum end time and communicate this to the scheduler. In this case rescheduling has to be done with the adapted processing time. If after some iterations (see Section 4) an optimal and feasible trajectory is found, this trajectory will act as a reference for the underlying controller.

Observer & event predictor (from control to scheduler)

The aim of the event observer is to estimate the continuous time state for the controller. Based on the measured state $\hat{\xi}$ and the model we are able to make a prediction $\hat{\theta}$ on the (remaining) processing times of the operations in the subsystems. These values are needed to solve this scheduling problem and will therefore be communicated with the scheduler.

4. OPTIMIZATION AND CONTROL

In the model predictive control approach for scheduling we have to perform the optimization in real-time based on measurements of the actual state and knowledge of delayed operations (possibly with estimation of the remaining processing times). In this way we can deal with changes in the system parameters, disturbances and failures of the system's components. So if we can expect changes in the occurrence of events due to disturbances or model errors, then we can include this information when determining the optimal schedule for the next cycles of the operation of the system.

Iterative procedure

An advantage of the sequential method is that the two subproblems, finding a schedule and optimizing the control input can be solved independently, while preserving the interaction. A disadvantage is that it is hard to prove convergence of the iterative procedure and optimality of the final solution cannot be guaranteed. Despite the fact that there is no proof of convergence and optimality we believe that for many applications the iterative procedure may improve the result dramatically and although the fact the final solution will still be suboptimal it will be significantly closer to the optimal one. To guarantee a limited computation time we introduce the following stopping conditions:

Stop if

- a previous set of discrete variables matches the current set, or
- the performance index has not decreased for a predefined number of iterations, or
- a predefined maximum number of iterations is reached.

The first condition prevents the system to enter an infinite loop of alternating between two solutions. The second condition states that if we do not find a better schedule for a predefined number of iterations, we are probably close to the optimal value. The third condition is obvious since we can control indirectly the computation time. If the iteration is terminated ,the solution with the lowest performance index so far is said to be most optimal in this set of iterations.

5. APPLICATION TO A RAILWAY TRACK

In this section we will show how the derived methodology can be used to (re)schedule a number of trains over a single track with block sections (de Bruijn, 2017). We consider different train types (intercity, local train, freight train). For safety reasons most railway tracks have block sections with signals to separate the trains. The safety of a railway network can be guaranteed by keeping all trains at a safe distance from each other in such a way that a train can always brake if a preceding train makes an unscheduled emergency stop. For each train we aim for a minimum energy speed profile.

Optimal speed profile for a single train

The sub-system in this case is the train with its local controller to obtain an energy-optimal speed profile, with closed-loop state equation for train i given by:

$$\dot{\xi}_i(t) = f_i(q_{\text{com}}(t), q_{\text{loc},i}(t), \xi_i(t))$$

Here ξ_i consists of the speed and position the train i, the variable $q_{\text{loc},i}$ is the local discrete state reflecting the driving phase and the common discrete state q_{com} indicates the ordering of the trains on the track. The optimal speed profile for a single train on a single track consists of five driving phases (Hansen and Pachl, 2008):

(1) In the first phase is *acceleration*, in which the train uses maximum acceleration $\ddot{\xi} = a_{\text{acc}}$, until the train's maximum allowed speed is achieved, at which point the train will go on to the next phase.
(2) The second phase is *cruising*, in which the train keeps the maximum allowed speed achieved in the previous phase $\dot{\xi} = v_{\max}$.
(3) The third phase is *coasting*, in which no traction effort is present and the train drives only against the resistance forces. $\ddot{\xi} = -a_{\text{coa}}$
(4) The fourth phase is *braking*, involves maximum braking after coasting or cruising. A train must break in order to safely reach the station. This point on the track where the braking phase starts is determined by the braking coefficient and the speed of the train $\ddot{\xi} = -a_{\text{bra}}$, such that the train's speed is equal to zero at the final destination.
(5) The fifth phase is *at-rest*, which is when the train has stopped: $\dot{\xi} = 0$.

Note that in this study we neglect the resistance force in the acceleration and braking phase. The basic optimal speed profile consists of acceleration phase, cruising phase and braking phase. The different phases can be recognized in Figures 2 and 3. Note that the traveled distance is the integral of the speed (size of the gray area). If the resulting arrival time is earlier than the planned arrival time, the coasting phase will be added to arrive at the destination at the desired arrival time.

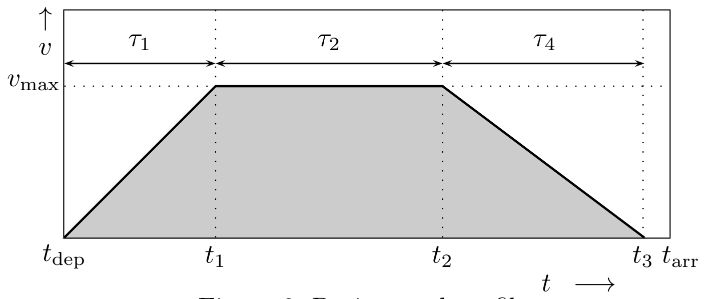

Figure 2: Basic speed profile.

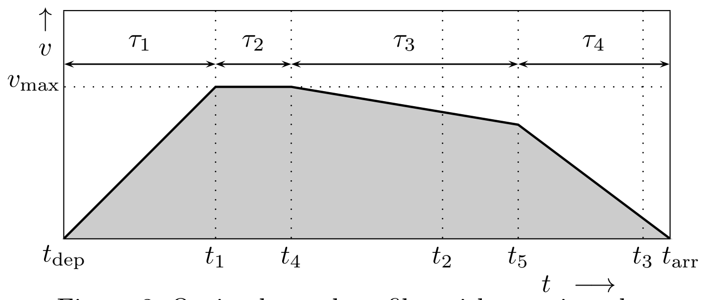

Figure 3: Optimal speed profiles with coasting phase.

Let t_{dep} be the planned departure time and t_{arr} be the planned arrival time of a train. Further, let $v_{\max}$ be the maximum allowed speed on the track, then the duration of the acceleration phase and the traveled distance will be given by

$$\tau_1 = t_1 - t_{\text{dep}} = \frac{v_{\max}}{a_{\text{acc}}} \text{ ,and } s_1 = \frac{v_{\max}^2}{2a_{\text{acc}}},$$

respectively. Let $v_{\max}$ be the maximum allowed speed on the track, then the duration of phase 4 and the traveled distance will be given by

$$\tau_4 = t_3 - t_2 = \frac{v_{\max}}{-a_{\text{bra}}} \text{ , and } s_4 = \frac{v_{\max}^2}{-2a_{\text{bra}}}$$

If the distance s_{tot} between two station is larger than $s_1 + s_4$ the minimal time in cruising phase will be given

$$\tau_2 = t_2 - t_1 = \frac{s_{\text{tot}} - s_1 - s_4}{v_{\max}}$$

If the desired arrival time t_{arr} is larger than t_3 a coasting phase will be introduced. The starting time t_4 and end time t_5 of the coasting phase is chosen such that the braking phase ends in the final destination, in other words the gray areas in the Figures 2 and 3 will be equal in size (Recall that the integral of the speed is equal to the total distance). If the distance s_{tot} between two stations is smaller than s_1+s_4, the cruising phase and coasting phase are skipped and the braking phase starts before the train reaches the maximum speed.

Multiple trains and signaling

Consider Figure 4. If a train is in a black section, it is protected by a stop signal (Pachl, 2002). In front of the previous block section a yellow approach signal is in place to ensure that the train approaches the red stop signal with a moderate speed. In front of the block section before the yellow approach signal, a green signal given and the train can proceed that block with maximum allowable speed.

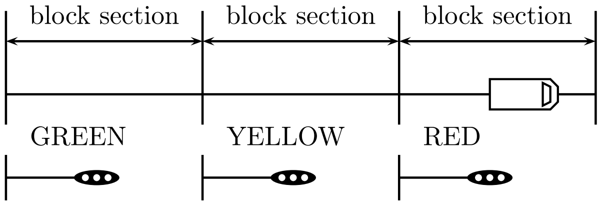

Figure 4: Block sections and signaling.

For safety reasons due to signaling, the intercity trains maintain a distance of at least two block sections (because we want these fast trains to always see a green signal) where the distance between intercity and a slow freight train is only one block section (the speed limit introduced by a yellow signal does not affect the already low maximum speed).

Let $x'_{i,\ell}$ be the time that train i enters the ℓth block section, and let $x'_{j,m}$ be the time that train j enters the mth block section. Let the max-plus variable $u_{i,j}$ determine the order between the trains i and j (for $u_{i,j}$=0 train j follows train i, for $u_{i,j} = \varepsilon$ train i follows train j). Then we have

$$x'_{i,\ell} \geq x'_{i,\ell+p_i} + u_{i,j}$$
$$x'_{j,\ell} \geq x'_{i,\ell+p_j} + \bar{u}_{i,j}$$

where $\bar{u}_{i,j}$ is the adjoint value of $u_{i,j}$ and where $p_i = 2$ if train i is an intercity train and $p_i = 2$ if train i is an intercity train.

Local interface and controller

The ith interface will compute, based on the state ξ_i and discrete state q, the optimal speed profile for the ith train. The profiles will be chosen such that the intercity trains always meet a green signal and the freight trains will always meet a yellow signal. Furthermore, the ith interface will make an estimate $\hat{\theta}_i$ of the time that train i will stay in the present block section.

The planner fits feasible trajectories for the trains with information from the scheduling level about the departure and arrival events of the preceding trains. Further the following rules for the planner given are:

- minimize the total run time.
- trains cannot depart or arrive before the time given by the normal schedule.
- trains are delayed at the departure station until they can run freely at maximum speed over the track i.e. the headway is adjusted such that the minimal distance between trains is equal to the length of the block.

The first item states that a train should be going to coast as early as possible in order to minimize energy consumption. The second item is to ensure that a train leaves the departure station as soon as it is allowed by the constraints. The third item results in a basic speed profile which is energy-optimal.

Based on the discrete states q_{com}, $q_{\text{loc},i}$, and the measured position/speed y_i, the ith local controller will use a model-based control strategy to follow the optimal speed profile r_i for train i computed by the ith interface.

Rescheduling

The (re)scheduler will (re)order the trains over the track, based on the running times estimated by the interface, in such way that the sum of delays is minimized using the techniques of Kersbergen et al. (2016).

6. CASE STUDY

In this case study we consider a small part of a large railway network. A corridor is considered that consists of three stations A, B, and C. We study only the train movements in one direction; so all trains run from station A, via station B to station C. We assume all trains make a stop at stations A and C, and at station B only local trains make a scheduled stop. The track between the stations consists of a single line per direction, and so trains cannot be overtaken once they are on the track. The total track with a length of 15000 meters is divided into 10 block sections (each block section has a length of 1500 meters) to guarantee a safe distance between all trains. The length of the block sections is larger than the longest worst case braking distance of all trains running over the track (Hansen and Pachl, 2008).

We assume three different train types on the track with different characteristics in maximum speed, acceleration, and braking behavior. The parameters of the rolling stock are given in Table 5 and comes from the typical values for the different train types in Profillidis (2014). One of the trains is a local train and has an extra stop at station B.

train type	freight train	local train	intercity
a_{acc} $[m/s^2]$	0.19	0.52	0.46
a_{coa} $[m/s^2]$	-0.0024	-0.051	-0.045
a_{bra} $[m/s^2]$	-0.19	-0.52	-0.52
$v_{\max}$ $[m/s]$	16.7	36.1	36.1

Table 5: Parameters per train type

In Table 6 the nominal timetable of the trains on the track, including planned departure time in A, and planned arrival time in C. The local train also has a planned stop in station B. The simulation considers four intercity trains, two local trains and one freight train.

Train	Planned departure [min]	Planned arrival [min]	Train type
1	30	39	Intercity
2	32	43	Local train
3	36.5	45.5	Intercity
4	38.5	55	Freight train
5	53	62	Intercity
6	60	69	Intercity
7	62	73	Local train

Table 6: Timetable and train type

We start a simulation at time $t = 25$ min. Assume that train 1 has 8.5 minutes delay and train 2 has 3.5 minutes delay. The first schedule is computed using the nominal running times of the trains. Due to the delay the original train order 1-2-3-4-5-6-7 will not give the minimal sum of delays. With a scheduling step we obtain the optimized order 2-4-1-3-5-6-7. With this new schedule the interface will compute the corresponding speed profiles, and based on these profiles, the updated departure and arrival times. After this first iteration the sum of delays is equal to 35.5 minutes. The updated running times of the trains are sent back to the scheduler for a second iteration step. The new schedule (2-3-1-4-5-6-7) with updated speed profiles and corresponding departure and arrival times gives a sum of delays of 31.4 minutes. A third iteration leads to another schedule (2-3-4-1-5-6-7) with the same sum of delays of 31.4 minutes, and a fourth schedule gives us the original schedule again with sum of delays 35.5 seconds. This terminates the optimization procedure. Based on the fact that the second schedule 2-3-1-4-5-6-7 is closer to the nominal schedule than the third schedule 2-3-4-1-5-6-7, we decide to pick the ordering from iteration 2 as most optimal schedule.

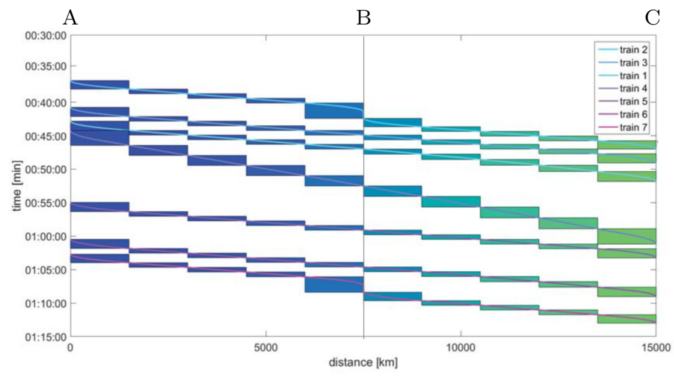

Figure 7. Time-distance diagram (order: 2-3-1-4-5-6-7)

7. DISCUSSION

In this paper we have tackled the combined scheduling-control problem for hybrid system with a (semi-)cyclic behavior by using an iterative algorithm in which the scheduling and control parts are optimized in an iterative way. First a provisional schedule is computed. Based on this schedule the interface will compute feasible trajectories for the local controllers. Based on measurements of the sub-system an estimation of the remaining operation times will be computed. They are sent to the scheduler that uses the estimates to re-optimize the overall schedule. In a case study the scheduling and control of rail traffic over a single track with seven trains has been optimized.

In future research we will study the optimality of the solution of the iterative procedure.

REFERENCES

Alirezaei, M., van den Boom, T., and Babuška, R. (2012). Max-plus algebra for optimal scheduling of multiple sheets in a printer. In *American Control Conference 2012*, 1973–1978. Montreal, Canada.

Baccelli, F., Cohen, G., Olsder, G., and Quadrat, J. (1992). *Synchronization and Linearity*. Wiley, New York.

Bouquard, J., Lenté, C., and Billaut, J. (2006). Application of an optimization problem in max-plus algebra to scheduling problems. *Discrete Applied Mathematics*, 154(15), 2064–2079.

de Bruijn, J. (2017). Abstraction of hybrid systems with an application in railway management. MSc Thesis, Delft University of Technology, The Netherlands.

Hansen, I.A. and Pachl, J. (2008). *Railway Timetable & Traffic: Analysis - Modelling - Simulation*. Eurailpress, Hamburg, Germany.

Houssin, L. (2011). Cyclic jobshop problem and (max,plus) algebra. In *IFAC World Congress*, 2717–2721.

Kersbergen, B., Rudan, J., van den Boom, T., and De Schutter, B. (2016). Towards railway traffic management using switching max-plus-linear systems - structure analysis and rescheduling. *Discrete Event Dynamic Systems: Theory and Applications*, 26(2), 183–223.

Maciejowski, J. (2002). *Predictive Control with Constraints*. Prentice Hall, Harlow, UK.

Mutsaers, M., Özkan, L., and Backx, T. (2012). Scheduling of energy flows for parallel batch processes using max-plus systems. In *Proceedings of the 8th IFAC International Symposium on Advanced Control of Chemical Processes 2012*, 176–181.

Nambiar, A. and Judd, R. (2011). Max-plus-based mathematical formulation for cyclic permutation flow-shops. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2, 85–97.

Pachl, J. (2002). *Railway Operation and Control*. Vtd Rail Pub, Mountlake Terrace (USA).

Profillidis, V.A. (2014). *Railway Management and Engineering*. Ashgate, 4th ed.

van den Boom, T. and De Schutter, B. (2006). Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice*, 14(10), 1199–1211.

van den Boom, T., Lopes, G., and De Schutter, B. (2013). A modeling framework for model predictive scheduling using switching max-plus linear models. In *52st IEEE Conference on Decision and Control (CDC 2013)*. Florence, Italy.

Yurdakul, M. and Odrey, N. (2004). Development of a new dioid algebraic model for manufacturing with the scheduling decision making capability. *Robotics and Autonomous Systems*, 207–218.