Abstraction of Hybrid Systems

With an application in railway management

J.H.C. de Bruijn

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Abstraction of Hybrid Systems
With an application in railway management

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J.H.C. de Bruijn

July 3, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Disturbances are an important factor in the performance of complex large scale systems. These complex large scale systems can be modeled by a broad class of hybrid dynamical systems. The current practice of controlling such processes is one way, from the scheduling level to the control level. The performance of these complex systems can be improved by allowing an exchange of information between both the scheduling and control level.

Although the techniques derived in this thesis are applicable to general class of hybrid systems, this thesis focuses on a railway network. This network, or a part of it, is modeled so that information at regular time instances can be sent to a dynamic scheduler which outputs an optimal schedule back to the control level, based on the current conditions in the network.

To this end, the railway system is abstracted into a Discrete-event system (DES). Essentially, the abstraction removes the continuous dynamics of a hybrid dynamical model and replaces them with discrete-events. All these discrete-events together form the railway network as a DES. Furthermore, the DES is modeled in max-plus algebra such that a Max-plus linear (MPL) is obtained.

The scheduling problem is solved by a Model predictive control (MPC) scheme which is solved by a Mixed-integer linear programming (MILP) problem. For this scheme, the MPL model is converted into a Switching max-plus linear (SMPL) model which is written as a set of mixed-integer linear constraints. In contrast to the scheduler which only contains a macroscopic model of the railway network, the trajectory planner includes microscopic constraints. Therefore, to complete the scheduling, the schedule found by the scheduler is iterated between the scheduler and a trajectory planner for further improvements. A part of a railway network is studied in a case study, showing that updating the schedule results in less delay in case a delay is introduced.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to thank my supervisor Ton van den Boom for his support and enthusiasm during the meetings.

I am grateful to my parents, brothers and sisters.

Delft, University of Technology — J.H.C. de Bruijn
July 3, 2017

De goede Herder gaat mij voor heel het leven door.

naar Psalm 23

Chapter 1

Introduction

1-1 Motivation

Complex large scale systems play an important role in our ever-expanding industrial society. Furthermore, when these processes are part of a growing demand that involves up scaling of the plants which consist of not only larger plants but also highly complex processes. When a small initial delay is introduced in a process, this can later have a large effect on other connected components of the plant. This effect can vary from a delay at the end process or even breakdown of the plant which is obviously highly inefficient. Therefore, control plays an important role in efficiently handling these processes and letting them work together, aiming to minimize cost and maximize profit. The current approach is that plants follow a predefined schedule where lower level controllers take care of keep track of the main schedule. Instead of constantly optimizing the whole process, the plants only have a single optimized schedule for the process, that has to deal with all disturbances. The main schedule must therefore be robust against disturbances and is, most of the time, not an ideal schedule for the current situation of the plant. Another drawback of a rigid schedule is that it cannot cope with a situation that is not taken into account in the robust timetable and this could result in a breakdown or at least decreased performance.
An integrated approach for controlling the plant processes is desirable, but causes large and complex optimization problems for which solutions are hard to find. Therefore the authors of [1] propose an integrated sequential approach where the control problem is broken down and solved sequentially for the different sub problems.

1-2 Research goals

This thesis is part of an ongoing research [1], the aim of which is to integrate both scheduling and control in industrial plants. The focus in this thesis is on abstracting a dynamical process at a certain time into a Discrete-event system (DES). This DES, made with real time information, is used for scheduling and possible rescheduling operations such that the main schedule can be adapted to the current real time situation. The abstracted system represents a macroscopic model of the plant, however in order to include microscopic control decisions in the optimized schedule, a trajectory planner calculates the actual trajectories on the basis of the control decisions made by the scheduler. The information of these actual trajectories is returned to the scheduling level to check for any changes in the control decisions made by the scheduler. This process is repeated until the control variables do not change and a previous solution is found. The research question is therefore formulated as:

How can a hybrid dynamical system be abstracted such that the model only contains discrete event timing information and no time-driven dynamics. Furthermore, how can a feasible schedule be obtained and further optimized during operation of the system.
Instead of considering a broad class of hybrid systems, this thesis focuses on an application in railway management because this brings down the complexity of the sub problems, namely abstraction, optimization and further optimization along with a trajectory planner.

1-3 Outline

The outline of the thesis is as follows: first, a background is given on hybrid systems, max-plus algebra and an abstraction technique for hybrid systems in Chapter 2. Following this, Chapter 3 shows the outline of a general sequential interactive model. In Chapter 4 we focus on a railway network to apply the general structure of Chapter 3 to. Next is Chapter 5, where scheduling is described with Model predictive control (MPC). In Chapter 6, a case study of a small railway network is executed in order to optimize and reschedule the nominal timetable in case of a delay. Finally, the conclusions are given in Chapter 7.

Chapter 2

Background

In this chapter, we introduce two special classes of hybrid systems in the first section: Piecewise affine (PWA) and Mixed-logical dynamical (MLD). Since the first class is the continuous-time equivalent of the later discussed, event-driven Switching max-plus linear (SMPL) systems. The latter class MLD systems is introduced because the PWA framework can be recasted in a MLD form and solved by a Model predictive control (MPC) optimization. Then Max-plus linear (MPL) and SMPL systems are introduced with the help of max-plus algebra, since the aim of this thesis is to write the abstraction of a hybrid system into a SMPL framework. Finally, we introduce the principles of train operation, train dynamics and safety since that is chosen to be the application for this thesis work.

2-1 Hybrid Systems

This introduction is based on [2], who gives a systematic overview of hybrid systems. Hybrid systems are a combination of both time-driven and event-driven dynamics in one framework. For example a gearshift in a car. The car can run in different modes, say gear one, two etc. Each mode has its own time-driven dynamics. A hybrid system contains a coupling between the time-driven dynamics which are formed by differential and difference equations and event-driven dynamics which are discrete transitions to another time-driven system.

There are multiple models to describe such a system, each with their own advantage such as stability, control and analysis techniques. These models can be divided in two categories, the explicit model description and the implicit model description. In the following section, first an explicit model is introduced because the graphical structure gives a clear view of the hybrid nature i.e. the switching between time-driven dynamics and the event-driven dynamics. Then two implicit models are presented, these are piecewise affine systems and mixed logical dynamical systems. This latter category is more general than the explicit model description and cannot be directly solved for an arbitrary model. The reason for using an implicit approach is that it is often easier to implement constraints [3] and the framework is more general.

2-1-1 Hybrid automata

A hybrid automaton [4] is a general model description for a hybrid system. In this description, a hybrid system is modeled as a finite automaton [2]. A finite automaton consists of edges, places and labels. In Figure 2-1, the labels (a,b) are labeling the edges which are indicated by arrows and the places are the vertices (l_1, l_2).

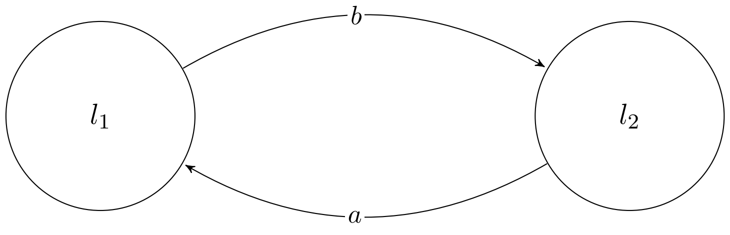


Figure 2-1: Finite automaton

A hybrid automaton combines the continuous and discrete dynamics in a graph in the same way as the finite automaton as shown in Figure 2-2. The discrete transitions are the jumps from block $q_0 \to q_1$ and vice versa indicated by the vertices in the graph. The continuous dynamics in the blocks (q_0, q_1) are the modes of operation within a certain range. This range is determined by Guards and Invariants. The Guards, given as G(q_*,q_*), provide a structure where a discrete transition takes place i.e. a guard condition says when a transition to another mode is enabled and the hybrid system can jump to the next mode. The Invariants, given as Inv(q_*), determine when the system must take the transition i.e. forced to another mode if the invariant is no longer satisfied, this in contrast to the guard which only enable the transition and does not force the system to jump.
The Resets, given as R(q_*,q_*), is a map how the continuous state is reset from one mode to another.

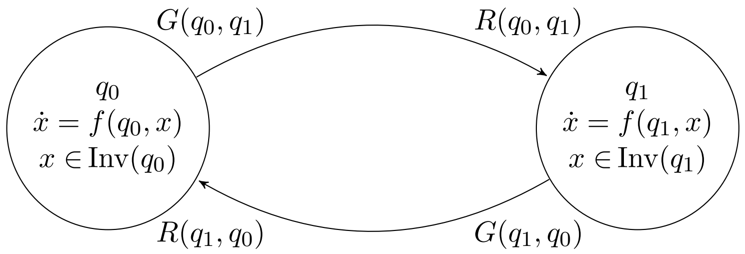


Figure 2-2: Hybrid automaton

The formal definition of a hybrid automaton is adopted from [5]

Definition 2.1. *A Hybrid automaton H is a collection $H = (Q, X, f, D, E, R)$, where*

The state of the hybrid automaton is $(q, x) \in Q \times X$, TX is the tangent bundle on X and $P(X)$ is the power-set of X which is the set of all the subsets of X.

The execution of a hybrid automaton consist of three hybrid signals. The hybrid time set, the discrete state q and the continuous states x. Formally defined as:

Q	is a finite set of discrete variables;
X	is a finite set of continuous variables;
$f : Q \times X \to TX$	is a vector field;
$\text{Init} \subseteq Q \times X$	is a set of initial states;
$D : Q \to P(X)$	is a domain;
$E \subseteq Q \times Q$	is a set of edges;
$G : E \to P(X)$	is the guard condition;
$R : E \times X \to P(X)$	is the reset map;

- τ a hybrid time set
- $q = \{q_i\}_{i=0}^N$ with $q_i : I_i \to Q$
- $x = \{x_i\}_{i=0}^N$ with $x_i : I_i \to X$

with initial condition $(q_0, x_0) \in \text{Init}$ providing that the system start at a desired position.

The discrete evolution given by the set of Edges, Guard condition and Reset map determine when the transition is enabled and how the state is related through the reset map is given:

for all i, $e = (q_i(\tau_i'), q + i + 1(\tau_{i+1})) \in E)$

- $x_i(\tau_i') \in G(e)$ and
- $(x_i(\tau_i'), x_{i+1}(\tau_{i+1})) \in R(e)$

The continuous evolution is given by a differential equation, where the discrete state remains constant during the evolution over the continuous vector field. Furthermore, the continuous part is allowed to flow within the domain which is indicated by the invariant. This results in:

for all i

- $q_i : I_i \to Q$ is constant, that is, $q_i(t) = q_i(\tau_i)$ for all $t \in I_i$;
- $x_i : I_i \to X$ is the solution of the differential equation $\dot{x} = f(q_i(t), x(t))$ on the interval I_i, with initial condition $x_i(\tau_i)$ at τ_I;
- for all $t \in [\tau_i, \tau_i']$ it holds that $x_i(t) \in \text{Inv}(q_i(t))$

2-1-2 PWA systems

The class of piecewise affine systems [6] forms a simple extension on a linear system and can still model hybrid phenomena. The system is described by

$$\begin{aligned} x(k+1) &= A_i x(k) + B_i u(k) + f_i \\ y(k) &= C_i x(k) + D_i u(k) + g_i \end{aligned} \qquad \text{for} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \Omega_i \tag{2-1}$$

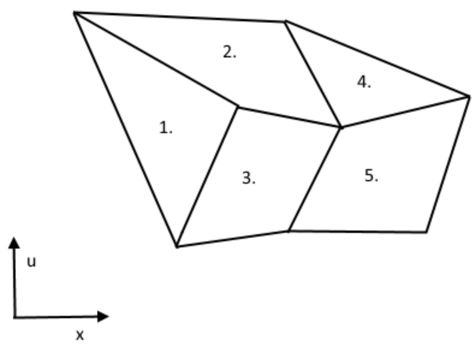


Figure 2-3: Polyhedral partition of the state and input space

Where Ω_i is a finite convex polyhedral partition of the state/input space. With finite index $i \in \mathcal{I}$ and with discrete time step k, $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$ and $y(k) \in \mathbb{R}^l$

An example of an arbitrary piecewise affine system will now be given with the help of a graph.

Example 2.2.

The graphical representation shows how the state and input space can be partitioned with $x(k), u(k) \in \mathbb{R}$

In figure 2-3, the numbers (1.-5.) indicate the polyhedrons $\Omega_1 - \Omega_5$ which are modeled by a finite number of inequalities. These five regions are each associated with a linear dynamical system. So this figure represents the mathematical equation 2-1 where the first part is the dynamical system and the latter in which this system is active.

2-1-3 MLD systems

This framework describes the continuous variables by linear equations and the discrete variables are expressed by logical statements. The logical variables are associated with a binary variable (0,1) after which the logical statements can translated into linear inequalities. However, the system contains both logic and continuous dynamics. This means that the resulting system contains mixed-integer linear inequalities.

The most important equations will now be shown, how these are obtained can be found [7].

$$[f(x) \leq 0] \Leftrightarrow [\delta = 1] \text{is true if and only if} \begin{cases} f(x) \leq M(1-\delta) \\ f(x) \geq \varepsilon + (m-\varepsilon)\delta \end{cases} \tag{2-2}$$

With $f : \mathbf{R}^n \to \mathbf{R}$ and $x \in \mathcal{X}$ where $\mathcal{X}$ is the interval for which the function f is defined. Where M and m are the maximum and minimum respectively of the function $f(x)$ on the defined interval $\mathcal{X}$. The variable ε is a small tolerance introduced because only non-strict inequalities are allowed in linear programming.

The nonlinear product $\delta(k)f(x(k))$ can be represented by linear inequalities by introducing an auxiliary variable $z(k)$.

$$z(k) \leq M\delta(k) \tag{2-3}$$
$$z(k) \geq m\delta(k) \tag{2-4}$$
$$z(k) \leq x(k) - m(1 - \delta(k)) \tag{2-5}$$
$$z(k) \geq x(k) - M(1 - \delta(k)) \tag{2-6}$$

with these equations, a system in the form

$$x(k+1) = \begin{cases} f_1(x) \text{ if } x(k) \geq 0 \\ f_2(x) \text{ if } x(k) < 0 \end{cases} \tag{2-7}$$

can be transformed if $\delta(k) \in \{0, 1\}$ is defined as $[\delta = 1] \Leftrightarrow [f \geq 0]$.
Then the equation (2-7) can be represented by

$$x(k+1) = f_1(x(k))\delta(k) + f_2(x(k))(1 - \delta(k)) \tag{2-8}$$

This system representation contains the nonlinear product $\delta f(x(k))$ with the help of the equations (2-3-2-6) this can be translated into linear constraints.

The generalized system description is of the form:

$$\begin{aligned} x(k+1) &= Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \\ y(k) &= Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \\ E_1x(k) + E_2u(k) &+ E_3\delta(k) + E_4z(k) \leqslant g \end{aligned} \tag{2-9}$$

where $\delta(k)$ is $\in \{0, 1\}$ and $x(k) = [x_r^T(k)x_b^T(k)]^T$ with $x_r(k) \in \mathbb{R}^{n_r}$ the continuous and $x_b(k) \in \{0, 1\}^{n_b}$ the binary variables. This structure is also applied to y and u.

An example from [7] will be given to illustrate the above

Example 2.3.

$$x(k+1) = \begin{cases} 0.8x(k) + u(k) & \textit{if } x(k) \geq 0 \\ -0.8x(k) + u(k) & \textit{if } x(k) < 0 \end{cases} \tag{2-10}$$

Where $x(k) \in [-10, 10]$, and $u(k) \in [-1, 1]$. The form is the same as in equation 2-7 and thus by defining $\delta(k) \in \{0, 1\}$ by $[\delta = 1] \Leftrightarrow [f \geq 0]$ the equation can be written as

$$\begin{aligned} x(k+1) &= (0.8x(k) + u(k))\delta(k) + (-0.8x(k) + u(k))(1 - \delta(k)) \\ &= 1.6x(k)\delta(k) - 0.8x(k) + u(k) \end{aligned} \tag{2-11}$$

To write equation 2-11 subject to the linear constraints (2-2-2-6) in the general form, define $z(k) = \delta(k)x(k)$

$$x(k+1) = -0.8x(k) + u(k) + 1.6z(k) \tag{2-12}$$

$$\begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ -1.6 \\ 1.6 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 10 \\ -10-\varepsilon \\ -M \\ M \\ M \\ -M \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta(k) + \begin{bmatrix} 0 \\ 0 \\ I \\ -I \\ I \\ -I \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} z(k) \leq \begin{bmatrix} 10 \\ -\varepsilon \\ 0 \\ 0 \\ M \\ -M \\ 10 \\ 10 \\ 1 \\ 1 \end{bmatrix} \tag{2-13}$$

2-2 Discrete event systems

The class of Discrete-event system (DES) describe systems where the state of the dynamical system does not change by the tick of the clock, but by events. Since the events are generally asynchronous it is therefore not efficient to use time as the element that drives the dynamical system. So, where the variable k in time-driven systems stands for the time, the k in event-driven systems is the event counter. From now on in this thesis the variable k refers to the event counter in event-driven systems.
Moreover, we define the DES as a system which is event driven with a discrete state space. The state space is some discrete set of events and whenever an event occurs this may cause a change in the state as opposed to discrete time systems where the state may change each discrete time instant. A DES consists of a finite number of resources shared by different users called jobs. These jobs all contribute to the same goal which is completing the product. The dynamics of a DES ca typically be described by synchronization, routing and ordering. Synchronization is needed if a job requires different resources at the same time. If a job can follow different routes through the system, then routing fixes the path for the job to follow. In the end, ordering plays a role when different jobs take the same route at the same time, so the order between the jobs must be determined.
Most of the models are nonlinear in traditional algebra, but there is a class of DES that can describe nonlinear behaviour in a linear manner in the max-plus algebra. This class contains only synchronization with a fixed order and a fixed route. With the help of SMPL models, the synchronization, ordering and route can be changed by changing the structure of the system matrix. As will be come clear in the next section, systems written in max-plus algebra are particular suitable for systems that behave cyclically and the tools from linear algebra can be applied.

2-2-1 Max-plus algebra

Max-plus algebra allows one to rewrite a non-linear system in conventional algebra to write it in a linear form in the max-plus algebra [8]. Max-plus algebra uses two operations

$$a \oplus b = \max(a, b) \tag{2-14}$$

and

$$a \otimes b = a + b \tag{2-15}$$

In addition, the max-plus zero element is defined as $\varepsilon = -\infty$ and the max-plus identity element is defined as $e = 0$. This forms the max-plus semi-ring which is the set $\mathbb{R}_{\max} = \{\varepsilon\} \cup \mathbb{R}$. Matrix operations in a max-plus fashion are defined as follows

for $A, B \in \mathbb{R}_{\max}^{n\times m}$

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij}) \tag{2-16}$$

for $A \in \mathbb{R}_{\max}^{n\times l}$ and $B \in \mathbb{R}_{\max}^{l\times m}$

$$[A \otimes B]_{ik} = \bigoplus_{j=1}^{l} a_{ij} \otimes b_{jk} = \max(a_{ij} + b_{ij}) \quad \text{for} \quad i \in n \quad \text{and} \quad k \in m \tag{2-17}$$

To illustrate equation 2-16 with an example, consider the matrices A, B both in $\mathbb{R}_{\max}^{2\times 2}$

Example 2.4. *let*

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \textit{ and } B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Then

$$A \oplus B = \begin{bmatrix} a_{11} \oplus b_{11} & a_{12} \oplus b_{12} \\ a_{21} \oplus b_{21} & a_{22} \oplus b_{22} \end{bmatrix} = \begin{bmatrix} \max(a_{11}, b_{11}) & \max(a_{12}, b_{12}) \\ \max(a_{21}, b_{21}) & \max(a_{22}, b_{22}) \end{bmatrix}$$

With the same A and B matrices, equation 2-17 reads as

$$\begin{aligned} A \otimes B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} &= \begin{bmatrix} (a_{11} \otimes b_{11}) \oplus (a_{12} \otimes b_{21}) & (a_{11} \otimes b_{12}) \oplus (a_{12} \otimes b_{22}) \\ (a_{21} \otimes b_{11}) \oplus (a_{22} \otimes b_{21}) & (a_{21} \otimes b_{12}) \oplus (a_{22} \otimes b_{22}) \end{bmatrix} \\ &= \begin{bmatrix} \max(a_{11} + b_{11}, a_{12} + b_{21}) & \max(a_{11} + b_{12}, a_{12} + b_{22}) \\ \max(a_{21} + b_{11}, a_{22} + b_{21}) & \max(a_{21} + b_{12}, a_{22} + b_{22}) \end{bmatrix} \end{aligned}$$

The conventional linear algebra rules can thus directly be applied to the matrix multiplication and addition in max-plus algebra. The only difference is the definition of the operators. In max-plus $\oplus, \otimes$ and in conventional algebra $+, \times$ respectively.

With the basic elements ε and e defined, the max-plus zero $\mathcal{E} \in \mathbb{R}_{\max}^{n\times m}$ and identity matrices $E \in \mathbb{R}_{\max}^{n\times m}$ defined as follows:

$$[E]_{i,j} = \begin{cases} e, & \text{if } i = j \\ \varepsilon, & \text{if } i \neq j \end{cases} \tag{2-18}$$

$$[\mathcal{E}]_{i,j} = \varepsilon, \ \forall\ i,j \tag{2-19}$$

Finally, let $u \in \mathbb{B}_\varepsilon = \{e, \varepsilon\}$ be a max-plus binary control variable. Its adjoint $\bar{u} \in \mathbb{B}_\varepsilon$ is defined as:

$$\bar{u} = \begin{cases} e, & \text{if } u = \varepsilon \\ \varepsilon, & \text{if } u = e \end{cases} \tag{2-20}$$

2-2-2 Solving linear systems and its characteristics

The equation

$$x = A \otimes x \oplus b \tag{2-21}$$

with $A \in \mathbb{R}^{n\times n}_{max}$ and $b \in \mathbb{R}^{n}_{max}$ can be solved by the vector

$$x = A^* \otimes b \tag{2-22}$$

where, $A^* \overset{\text{def}}{=} E \oplus A^+ = \bigoplus_{k=0}^{\infty} A^{\otimes k}$

This equation (2-22) is an important result in solving max-plus linear systems.

Eigenvalues and eigenvectors are defined in the same manner as conventional algebra. If for a matrix $A \in \mathbb{R}^{n\times n}_{\max}$, a vector $v \in \mathbb{R}^{n}_{\max}$ and a scalar $\lambda \in \mathbb{R}_{\max}$ it holds that

$$A \otimes v = \lambda \otimes v \tag{2-23}$$

with $v \neq \mathcal{E}$, then λ is an eigenvalue of A and v is an eigenvector of A. An eigenvalue in a max-plus system represent the minimal time between two events and the eigenvectors show the steady state behaviour of the system. If a matrix is irreducible, then the communication graph related to this matrix is strongly connected and there is only one unique eigenvalue.

Now the cyclicity theorem of max-plus algebra will be introduced. This theorem provides a way to obtain unique eigenvalues for irreducible matrices.

Theorem 2.5. *[9] Let $A \in \mathbb{R}^{n\times n}_{max}$ be an irreducible matrix with eigenvalue λ and cyclicity $\sigma = \sigma(A)$. Then there is an N such that*

$$A^{\otimes(k+1)} = \lambda^{\otimes\sigma} \otimes A^{\otimes k} \tag{2-24}$$

for all $k \geq N$.

2-2-3 Max-plus linear systems

As stated before, a certain class of discrete-event systems can be rewritten with max-plus algebra and then become linear in this algebra. This is a class of systems where only synchronization occurs. Synchronization can be thought of as different resources must be available at the same time. For example, in a railway environment, one train must wait for another train to let passengers transfer. Or in a chemical plant, a reaction takes place in a reactor vessel but this reaction starts only when all resources are available. These systems can be described by a model of the form

$$\begin{aligned} x(k) &= A \otimes x(k-1) \oplus B \otimes u(k) \\ y(k) &= C \otimes x(k) \end{aligned} \tag{2-25}$$

where, $A \in \mathbb{R}_{\max}^{n\times n}$, $B \in \mathbb{R}_{\max}^{n\times m}$, $C \in \mathbb{R}_{\max}^{p\times n}$ and x the state, u the input and y the output vectors.

This form relates events from the current cycles to events in the previous cycle. The input $u(k)$ contains information about when the events happen in cycle k. If the input is omitted, then events happen as soon as possible. A railway system is a simple example to show how the input acts as a timetable reference. The input is the timetable for when trains are leaving and arriving at stations. The trains cannot leave or arrive earlier than the scheduled times in the timetable since the input is connected to the model with a ($\oplus$, max-operation). Meaning that a train is only allowed to depart from a station if the departure time is greater than or equal to the arriving time from a previous station plus the dwell time or the scheduled departure time provided by $u(k)$.

2-2-4 Switching max-plus linear systems

In a MPL system the structure is fixed. This means that the order and synchronization of events cannot be changed. To break and change this structure SMPL systems [10] are introduced.

$$\begin{aligned} x(k) &= A^{l(k)} \otimes x(k-1) \oplus B^{l(k)} \otimes u(k) \\ y(k) &= C^{l(k)} \otimes x(k) \end{aligned} \tag{2-26}$$

where $l(k) \in \{1, .., n_m\}$ is the mode for the k-th event step and n_m is the number of modes. Furthermore, each mode contains max-plus linear model as in equation 2-25.

The switching variable $z(k) \in \mathbb{R}_{\max}^{n_z}$ is determined by function which may depend on the previous state $x(k-1)$, the previous mode $l(k-1)$, input $u(k)$ and a control variable $v(k)$.

$$z(k) = \psi(x(k-1), l(k-1), u(k), v(k)) \in \mathbb{R}_{\max}^{n_z} \tag{2-27}$$

To obtain the mode $l(k)$, the domain $\mathbb{R}_{\max}^{n_z}$ is partitioned in n_m subsets $\mathcal{Z}_i$, $i = 1, .., n_m$. Now the set in which the variable $z(k)$ is in event k determines the mode $l(k)$. So if $z(k) \in \mathcal{Z}_i$, then $l(k) = i$.

As stated before, the max-plus algebra allows for rewriting the dynamical model of a DES and transforms it into a linear model. This can only be done with systems where only synchronization occurs. Now with the help of breaking the structure in a SMPL system, every mode $l(k)$ contains a MPL model and all n_m modes together form the possible combinations of synchronization, routing and ordering decisions.

2-3 Abstraction of hybrid systems

The models of dynamical systems contain information about the properties of these systems. When analyzing such a system some properties are important for that particular analysis and some are not. System abstraction deals with removing the undesired properties while preserving others. The resulting model belongs to a simpler class than the original model. In the case of this thesis, we aim at removing all dynamics of the hybrid system and describe the evolution of the system by time intervals for which an event takes place. This is the hybrid time set as defined in [11].

2-3-1 Hybrid time trajectories

The evolution of a hybrid system over a certain time is called the hybrid time set. The execution of a hybrid system takes place over time and within this horizon the hybrid system executes a trajectory from an initial position to a desired state. The executions exist of the evolution of continuous dynamics and discrete jumps. So, the hybrid time set exist of some discrete time intervals where the start of the next interval begins at the end of the previous interval such that the time makes no jumps and is continuous.

Definition 2.6. *[11] A Hybrid time set is a sequence $\tau = \{I_1, I_2, ..., I_N\} = \{I_i\}_{i=1}^N$, finite or infinite (i.e. $N = \infty$ is allowed)such that*

- $I_i = [\tau_i, \tau_i']$ *for all* $i < N$;
- *if* $N < \infty$ *then either* $I_N = [\tau_N, \tau_N']$ *or* $I_N = [\tau_N, \tau_N')$; *and*
- $\tau_i \leq \tau_i' = \tau_{i+1}$ *for all* i

where τ indicates the start time of the event and τ' is the end time of the event. With N as the number of events. There can be multiple events at the same time which is indicated by the counter i. Each event is associated with a discrete state.

Example 2.7.

Let $N = 4$, there are 4 intervals $I_1 = [\tau_1, \tau_1']$, $I_2 = [\tau_2, \tau_2']$, $I_3 = [\tau_3, \tau_3']$ and $I_4 = [\tau_4, \tau_4']$ according to definition 2.6. Let the duration of I_0, I_2, I_3 be 5 seconds and I_1 be 0 seconds.

Then figure 2-4 shows how the time τ and the events are related. For this example, the hybrid automaton is thus defined on the closed interval $\tau = [1 - 15]$ seconds. At $\tau = 5$ seconds it can be observed that 3 events are executed at the same time, these are event 1, event 2 and event 3.

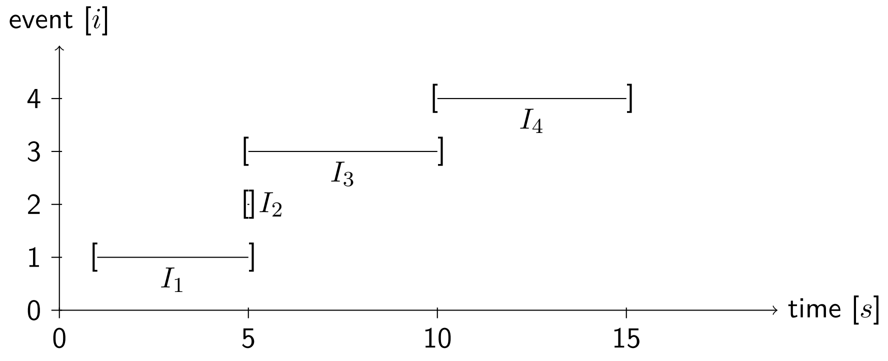


Figure 2-4: Time set of a hybrid automaton

2-4 Asymptotic abstraction hybrid systems

This section is mainly based on [12]. Abstraction can be done using simulation relations for continuous systems or partitioning the state space. The abstraction mapping relates the vector fields of the abstracted system and original system with each other under a surjective map. This precise mapping is not always desirable and particularly in the case where the aim is to map continuous hybrid dynamics to purely discrete dynamics and being only left with the maximum time it takes for the system to make a discrete transition. The authors of [12] also make use of abstracting hybrid dynamical systems by partitioning the state space. This abstraction approach is based on the asymptotic behaviour of a stable dynamical system, the assumptions of this system are that there must exist a finite number of disjoint positive limit sets [13]. For a hybrid system, the limit sets are contained in the guards. Defining the positive limit set it is no longer necessary to make a reachability analysis (which can be very hard due to undecidability [14],[15]). The concept of finite time abstraction is introduced which partition the state space in equivalence classes according to the distance at time T away from a point in the limit set. The abstraction is constructed by the following two definitions.

Definition 2.8. *(Finite-time Equivalence relation [12]) Consider an autonomous system (Z, f), where Z is a compact subset of a smooth manifold M and f a vector field that represent the closed loop dynamics. Let $L^+ = \bigcup_{k=1}^{l} L_k^+, l < \infty$, be the positive limt of (Z,f), where each L_k^+ is simply connected. Two points $z_1, z_2 \in Z$ are said to belong to the same T-equivalence class, and we write $z_1 \sim_T z_2$, if*
1) $z_1 \sim z_2$, and
2) if for some k, $lim_{t\to\infty}$ dist$(\Phi_t(z_1), L_k^+) = lim_{t\to\infty}$ dist$(\Phi_t(z_2), L_k^+) = 0$, then dist$(\Phi_T(z_1), L_k^+)$ $=$ dist$(\Phi_T(z_2), L_k^+)$.

Where $\Phi_t(z_{(1,2)})$ is the flow of system (Z, f) passing though point $z_{(1,2)} \in Z$ at time $t = t_0$. The distance of a point to a subset is dist(point, Subset)

Now, $L_k^+ + \mathcal{B}_d$ is used to define the set $\{x + y | x \in L_k^+, y \in \mathcal{B}_d\}$ with $\mathcal{B}_d$ the ball of radius d.

Definition 2.9. *(Finite Time Abstraction [12]) Consider a system (Z,f), where Z is a compact subset of a Banach manifold M, and let $\Phi_t(p)$ is the flow of f from $p \in M$. Suppose that (Z, f) has a positive limit set $L^+ = \bigcup_{i=1}^{l} L_i^+, l < \infty$. The finite-time T-abstraction of (Z, f) is a (set valued) map, that associates each point $p \in M$ to the set $L_k^+ + \mathcal{B}_d$, where k is such* *that $lim_{t\to\infty}$ $dist(\Phi_t(p), L_k^+) = 0$, $d = dist(\Phi_T(p), L_k^+)$, and $\mathcal{B}_d$ is the ball of radius d centered at the origin.*

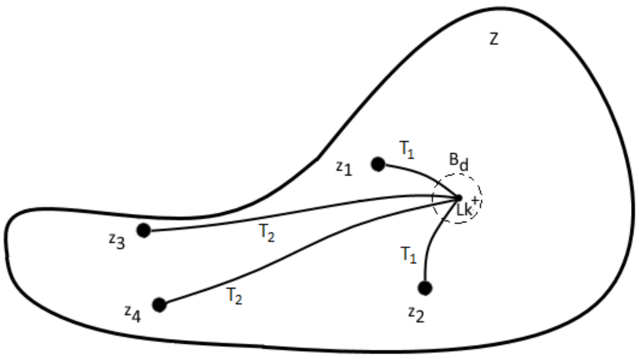


Figure 2-5: Time equivalences of points in $Z \subset M$

Figure 2-5 shows a graphical representation of definition 2.8, note that only the compact subset $Z \subset M$ is shown. The lines drawn from the points $z_1..z_4$ to positive limit point L_k^+ are the flows $\Phi_{t_n}(z)$ for $n \to \infty$. The ball of radius B_d around L_k^+ defines what points are in which finite T-class. Since the trajectories are flowing towards L_k^+ when time go to infinity, the flows reaching the ball of radius d in finite time T. According to 2.9 the four points are partitioned into two T-equivalence classes. Points (z_1, z_2) are in the same T-class and (z_3, z_4) in another.

The two definitions mention only an autonomous system, but they also hold for the non-autonomous case. The smooth vector field can represent closed loop dynamics without showing the actual inputs. This is because a stable system is needed in order to obtain the positive limit sets. The construction of the T-equivalence classes is done by defining a Lyapunov-like function for the trajectories of the system. The Lyapunov functions is conservative with respect to the actual trajectory and only proves reachability after some time i.e. the system will reach the limit set in time T.

The Lyapunov-like function is proposed, in which all the trajectories of the actual system are contained. With this information can be said that the system will always reach a specified point in the time calculated with the help of the Lyapunov function. This as a downside bring in conservatism to the system, however the avoidance of directly calculating the trajectories is preferable for most dynamical systems where no analytic solution exist and numerical integration is computationally expensive.

In order to complete the abstraction of the original hybrid system, a few assumptions by [12] need to be made in order to transform the finite time abstraction of definition 2.9 into a hybrid automaton (see definition 2.1).

- X is subset of a Banach manifold M, modeled on $\mathbb{R}^n$;
- $G(e) \neq \emptyset, \forall e \in E$;

- $R(e,x) \neq \emptyset, \forall x \in G(e)$;
- For each $q \in Q$, the positive limit set L^+ of the flows $f(q,x)$ satisfies $L^+(q) \subseteq G(e)$, for $e \in \{(q,p)|(q,p) \in E\}$;

These assumptions ensure that the resulting hybrid automaton is deterministic. This is necessary because if the hybrid system can 'choose' between continuous evolution or discrete transition, then no useful timing information can be subtracted because the trajectories are not unique.

2-4-1 Obtaining event-times

In order to construct the T-equivalence classes, the authors of [16] make a proposition. Before stating this proposition, a general procedure to find Lyapunov functions is introduced because the proposition makes use of this procedure.

A first general procedure for finding a Lyapunov function was introduced by [17]

Theorem 2.10. *([16] from [17]) The set Ω is the region of attraction of a periodic orbit $x = \varphi(t)$ with period T, if and only if there exist two functions $V(x)$ and $W(x)$ defined on Ω satisfying:*

1. *$V(x)$ is continuous on Ω and the domain of $W(x)$ can be extended to the entire state space X,*
2. *$V(x) \in (0,1) \forall x \in \Omega \setminus \varphi$, and $V(x) = 0$ for $dist(x,\varphi) = 0$,*
3. *$W(x) > 0$ for $dist(x,\varphi) > 0$ and $W(x) = 0$ for $dist(x,\varphi) = 0$,*
4. $$\nabla V^T f(x) = -W(x)\sqrt{1 + ||f||^2}(1 - V), \qquad \textit{with } f \textit{ the dynamics of the hybrid system} \tag{2-28}$$
5. *$lim_{x \to \partial\Omega} V(x) = 1$.*

From this theorem, a Lyapunov function can be calculated and that brings to the following proposition

Proposition 2.11. *([16]) Let $L_k^+(q)$ be an asymptotically stable limit set with region of attraction $\Omega_k(q)$ and $V_k(q,x)$ a solution to equation 2-28 that satisfies the requirements of Theorem 2.10. Then the (T,d)-equivalence class for $\dot{x} = f(q,x)$ is a subset of the level set $V_k(q,x) = 1-(1-c)e^{-dT}$, where c is the minimum of $V_k(q,x)$ on the boundary of $L_k^+(q)+\mathcal{B}_d$. For the proof is referred to [16]*

The time bound obtained with this method is conservative because the T-equivalence class is a subset of the Lyapunov function. This conservativity can be reduced by introducing a scaling parameter $\lambda > 0$. The function $W(x)$ is now replaced by $\lambda V_k(x)$ such that $\lambda V_k(x) \leq W(x)$.

2-5 Summary

In this chapter, we gave summary of what already exists in the literature about hybrid systems. As a general modeling class, automata and two specific classes, the PWA and MLD systems. The former is the continuous time equivalence of a SMPL system whereas the latter is used to serve as an example how to introduce logical binary variables in the SMPL system. Thereafter, max-plus algebra is introduced because we are going to model the abstracted hybrid system into this algebra as a SMPL model. Finally, a strategy for the abstraction of stable hybrid systems is presented that preserve the reachability properties of the original hybrid system.

Chapter 3

Abstraction used in a interactive sequential model framework

Scheduling and control of dynamical systems are an important part of the operation of complex processes like in the process industry or traffic management. The current approach is that the scheduling level passes information to the control level. However for an optimal performance it would be desirable to update the schedule regularly to avoid breakdown and optimize the processes. Current models that integrate both levels are large, nonlinear optimization problems and result in difficulties to implement this in real time application. This chapter focuses on the approach of [1] which proposes a distributed sequential interactive control framework. For this thesis, a simplified version of this model is set up such that we can study the how the abstraction model behaves in this framework.

3-1 General model

However, the main interest in this thesis is in the abstraction, this is part of a larger structure where it fits into a sequential framework that controls the system. A finite number of resources where one or multiple jobs can be processed by the system. The abstraction makes it so the dynamics are omitted and only the timing information is preserved. This timing information is fed into a scheduler. The scheduler then decides if and what control actions are needed in order to meet the objective. The resulting schedule goes to a path planner where these paths serve as a reference for the low level controller that controls the actual system.

The upper part of Figure 3-1 is driven by discrete-events where the lower part is driven by time. Connection of the lower and upper part will result in timing issues for the optimal control problem solved by the scheduler. These issues are dealt with in Chapter 5 by including certain states in the constraints and by choosing which cycle as the current cycle carefully.

Figure 3-1 reads as follows. The dynamic scheduler provides a feasible schedule for the jobs to complete their tasks. Vector $x(k, t_c)$ is the state of the Discrete-event system (DES) and

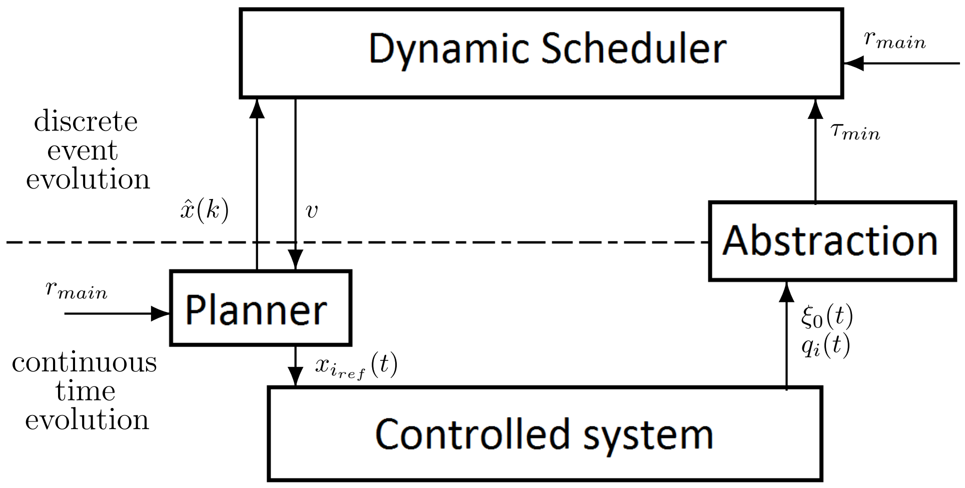


Figure 3-1: Sequential interactive model

contains information about the begin and end times of a task, furthermore synchronization between jobs is handled by the scheduler as is the order and route of the different jobs. Where $t_c \in \mathbb{R}$ is the point in time at which the continuous system is being abstracted and an optimal timetable is produced. This timetable is valid for all $t \geq t_c$. Index $k \in \mathbb{N}$ is the cycle for which the planner makes a schedule.

In the trajectory planner, a cost function decides how the optimal output trajectory of the system should evolve over time. For chemical plants this can be the efficiency of the raw product that is used for the end product, or minimal energy consumption during the process. Information stream $x_{i_{ref}}$(t) contains the optimal trajectories for all jobs in the controlled system and of course, the synchronization, routing and ordering decisions are now fixed into these reference trajectories and thus indirectly passed from the scheduling level to the control level.

It is assumed in this schematic model that the hybrid dynamical system is controlled by a lower level controller that is not specified. The controller is assumed to be able to comply with the reference trajectories provided by the dynamic trajectory planner. The continuous time state space is given by differential equations of the form $\dot{\xi}(t) = f(q(t), \xi(t), v(t))$, with q_i the mode of the hybrid system, ξ the state and v a control input. Where q_i is constant $\forall\ t \in I_i$ i.e. the hybrid mode remains constant until the system jumps to another mode. From this block, the state $\xi_0(t_c)$ is extracted at regular time intervals. These states are taken as initial conditions for the abstraction level.

The hybrid dynamical system will now be abstracted into a hybrid time set every time it sends its current state to the abstraction level. The abstraction level provides multiple hybrid time sets $\tau_{min,j}(t_c)$, for $j = 1...N_j$ where N_j is the number of jobs, to the scheduling level where each $\tau_{min,j}(t_c)$ represents a minimal process time for a job to complete on one or several resources.

The sequential model as given in Figure 3-1 connects at two positions, the discrete-event evolution with continuous time evolution.

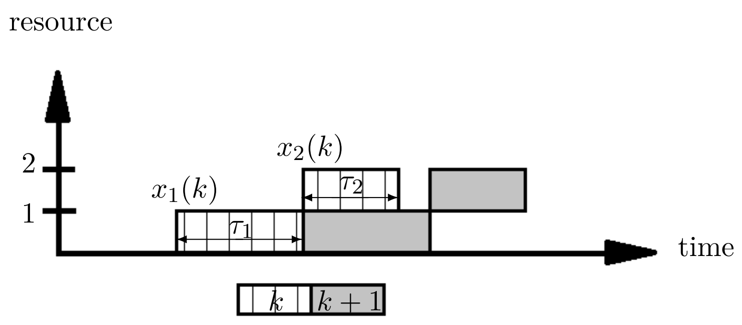


Figure 3-2: model of a printer with two resources

3-1-1 Scheduling

An important part of this whole system is the scheduling level. Normally the system is scheduled according to predefined process times and a schedule for this normal situation is implemented in the dynamical system. However the framework shown in Figure 3-2 assumes a continuous interaction with the system and provides a schedule each instance the hybrid dynamical system sends its characteristics to the abstraction level. Hence the schedule is not a fixed quantity from where information is only given downwards to the control level, but interactive. Now, not only in case of serious events happening at the plant level i.e. breakdown of the system, but also if trajectories are not able to follow the predicted path due to, for example unforeseen inputs or modeling errors, the schedule is rescheduled accordingly. This thesis aims at scheduling using max-plus algebra. The scheduling is therefore not performed in the continuous or discrete time domain, but the discrete-event domain. To enter this domain, abstraction of the continuous dynamics is required. The abstract model only needs to preserve timing information of the trajectories and not exactly how the system evolves over time. More specifically this model does not need to preserve timing information about all the trajectories of the system, but only of the fastest trajectory the system can achieve with the state initialized at some point ξ_{0t_j} for process $j \in N_j$ at time t.

Scheduling example with $\tau_{\mathbf{min}}$:

To illustrate this behaviour consider the following scheduling problem of a printer, this example is partially derived from [18].

$\tau_{min,1}$	$\tau_{min,2}$	$\tau_{nom,1}$	$\tau_{nom,2}$
6	4	7	5

Table 3-1: process times

This simplified version of a printer setup is modeled with two resources for a job to complete the printing task. The first resource includes the paper input, applying ink to the front, inverting the paper and bringing it into position for the second resource. This second resource applies ink to the back of the paper after which the paper leaves the printer. The dynamical model for this semi cyclic printer system can be derived from Figure 3-2 as follows:

$$\begin{aligned} x_1(k+1) &= \max(x_1(k)+\tau_1, x_2(k+1)-\tau_1) \\ x_2(k+1) &= \max(x_1(k)+\tau_2, x_1(k+1)+\tau_1) \end{aligned} \tag{3-1}$$

$$x(k+1) = (A_0 \otimes x(k+1)) \oplus (A_1 \otimes x(k)) \tag{3-2}$$

with

$$A_0 = \begin{bmatrix} \varepsilon & -\tau_1 \\ \tau_1 & \varepsilon \end{bmatrix}, \quad A_1 = \begin{bmatrix} \tau_1 & \varepsilon \\ \varepsilon & \tau_2 \end{bmatrix} \tag{3-3}$$

This implicit model can be solved for its explicit form with A_0^* since

$$x(k+1) = A_0^* \otimes A_1 \otimes x(k) \tag{3-4}$$

$$\text{with } A = A_0^* \otimes A_1 = \begin{bmatrix} \tau_1 & \tau_2 - \tau_1 \\ 2\tau_1 & \tau_2 \end{bmatrix}$$

The system described by (3-5) is for the autonomous case, for the nonautonomous case this is:

$$x(k+1) = A \otimes x(k) \oplus u(k) \tag{3-5}$$

where $u(k) = T^k \otimes u(0)$ and T the cycle time.

The state x at $k = 0$ is calculated by:

$$x(0) = A_0^* \otimes x_0 \tag{3-6}$$

This example shows the importance of τ_{min} versus τ_{nom} for obtaining schedules that are as fast as possible given the dynamical system constraints and a possible input. The input for this particular system is, if it exists, the timetable according to which the events must be scheduled. This is done by considering the following cases for both τ_{nom} and τ_{min}. Both τ's are put in an autonomous and nonautonomous system, after which both systems give results for the on-time case and the nonautonomous is also delayed to show the time is takes to return to the schedule.

The initial conditions for all cases are the same we assume $x_0 = [9\ 9]^T$, $u(0) = [0\ \tau_1]^T$, a cycle time of $T = 11$ and the parameters as given in Table 3-1, the eigenvalue for all cases is determined such that the minimum cycle time is known . By letting the state evolve over k and observe when $(x(k+1) - x(k))$ does not change anymore, then steady state is reached and eigenvalue λ can be determined by $(x(k+1) - x(k))$ for k.

The autonomous behaviour of the system is given by:

The transient behaviour of the system for τ_{nom} is given below and is in steady state for $k \geq 1$ with a cycle time of 7.

$$x(1) - x(0) = \begin{bmatrix} 7 \\ 7 \end{bmatrix}, x(2) - x(1) = \begin{bmatrix} 7 \\ 7 \end{bmatrix}, x(3) - x(2) = \begin{bmatrix} 7 \\ 7 \end{bmatrix}, x(k+1) - x(k) = \begin{bmatrix} 7 \\ 7 \end{bmatrix} \text{ for } k \geq 0$$

k	0	1	2	3
$x_1(k)$	9	16	23	30
$x_2(k)$	16	23	30	37

Table 3-2: nominal τ autonomous

k	0	1	2	3
$x_1(k)$	9	15	21	27
$x_2(k)$	15	21	27	33

Table 3-3: minimal τ autonomous

Similarly, for τ_{min} the state of the system given by Table 3-3 reaches steady state at $k = 1$ but now with a cycle time of 6.

$$x(k+1) - x(k) = \begin{bmatrix} 6 \\ 6 \end{bmatrix} \text{ for } k \geq 0$$

Clearly the eigenvalue related to τ_{min} is $\lambda = 6$ and the eigenvalue related to τ_{nom} is $\lambda = 7$. In this case no input is available and the events are scheduled as fast as possible.

The nonautonomous system behaviour is given below by Table 3-6 and Table 3-7

k	0	1	2	3
$x_1(k)$	9	16	23	33
$x_2(k)$	16	23	30	40

Table 3-4: nominal τ nonautonomous

k	0	1	2	3
$x_1(k)$	9	15	22	33
$x_2(k)$	15	21	28	39

Table 3-5: minimal τ nonautonomous

For both systems the cycle time is 11 since this is coordinated by the input timetable $u(k)$. However, there is a delay in the system introduced by the initial conditions for the nonautonomous system. We view the delay propagating through the system and leaving it after some k since the system is stable because of $\lambda \leq T$.

The delay z is defined as

$$z(k) = x(k) - u(k) \tag{3-7}$$

k	0	1	2	3
$z(k)$	$\begin{bmatrix} 9 \\ 9 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 5 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Table 3-6: delay propagation τ nominal

k	0	1	2	3
$z(k)$	$\begin{bmatrix} 9 \\ 9 \end{bmatrix}$	$\begin{bmatrix} 4 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Table 3-7: delay propagation τ minimal

The settling period for the nominal event times is $k_{s(nom)} = 3$ and for the minimal events times τ_{min}, the settling period is $k_{s(min)} = 2$.

This example shows the behaviour of a printer for a case where the system is autonomous and where an input is present. The printer that can print a paper the fastest is the printer for which no schedule is given and a minimal τ is provided. However, when the printer is controlled by a schedule, the limit for handling a paper is given by the maximum of the cycle time T and eigenvalue λ. Since in this example we picked $T > \lambda$, both the cycle time of τ_{min}

and τ_{nom} are equal and when the printer is in steady state, the paper is printed equally fast for both event times. When the system is delayed, the desired timetable cannot be followed by the printer, but the settling period for τ_{min} is smaller than the settling period of τ_{nom} and so the system reaches steady state in a shorter time and is therefor faster.

This example shows why we picked τ_{min} instead of τ_{nom} or even τ_{max} as the variable that has to be send to the scheduler if, as is generally the case, the objective of the scheduler is to maximize the output of the system and in doing so minimize the total production time.

3-1-2 Abstraction

It is important to note that the abstraction block takes τ_{min} as an input. The variable τ_{min} is not just a process time, but the minimum time required by the system to reach a specified goal. This will ensure that the scheduler can come up with the fastest possible timetable. Fastest in the sense that if there is no input $u(k, t_c)$ provided, the jobs will be scheduled as soon as they can possibly be processed by the system. If an input $u(k, t_c)$ is provided in the scheduler and this input happens to be the nominal schedule the system has to follow, then this input limits the throughput of the system. However, if the τ's provided by the abstraction are not the minimal τ_{min} but the nominal process times τ_{nom}, then the scheduler can never provide a schedule with a faster throughput than the τ_{nom} process times indicate. When there is no input $u(k, t_c)$ provided or even when a faster timetable for the system is provided, the limiting behaviour of the τ_{nom} is the bottleneck for faster scheduling. So, the abstraction scheme should be made valid for obtaining the minimum event times.

3-1-3 Trajectory planning

The trajectory planner provides sufficient information to the hybrid system to keep track of the general schedule provided by the scheduler. The input for this operation is the part of the state vector $x(k)$ of the scheduler which contains the begin $t_b(k)$ and $t_e(k)$ end times of all jobs that are taken into account in the scheduling operation. These are the processes that are in the predicted future and for which it holds that $t_c < t_b(k)$. Furthermore, the processes that are currently running and cannot be rescheduled anymore, i.e. they are contained in the constraints of the scheduler, for these it holds that $t_c \geq t_b(k)$.

The connection at the trajectory planning block will take the framework from the event domain to the time domain. This implies that k is omitted as driving variable and replaced with t. Since the state is now again the state of the dynamical system $\xi(t)$ this state evolves over time. Therefore the trajectory planner needs access to the state of the dynamical system at time t_c in order to acquire the correct initial state conditions.

The reference trajectory is generated using a mapping:

$$\xi_{ref_j}(t) = \begin{cases} \phi_j(\xi_{0t_j}(t), t_{e_j}(k)) & \text{if } t \in [t_{b_j}(k), t_{e_j}(k)) \\ \phi_j(\xi_{0_j}, t_{b_j}(k), t_{e_j}(k)) & \text{if } t < t_{b_j}(k) \end{cases} \tag{3-8}$$

Where $\xi_{0_j} \in \mathbb{R}^n$ is time invariant for a particular process which holds $t < t_{b_j}(k)$ i.e. process has not started yet and therefore no continuous dynamics are associated with it unless $t_{b_j}(k) \leq t \leq t_{e_j}(k)$.

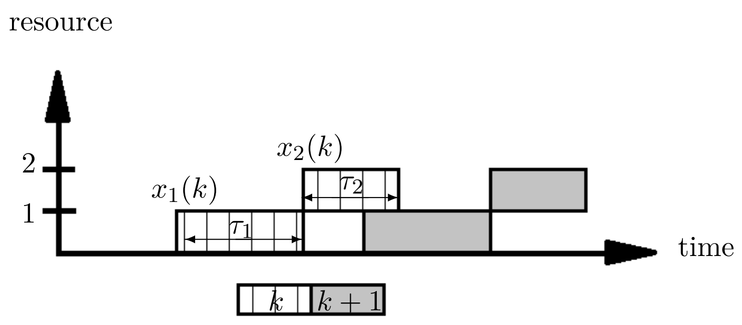


Figure 3-3: model of a printer with internal synchronization

From the section timing issues the convention is set up that the current cycle k for $x(k)$ is chosen so that all events are completely known in cycle $k-1$. This means that the initial conditions put forward to the abstraction level are all at least in cycle k and possibly in future cycles. To complete this reasoning, if the scheduler provides cycle k until the greatest cycle for which the initial conditions are provided, then at least the schedule is provided for the events that are currently running. Extending this with more future processes where $t_c > t_{b_j}(k)$ is easy since the initial state for these processes is fixed and known.

3-1-4 Handling events

A delay on the hybrid system is not modeled in the hybrid system itself, but is handled by the scheduler. This ensures that several hybrid systems can fulfill a task together in an optimal way, because the scheduler has the overview of what has to be done when and where and what operations could possible wait in order for the whole system to benefit from it.

There are three types of scheduling that can be handled by the scheduler: synchronization, routing and ordering of jobs. However, which jobs are handled by the hybrid system and which jobs are handled by the scheduler is dependent of the level of abstraction that is wanted. If there is still a high level of detail in processes then the max-plus model is very large and may contain scheduling types that cannot be rescheduled (such as a fixed route or order or synchronization between multiple processes. Then the model becomes unnecessarily complex and in that case some processes have to be grouped together in order to obtain a smaller scheduling model with a higher abstraction level. The switching points of a hybrid dynamical system form natural process times for a job to complete a task, however these switching points do not necessarily, and in practice they will usually not, intersect with the three scheduling control operations. This will be illustrated with an example of the print job as seen above, but slightly modified:

Synchronization between the processes ensures maximization of plant productivity. However, in this case, synchronization does not depend on an event time directly as can be seen from Figure 3-3. The hybrid model of the printer is the same as the previous example, but this time it is assumed that the next printing job cannot start until the tail of the current paper reaches some point in the printer to make sure that both papers keep a certain distance such that both printing jobs do not disturb each other. This is a problem in modeling since the events cannot be directly related to each other by their respective begin and end times

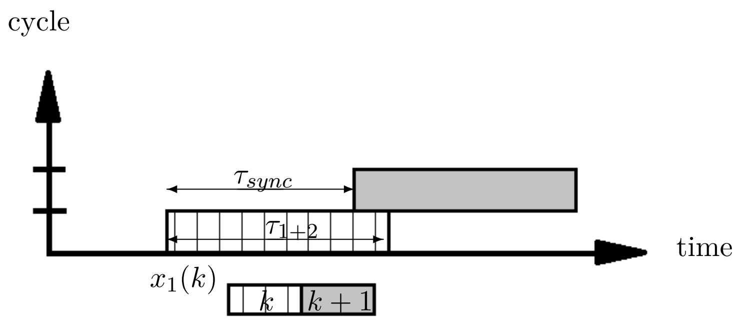


Figure 3-4: simplified printer model with general distance simulating internal synchronization

since they are internally synchronized by a different state which is not contained in the begin or end-point of the system trajectory and therefore cannot be directly accessed. Therefore, this synchronization problem is not easily handled by the scheduler directly. The trajectory planner has direct access to all the states of a system and the scheduler has not. So instead of trying to model all synchronization process times explicitly we aim at grouping together many process times that belong to one job into one process time and schedule according to a general notion of synchronization and let the trajectory planner handle the specific synchronization. The job of the scheduler is to come up with an overall best schedule, but minor corrections can take place introduced by the trajectory planner. By applying this to the printer, the model showed in Figure 3-4 is obtained where the two process times τ_1 and τ_2 are grouped together. However, the scheduler still needs a prediction of the synchronization time τ_{sync} between the processes. This will allow the scheduler to come up with a feasible schedule on a macroscopic level, where the trajectory planner fine-tunes this schedule on a more microscopic level. For a feasible timetable for the printer, τ_{sync} is introduced based on the paper dimensions and worst case scenario. This ensures that the distance between successive papers is always respected and can be relaxed by the planner. Note also that the axis that showed the resources is replaced with cycle index k.
This grouping of τ_1 and τ_2 makes sense for this model since synchronization does not depend on one of the processes τ_1, τ_2. Because the both processes are not explicitly needed by the scheduler the state of the scheduler can be reduced if the two processes are combined.

3-2 Summary

In this chapter, we introduced a generally distributed interactive sequential model that continuously (by this we mean: at regular time intervals) exchange information from plant level to scheduling level and from scheduling back to the controller of the plant level. Scheduling is performed with abstracted trajectories of the continuous time-domain hybrid dynamical system in the even-domain. Furthermore, we have shown that the process time that has to be communicated to the scheduling level must be the minimal process time.

Chapter 4

Sequential model application in railway management

The previous chapter shows an overview of a hybrid dynamical system that is sequentially controlled and scheduled for optimal performance. However, there does not exist a general procedure for verification of hybrid systems that can be implemented directly for all systems. The general model is more a general idea than a procedure that can be directly applied to all hybrid dynamical systems. Since the problem of solving the model is so difficult we choose to apply this to a railway management problem. By limiting the problem to this railway system, we can manage the complexity and be more specific about the data streams in the model.

4-1 Dynamical model for train operation

The trains are modeled as a point-mass using Newton's equations of motion. This type of modeling is accepted and frequently found in the literature [19],[20],[21]. It is assumed that the forces acting on the point-mass are tractive effort and resistance forces. This yields the following model:

$$m\frac{dv}{dt} = f_t(v) - r_{es}\left(v(t)\right) \tag{4-1}$$

Where m is the mass of the train, dv/dt is the derivative of the velocity, f_t is the control variable for force i.e. the traction or braking force and r_{es} the resistance forces. Both forces r_{es} and f_t depend on the speed making this a non-linear equation. Resistance forces are built up of roll resistance, air resistance and resistance forces due to the topology of the track [22]:

$$r_{es}(v) = m(a_0 + a_1 v + a_2 v^2 + f_{curve} + f_{line}) \tag{4-2}$$

coefficients a_0 and a_1 represent the mechanical resistance where a_3 is the coefficient for the air resistance force. In this equation, the track resistances f_{curve} and f_{line} are considered constant.

$$f_{t,max}(v) = \begin{cases} c_0 + c_1 v & \text{if } v \leq v_{limit} \\ c_2/v & \text{if } v > v_{limit} \end{cases} \tag{4-3}$$

The maximum tractive effort is linear for speeds lower than v_{limit} where adhesion limits the hyperbolic curve of the tractive power [19].

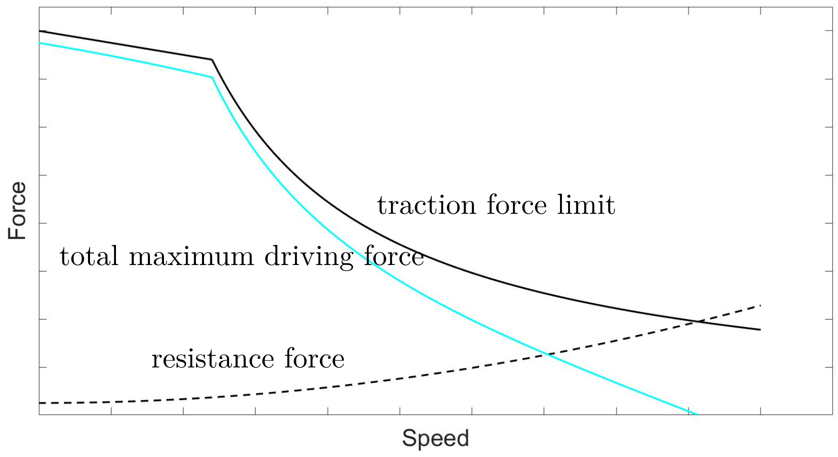


Figure 4-1: traction and resistance forces

Modeling the nonlinear hybrid system of (4-1) into a general class of hybrid systems as a hybrid automaton yields the model as in Figure 4-2 wherein this system is modeled into five modes $q \in \{1, 2, 3, 4, 5\}$ and each mode is accessible from any other mode. The model does not contain any resets of the state, however there are guards and invariants assigned to the model, but these guards and invariants come from the constraints as established by the railroad track. Therefore, the guards and invariants are not set up explicitly at this point, because they only hold for specific cases. The guards are placed at the beginning of each arrow $G(q_i, q_j)$ with $i \neq j$ and $i, j \in \{1, 2, 3, 4, 5\}$

4-1-1 Optimal trajectory

A trajectory for trains is determined by three or four phases [19] through which the state evolves over time:

1. The first phase is acceleration, in which the tractive effort is used to achieve acceleration until a certain speed is achieved at which point the train will go on to the next phase. $(f(t) > r_{es}(v))$
2. Second phase is where the train keeps the constant speed achieved in the previous phase. Tractive effort is this phase equals the resistance forces. $(f(t) = r_{es}(v))$

3. Third phase is coasting, in which the train does not input any power into the system and the train drives only against the resistance forces. ($f(t) = 0$)

4. The last phase involves braking after coasting or cruising. A train must break in order to safely reach the station. This braking point is determined by the braking coefficient and the speed of the train. ($f(t) < 0$)

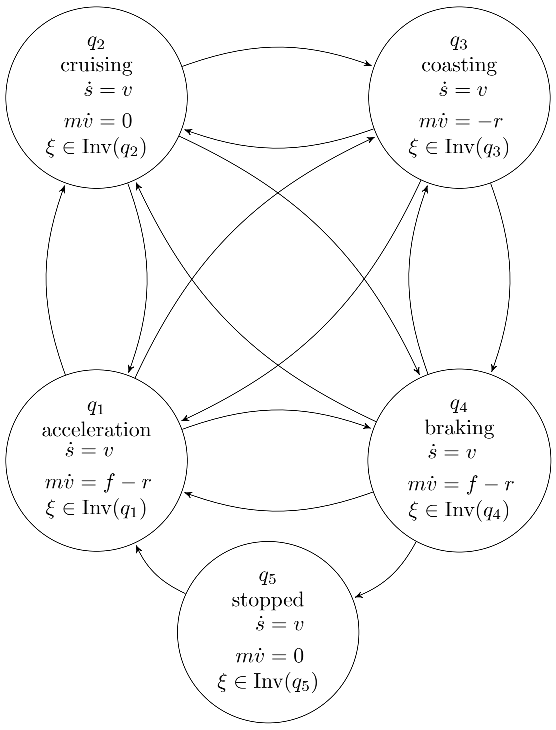


Figure 4-2: hybrid automaton train dynamics

For a minimal running time, the trajectory consists of at least the first mode and the fourth and fifth mode, the second mode is applied if the train does not drive far enough to reach the next station and since the third mode is used for energy optimal driving, this mode is completely left out when considering only the shortest time path. To achieve the shortest possible running time, the train has to use maximal traction in order to accelerate as fast as possible until the speed limit is reached or the breaking point for the next stop. If the train reaches the speed limit before the breaking point, then this train cruises at maximal speed until the breaking point is reached and it has to brake with maximal force in order to stop on time at the next station.

For energy optimal driving the train trajectory is not as obvious as that for time optimal driving. It is unclear whether a train has to accelerate fast or slow, drive at maximum speed or below speed limit, include coasting or not and apply moderate or maximal braking for arriving at a station. The energy optimal trajectory minimizes the energy over a given time T and takes into account the track constraints.

The optimal driving regimes can be simply obtained by using the Pontryagins maximum principle [19]. For braking, we assume just as [19] that this does not cost or regenerate energy. Therefore, if the traction $f(t)$ is below zero i.e. braking, the force is set to $f(t) = 0$. The objective is to minimize the mechanical energy that drives the train:

$$J = \int_0^T f(t)v(t) \tag{4-4}$$

s.t.

$$f_{min}(v) \leq f(t) \leq f_{max}(v) \tag{4-5}$$

$$0 \leq v \leq v_{max} \tag{4-6}$$

$$\begin{aligned} s(0) &= 0, & v(0) &= 0 \\ s(T) &= s_{end}, & v(T) &= 0 \end{aligned}$$

$$\dot{\xi} = \begin{bmatrix} \dot{s} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ f(t) - r_{es}(v) \end{bmatrix} \tag{4-7}$$

The Hamiltonian will read as:

$$H(s, v, f, p) = -fv + p\dot{\xi} \tag{4-8}$$

Where p is a Lagrange multiplier that is introduced by the constraints of the train. Since no regenerative braking is assumed, the Hamiltonian is split up in two regimes

$$\begin{aligned} H(s, v, f, p) &= \begin{cases} p\dot{\xi} \\ -fv + p\dot{\xi} \end{cases} \\ &= \begin{cases} p_1 v - p_2 r_{es} + p_2 f & \text{if } f(t) < 0 \\ p_1 v - p_2 r_{es} + (p_2 - v)f & \text{if } f(t) \geq 0 \end{cases} \end{aligned} \tag{4-9}$$

For all four phases the energy can be minimized by adjusting the traction force which is the control variable, and maximizing the Hamiltonian function. With the help of the Lagrange multiplier p_2 the following solutions are obtained:

optimal value $f(t)$	Lagrange multiplier condition	phase
$f_{\max}(t)$	$p_2 > v$	acceleration
$[0..f_{\max}]$	$p_2 = v$	cruising
$[0..f_{\max}]$	$p_2 = 0$	cruising
0	$0 < p_2 < v$	coasting
$f_{\min}(t)$	$p_2 < 0$	braking

Table 4-1: Optimal driving regimes

For both the first and last phases, this shows that the train must accelerate and brake with the maximum convenient traction and braking force for an energy optimal performance. This behaviour can be intuitively explained by considering two trains, the first train accelerates and brakes with maximum force whereas the second train accelerates and brakes at a lower rate. In order for the two trains to achieve the same running time, the second train must accelerate to a higher maximum speed to compensate for the slower acceleration and braking rates. Therefore the first train is more energy efficient. The optimal driving regimes only mention a cruising and coasting phase, no optimal maximum speed can be derived from Table 4-1 nor any coasting point. About coasting can only be stated that the earlier the coasting phase starts the more energy is saved. Furthermore, it can be concluded that if a trajectory consists of three or four phases that include maximum acceleration and maximal braking, then this trajectory is considered to be an energy optimal trajectory for some running time T.

4-2 Signaling

Modelling train paths so that the capacity on the line is as high as possible depends on the minimum headway between two successive trains. The minimum headway is determined by the blocking times [23] of the scheduled path of the trains. A blocking time is the total amount of time in which a section of the track is exclusively allocated to a train and therefore blocked for any other trains. A block section on a track is determined to be the length that is covered between two control signals, Figure 4-3 shows how the line is divided into block lengths, each section limited with signals. The signals provide authority over that block length and indicate an approach for the whole block section. Signaling is done so that the train must have cleared the block section and is protected by a stop signal [23]. Before this red stop signal, a yellow approach signal is in place to ensure that the train approaches the red stop signal with a moderate speed. Before the yellow approach signal, the block section is fully clear with a corresponding green signal and the train can proceed that block with maximum allowable speed.

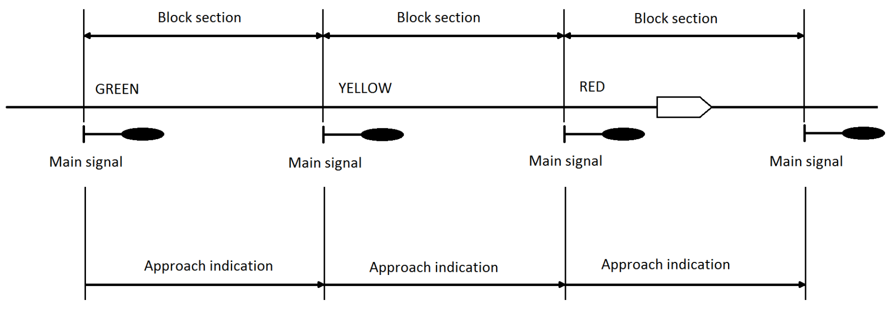


Figure 4-3: Line divided into blocks

The actual blocking times are not only the physical occupation of the train, but it also contains:

- clearing signal and reset to normal position
- signal watching time

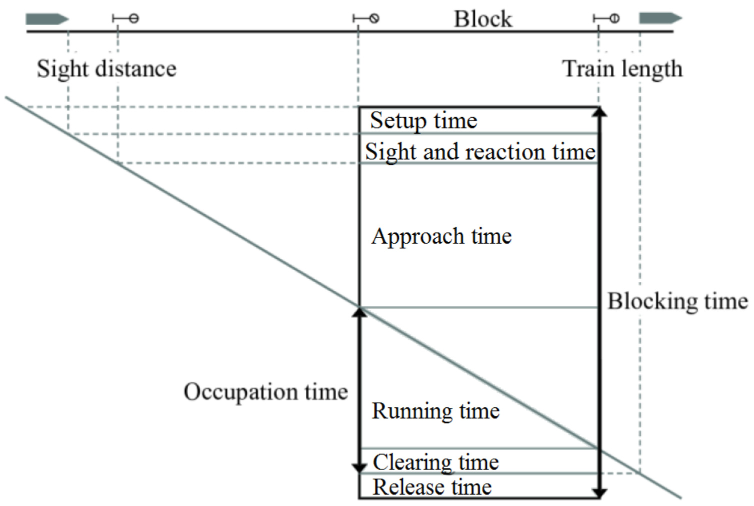


Figure 4-4: blocking time for fixed blocks (Source: Pachl (2002))

- approach time
- physical occupancy of the block section
- time to clear the block section by reaching the clearing point
- release time to unblock the block section

All components of the blocking time can be viewed in a distance time diagram with the line divided into blocks, Figure 4-4.

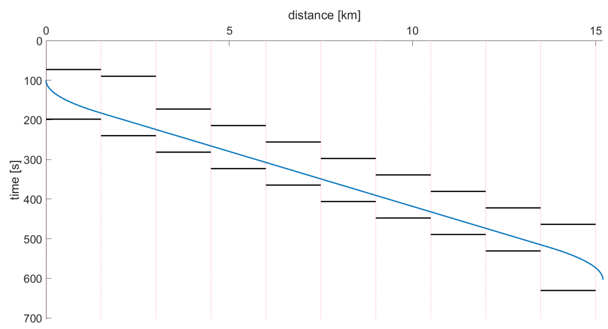
This will be illustrated with an example of two trains running over a single train line of 15 kilometer in the same direction. We assume the following characteristics for the two trains (for convenience, these train characteristics are derived from Table 6-1 which are used in the case study from Chapter 6):

train type	max speed [m/s]	max braking-distance [m]	duration max braking [s]
slow, heavy (freight)	16.7	731	88
fast, lighter (intercity)	36	1490	84

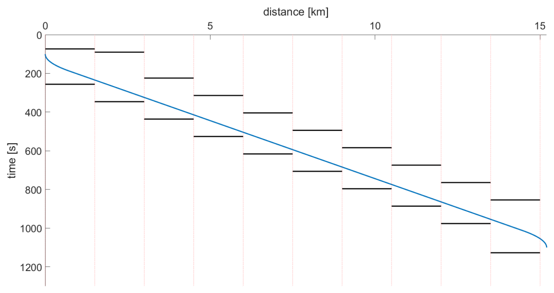
Table 4-2: Maximum braking characteristics, two train example

With the blocks optimized for the speed profile, the block length is the length of the braking distance of the train. For multiple trains on the track, the block length is chosen according to the train with the largest braking distance.

With the train characteristics as in Table 4-2, the freight train has the shortest breaking distance of 731 meter but the longest stopping time of 88 seconds. The longest braking distance comes from the intercity which is lighter than the freight train, but this is mainly because this train has a much higher maximum speed compared to the freight train. The



(a) blocking time freight train



(b) blocking time intercity

Figure 4-5: blocking times

block length for safe operation should be the highest from Table 4-2 since that represents the worst case scenario.

For simplicity in this example, the speed profiles pictured in Figure 4-5a and Figure 4-5b are the profiles for minimum running time. Later on an optimal speed profiles can be applied with coasting involved.

The minimum headway with the freight train going first is the minimum time that the intercity has to leave after the freight train in order to guarantee safe operation of the system i.e. no overlapping blocking times. The minimum headway can be calculated by letting the two trains start from the same starting time. Then define the lower and the upper limit of the blocking times, $t_{L,i(b)}$ is the begin of the blocking time of train i in block j and $t_{U,i(b)}$ is the end of the blocking time train i in block b.

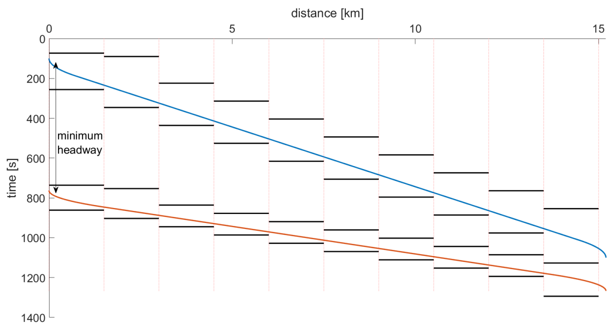
The minimum headway is defined as [23]:

$$t_{h,ij} = \max(t_{U,1(b)} - t_{L,2(b)}) \text{ for } b = 1 \ldots n_b \tag{4-10}$$

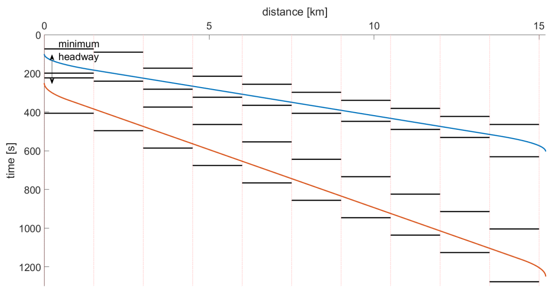
where $t_{h,ij}$ is the minimum headway for train j following train i and n_b is the number of block sections.

The minimum headway if the freight train is i and intercity j is then, $t_{h,ij}$ =663 [s] and $t_{h,ji}$ =150 [s].

The definition of the minimum headway given in (4-10) holds if only one headway is given and states when a consecutive train can leave the station behind the leading train. Later on in the case study Chapter 6 we make use of a slightly modified version of the minimum headway as used in [24]. This modification consists of introduce a separate headway for departure and arrival events. So instead of taking the minimum time between two trains for a conflict free train run, the headway times are purely extracted from the separation of the blocks at the start and end of the line. This makes the separate headways both smaller as one overall headway as in (4-10) and more suitable for rescheduling actions.



(a) minimum headway with freight train first



(b) minimum headway with intercity first

Figure 4-6: minimum headway's

4-2-1 Linearization of train dynamics

The dynamical model in (4-1) is nonlinear due to the dependence of the speed. However, to simplify the hybrid system we want to end up with a time-driven model that can be linearly described by the speed and distance. Obviously, in cruising mode the system behaves linearly, but when the speed changes, nonlinearity arises. However, in order to abstract the train dynamics and send the running times to the scheduler, linear dynamics are much more tractable and therefore desirable to handle. The goal is to end up with a linear hybrid system where:

1. All nonlinearity that is introduced due dependence of speed is omitted without compromising the trajectory simulation.
2. The time consumed for the trajectory of the linearized system should be equal to the time consumed by the nonlinear system.

The first condition advocates that the nonlinearities are taken care of by approximating them by a Piecewise affine (PWA) system just as in [21]. The second condition resembles mostly the first, but emphasizes that though there may be a mismatch between the real and simulated trajectory, the part that is most important for scheduling, timing information, is preserved. As can be seen from Figure 4-1, the total force behaves less nonlinearly than the traction force limit curve, this is because the total force is the difference between the resistance and traction force and some of the nonlinearities cancel out in this equation or are less significant. Therefore, the following PWA approximation is proposed for the maximum traction force:

$$f_{PWA,tot} = \begin{cases} s_0 + c_1 v & \text{if } v \leq v_{switch} \\ s_1 - s_2 v & \text{if } v > v_{switch} \end{cases} \tag{4-11}$$

With the new coefficient $s_0..s_2$ are chosen so that the error between $f_t(v) - r(v)$ and $f_{PWA,tot}$ is minimized.

So far, we have only looked at the Force - Speed dynamics, when actually the more interesting dynamics are the Speed - Time and Distance - Time relations since they directly show the

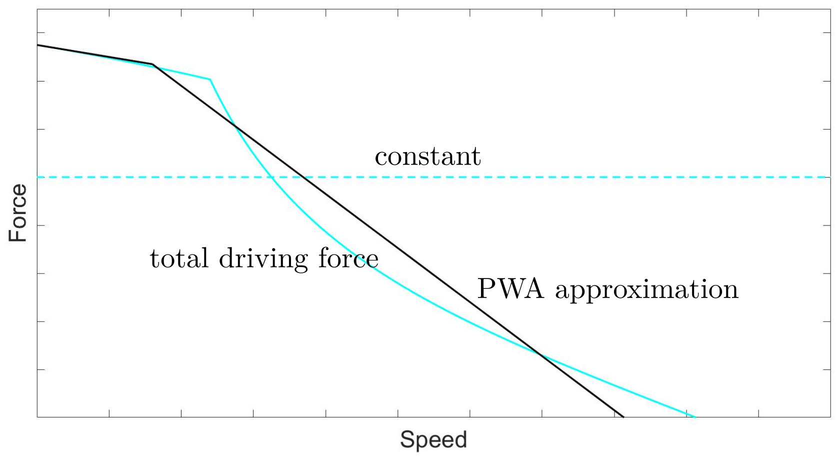


Figure 4-7: PWA approximation total driving force

nonlinear behaviour of the state of the train. Where the Force - Speed diagram shows only a function of the input and one state variable, Figure 4-8 and Figure 4-9 indirectly show the nonlinear input force for the linear approximation from Figure 4-7 as a function of speed, but explicitly the components of interest i.e. the state ξ as a function of time. From these two figures the PWA system is further simplified by grouping together linear vector fields.

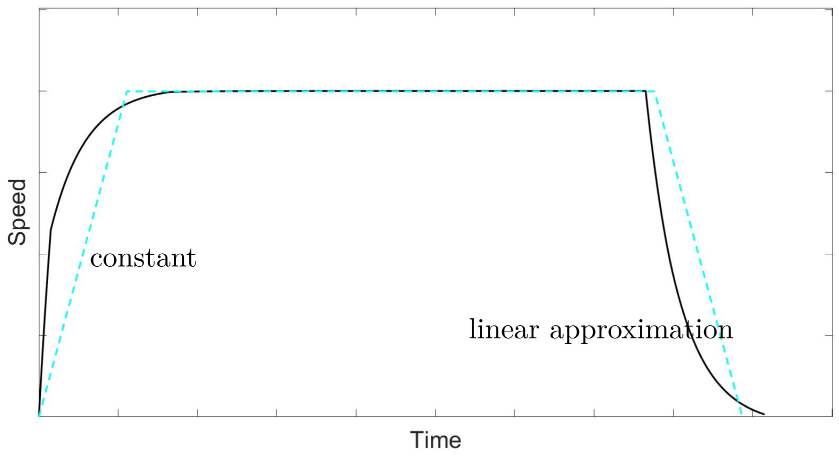


Figure 4-8: Speed - Time diagram

Both figures, Figure 4-8 and Figure 4-9 only shows a trajectory involving the three phases of acceleration, cruising and braking. The coasting phase is not shown here, but it is easy to see from Figure 4-1 that the resistance force also can be approximated with a single affine function, therefore the cruising phase consists of one mode.

Grouping together the affine flows of the multi-mode acceleration- and braking phase, (4-11), reduces each of these phases to one mode and therefore the resulting hybrid model has again five modes, for each phase one mode. More specifically, this results in a hybrid dynamic model with linear dynamics associated to each mode.

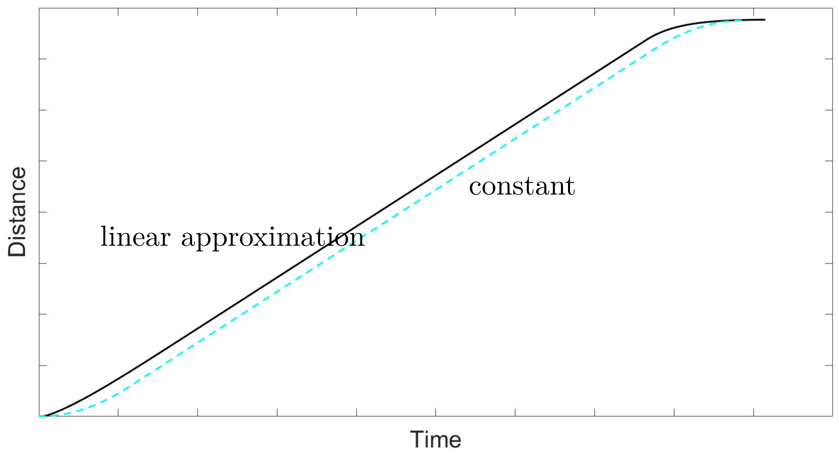


Figure 4-9: Distance - Time diagram

Furthermore, to simplify the system even more, every PWA approximation for the phases acceleration,coasting and braking are replaced with a constant value independent of speed. Figure 4-8 and Figure 4-9 also show the speed-time and distance-time profiles for a train that is approximated with constant values for acceleration and braking.

What can be seen from these figures is that the total trajectory can be approximated with the constants tuned so that this linearized, simplified model matches the nonlinear trajectory as close as possibleat the start and end of the train run. This model is therefore a good approximation for the total running time but causes an error; the trajectory is approximated when the train is already running with an initial speed $0 < v_{\text{init}} < v_{\max}$. This error causes a shorter running time than in reality and causes trains to be scheduled to early. To compensate for this and be on the safe side, the maximum error can be calculated and used as a compensation term. For this thesis we do not go into that direction since the main goal is to show that the railway network or a part of it can be optimized during operation.

With the forces modeled as constants, the following simplified free body diagram for modeling a train can be drawn:

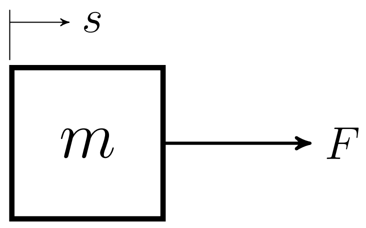


Figure 4-10: free body diagram train

The equations of motion of the free body diagram can be derived from Newton's law,

$$F = ma \tag{4-12}$$

where m is the mass of the train, a the acceleration and F the vector sum of all forces acting on the train without any dependency on speed.

For the system in figure 4-10 this can be written as

$$F = m\ddot{s} \tag{4-13}$$

The state variables

$$\begin{aligned} &\xi_1 = s(t), \quad \dot{\xi}_1 = \dot{s}(t) = \xi_2 \\ &\xi_2 = \dot{s}(t), \quad \dot{\xi}_2 = \ddot{s}(t) \\ &u = F(t) \end{aligned} \tag{4-14}$$

The state space model becomes

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u \tag{4-15}$$

where, ξ_1 is the distance traveled in m
ξ_2 is the speed of the train in m/s
u is the normalized force in N/kg
m is the mass of the train in kg

In order to implement this model for all five modes in a computer, the model must be discretized. The sampling time is T_s and is chosen as a tradeoff between calculation time and accuracy, the maximal speed of a train limits the sampling time as the error is the largest at high speeds. The method for discretization is 'zoh' which means zero order hold. This method is used because the input signal is discontinuous i.e. when the hybrid train model switch to the next mode, this is assumed to be instantaneous.

The discretized model of equation 4-15 is

$$\xi(z+1) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \xi(z) + \begin{bmatrix} 0 \\ T_s/m \end{bmatrix} u \tag{4-16}$$

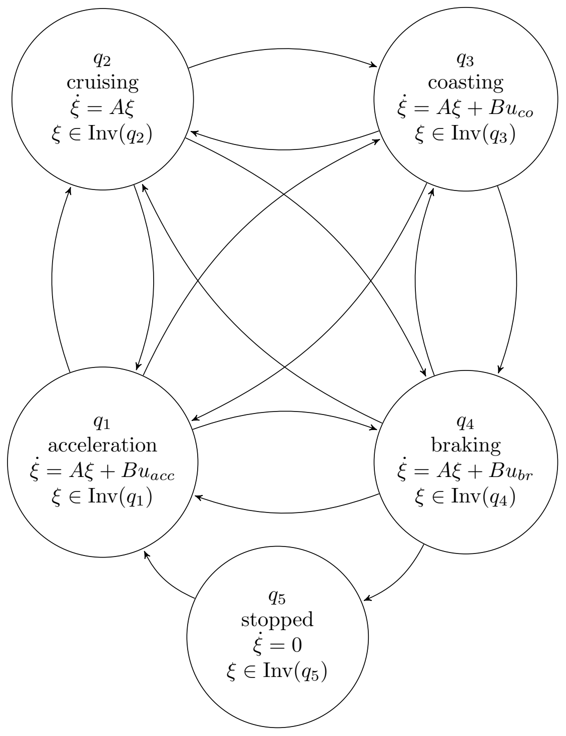
Both of the above rules are valid for the linearized hybrid model with F tuned for the three phases where a net force acts on the system, acceleration, coasting and braking.

This system can be formalized into the hybrid automaton which is shown in Figure 4-11

Where $\xi \in \mathbb{R}^2$ is the two-dimensional state of the train, consisting of the distance $s \in \mathbb{R}_+$ and speed $v \in \mathbb{R}_+$. Matrix $A \in \mathbb{R}^{2\times 2}$ contains the linear system description and matrix $B \in \mathbb{R}^2$ is the input matrix, input $u \in \mathbb{R}$ is either the net (constant) normalized input. The net forces acting on the mass are; acceleration force F_{acc} in mode q_1, coasting force F_{co} in mode q_3, braking force F_{br} in mode q_4 or zero in both modes q_2, q_5.

4-3 Abstraction using timed automata

In this section, we are setting up a procedure to abstract the continuous dynamics of each mode of the hybrid automaton into clocks of a timed automaton. The timed automaton should be able to follow the trajectories of the hybrid automaton and should therefore preserve



q_2
cruising
$\dot{\xi} = A\xi$
$\xi \in \mathrm{Inv}(q_2)$
q_3
coasting
$\dot{\xi} = A\xi + Bu_{co}$
$\xi \in \mathrm{Inv}(q_3)$
q_1
acceleration
$\dot{\xi} = A\xi + Bu_{acc}$
$\xi \in \mathrm{Inv}(q_1)$
q_4
braking
$\dot{\xi} = A\xi + Bu_{br}$
$\xi \in \mathrm{Inv}(q_4)$
q_5
stopped
$\dot{\xi} = 0$
$\xi \in \mathrm{Inv}(q_5)$

Figure 4-11: hybrid automaton, linear train dynamics

reachability. The goal is to extract timing information and collect data from multiple hybrid models, in this research this means multiple trains, and bring this timing information together in a max-plus linear model that describes the interaction between the trains in a railway network.

The idea to verify a hybrid dynamical system in a timed automaton is proposed by [16]. This idea is briefly introduced in Chapter 2, the method [16] introduced depends heavily on the assumption that the flows of each mode are bounded and go, if time goes to infinity, to one or more disjointed positive limit sets. The strenght of this concept is that they do not assume asymptotic stability of the equilibria, which is in general stricter than the existence of limit sets. However, the strength of this method is unfortunately also the main drawback. The flows must be bounded and have a finite limit point. Now, if the hybrid automaton of Figure 4-11 is considered, it is easy to see that the flows of the vector fields are not bounded and do not sink into finite limit points. Therefore, the method of [16] can not be directly applied. However, in this section we propose new conditions that hold for the hybrid automaton representing the train dynamics as in Figure 4-11.

The first condition introduced is that in every mode of the hybrid system, the state space of that mode described by differential equations, is locally Lipschitz continuous. If M is a compact metric space, then the function describing the vector field is locally Lipschitz if and only if it is Lipschitz continuous for every compact subset of M, given as; $f : M \to M$. This condition states that the functions in every mode are continuously differentiable and this will simplify the calculation of timing constraints for the timed automaton.

The guards and invariants play an important role in switching between different modes in the hybrid automaton. The guards allowing a discrete jump to another mode, whereas invariants dictate a discrete jump, so if the invariant is not satisfied anymore, the system is forced to jump to another hybrid mode. So guards can also be viewed as lower limits for a discrete jump and invariants are the upper limits.

Proposition:
- To provide unique trajectories that hold for the lower and upper time bounds, the transition relation given by the guard does only enable a jump to the next hybrid mode at the edge of the set. So, when a trajectory did not take the lower bound to jump, it must remain in the hybrid mode until the invariant condition states that a transition must take place. This way, the trajectory is only allowed to make a discrete transition at two distinct points in times, namely when the edge of the guard set is reached and when the invariant condition is no longer satisfied.

All flows in a mode must flow through the guards and invariants of that mode. Formally we define a distance metric dist(p,s) which is zero when the flow from a certain point p reaches the guard or invariant set.

With the sets defined as: $G_s \subset M$ contains the guards and invariant set and $Q \subset M$ where $Q \cap G_s = \emptyset \; \forall p \in Q$ we have that:

$$\begin{aligned} &\lim_{t \to t_G} \operatorname{dist}(\phi_t(p), G(q, q*)) = 0 \\ &\lim_{t \to t_I} \operatorname{dist}(\phi_t(p), \operatorname{Inv}(q)) = \mathcal{B}_r \end{aligned} \tag{4-17}$$

with $t_G \leq t_I < \infty$. If p already starts at a point where the guards are enabled, $\forall p \in G_s$ it holds that:

$$\lim_{t \to t_I} \operatorname{dist}(\phi_t(p), \operatorname{Inv}(q)) = \mathcal{B}_r \tag{4-18}$$

with $t_I < \infty$

where dist(p,s) is the euclidian metric $||p - s||$ with $p, s \in M$.

This assumption prevents that the flows in a certain mode have an equilibrium point that is somewhere in the invariant set, therefore preventing the system to jump to the next mode. However, if an equilibrium point exists in the boundary of Inv(q), a small ball of radius $\mathcal{B}_r$ [13] around this equilibrium is included. This ball of radius ensures that the flows will reach the invariant in finite time. Because, if there is an equilibrium in a mode that lies within the invariant set, then the hybrid system is not able to perform any transitions to other modes and therefore cannot reach the goal-set.

An example that shows these properties is illustrated in Figure 4-12

The flows ϕ_t will first reach the guard set in finite time and later reach the ball of radius around the invariant set. The ball of radius is needed to reach the edge of the invariant in finite time, because the stable limit point of the system is included in the invariant. If a point p is at a point that the guards are enabled, then the flows will not leave the guard set but will flow through to the invariant.

Figure 4-13 clearly shows that it is not necessary to have an equilibrium point contained in the guards. However, both (4-17) and (4-18) are valid and provide a consistent mapping for all the flows that fall into the compact subset of M.

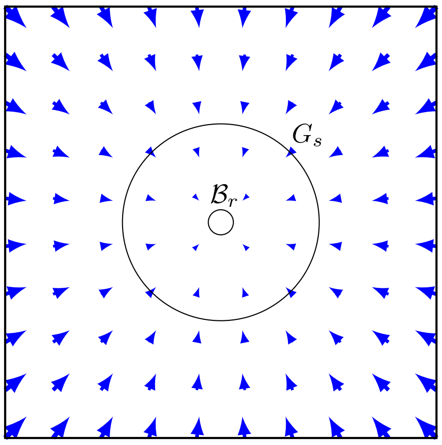


Figure 4-12: flows whithin a hybrid mode with equilibrium point

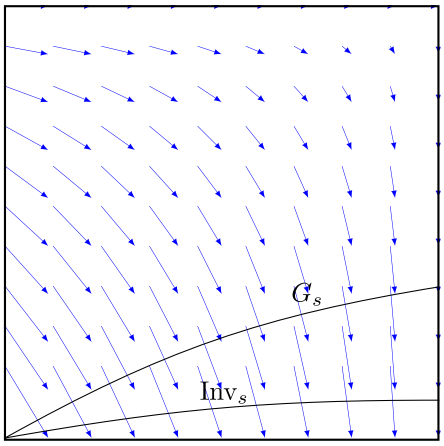


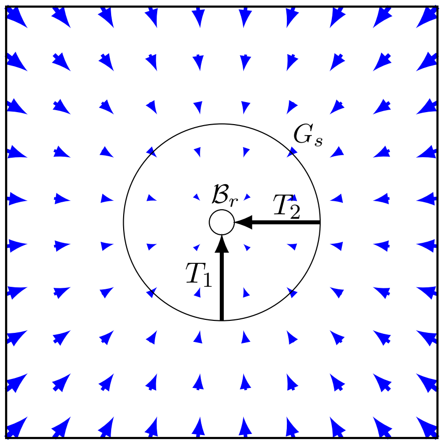
Figure 4-13: flows within a hybrid mode without equilibrium point

Furthermore, it must be ensured that the guard set G_S and the invariant set Inv_S are at the same distance in a fixed time, because we want to ensure a consistent switching between the hybrid modes in terms of a lower and upper bound on the process time. This means that it should hold

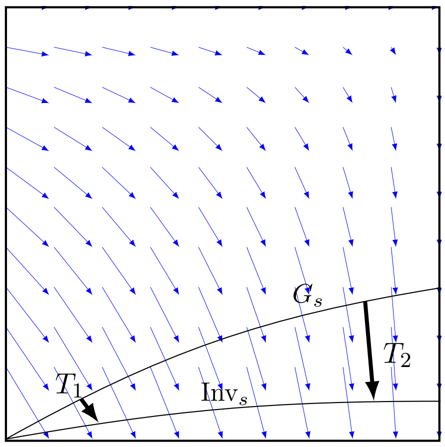
$$\mathrm{dist}(G_s, \mathrm{Inv}_s) = \inf\{\mathrm{dist}(\phi_T(p), g) | p \in G_s, g \in \mathrm{Inv}_s\} \text{ for a fixed } T. \tag{4-19}$$

This means that the two sets are at a constant 'distance' (actually the distance is a time along the trajectories of function f) T away from each other. Applying this to both figures, Figure 4-12 and Figure 4-13 yield the following:

Figure 4-14 shows for both figures two different trajectories to go from the edge of the guard set to the edge of the invariant set. It is clear that all the trajectories in Figure 4-14a flow from the guard to the edge of the invariant set in the same time. Therefore (4-19) holds and $T_1 = T_2$ for any starting point p on the boundary of set G_s. For Figure 4-14b is it clear that this will not hold. Imagine that a first trajectory starts at the intersection of the guard set with the invariant set. Clearly, the same time the guard is reached, the invariant is no



(a) time duration for the flows to go from the guard edge to the ball of radius around the equilibrium



(b) time to go from the guard to the invariant set

Figure 4-14: example regarding the positioning of the guard set w.r.t. the invariant set

longer satisfied and the system is forced to jump to the next mode i.e. $T_1 = 0$. However, if a trajectory starts from a second point that does intersect with both sets then $T_2 \neq 0$ and therefore (4-19) does not hold for this second figure.

For the hybrid automaton in Figure 4-11 holds that multiple invariant sets, $\mathrm{Inv}_n i \in I$ where $ni = \{1...Ni\}$ and Ni the number of invariant sets, can identified on the basis of track constraints. A track can be divided into multiple regions where each region holds a different speed limit. We propose that each region is considered as a separate invariant set. By dividing the total trajectory into these regions, every region has a single speed limit associated to it and can therefore described by an invariant of the speed limit, where the speed limit is continuous with no discrete transitions to another speed limit in that invariant.

If we have for example a track between two stations with three different speed limits, $v_{\max,1}$, $v_{\max,2}$ and $v_{\max,3}$ divided over the track as follows:

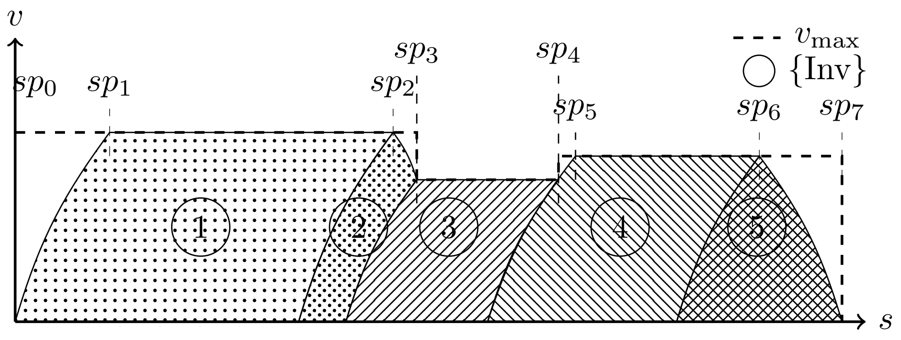


Figure 4-15: identify invariant sets for 3 different speed limits

From Figure 4-15 can be determined that there are five invariant sets where for each of these regions holds that the invariant has its own dependence on the maximum speed. The invariants are configured so that they divide the state space into time equivalence classes if the minimal running time is considered. Sets 1,3 and 4 have the constraint that $v \leq v_{\max}$ where $v_{\max}$ is a constant. Set 2 and 5 exist because the maximum speed makes a transition between two speed limits. In these sets, the constraint $v \leq v_{\max}$ still holds, but now the maximum speed is a function of the distance s.

In Figure 4-16, where the speed against the time is plotted, it can be clearly seen how a timed trajectory for the minimal time is obtained. For the minimal running time, the paths of the hybrid automaton are forced to take the fastest route in order to reach the next destination as fast as possible. Minimization of the running time leads thus to a reduced timed automaton. In contrast to the hybrid automaton in Figure 4-11 where any edge can be taken if the guard is enabled. For minimal running time, the guard set is reduced to the invariant set such that a transition is forced and therefore can be steered into the optimal direction i.e. minimizing the running time.

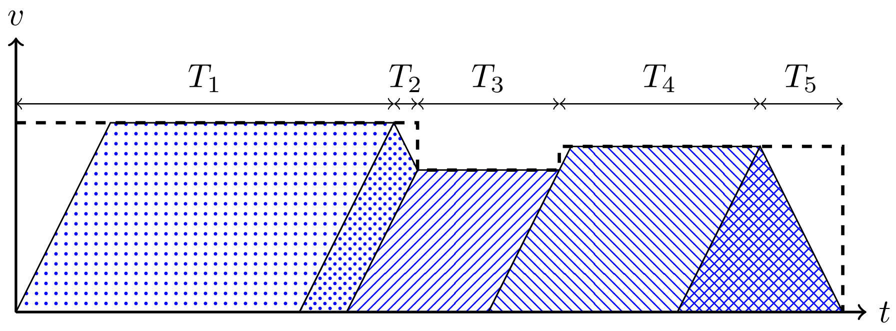


Figure 4-16: uninitialised time equivalence classes for minimal running time

Every invariant set consists of at most two hybrid states to reach the next set along the optimal trajectories of a minimal running time. Furthermore, in this example the state space is partitioned so that the invariant set $I = \{1, 2, 3, 4, 5\}$ consists of multiple positively invariant sets P where:

$$\begin{aligned} P_1 &= \{2,3,4,5\} \\ P_2 &= \{3,4,5\} \\ P_3 &= \{4,5\} \\ p_4 &= \{5\} \\ P_4 \subset P_3 \subset P_2 &\subset P_1 \subset I \end{aligned} \tag{4-20}$$

This means that if a trajectory starts in $P_2 \setminus P_3$, the trajectory has to propagate to P_3. Because the sets are positively invariant, a trajectory that starts in P_3 cannot propagate to $P_2 \setminus P_3$. This means that a train is not allowed to head back to the starting position, but is forced towards the destination station. The different sets and invariant classes are depicted in Figure 4-17

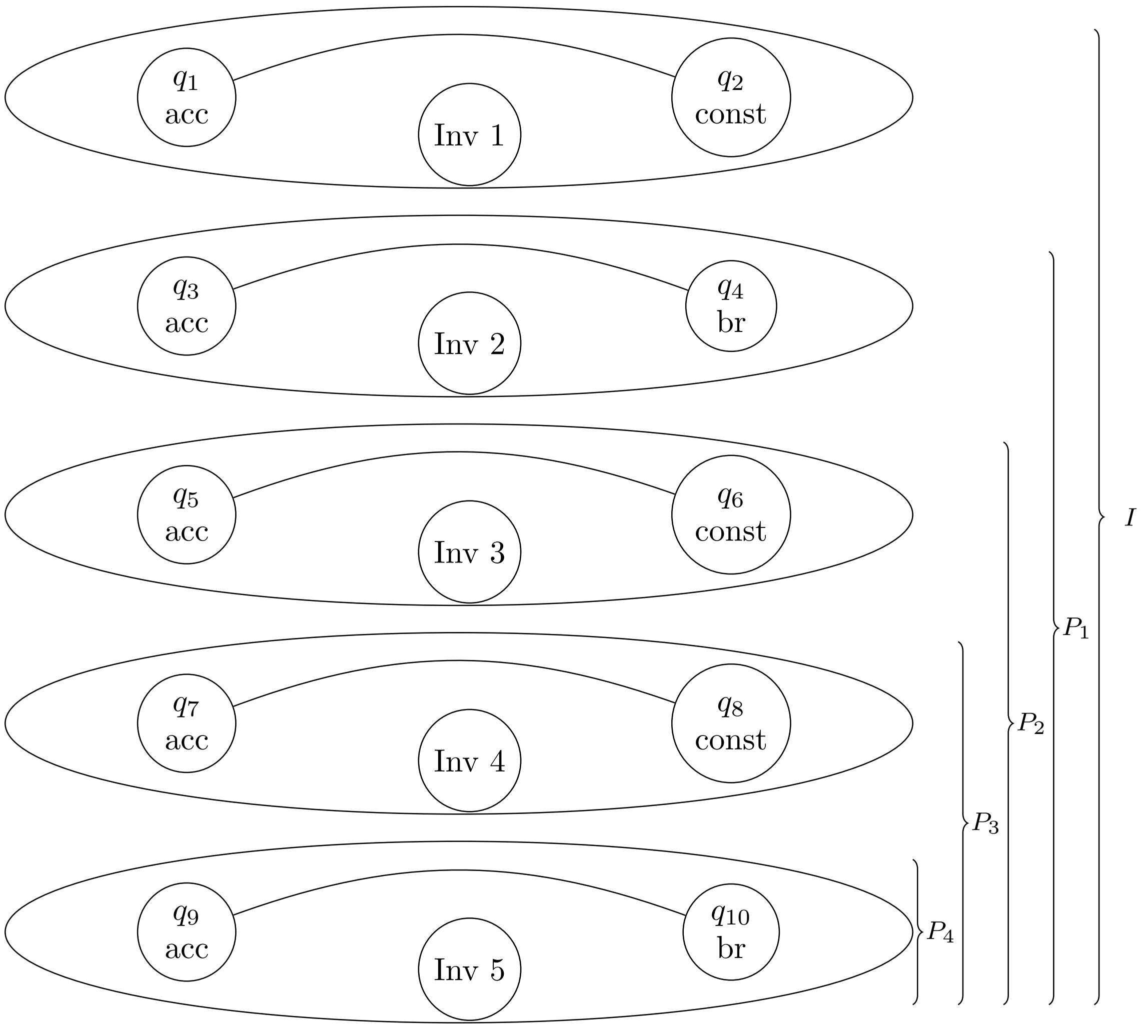


Figure 4-17: initialization for every invariant set in I

4-3-1 Time equivalence classes of the timed automaton

The procedure for finding the timed trajectory for the minimal running time first needs the calculation of the time equivalence classes for every invariant set in I. Thereafter, the timed automaton has to be initialized in order to get the real timed trajectory from the point at which the train is on the track when the state is transmitted towards the scheduler. A minimal time trajectory is built from a maximal acceleration curves, cruising at maximal speed and maximal braking curves. By integration of (4-15), for uniform acceleration, the formula for the distance is given by:

$$\begin{aligned} v &= \int \frac{u}{m}\ \mathrm{dt} = \int a\ \mathrm{dt} \\ &= v_0 + at \end{aligned} \tag{4-21}$$

and

$$\begin{aligned} s &= \iint \frac{u}{m}\ \mathrm{dtdt} = \iint a\ \mathrm{dtdt} \\ &= s_0 + v_0 t + \frac{1}{2} a t^2 \end{aligned} \tag{4-22}$$

combining (4-22) and (4-21) yield the formula that relates directly the speed and distance by:

$$v = \sqrt{2a(s - s_0) + v_0^2} \tag{4-23}$$

and rewritten for the distance:

$$s = \frac{v^2 - v_0^2}{2a} + s_0 \tag{4-24}$$

With a the acceleration in [m/s^2], s the displacement in [m], v the velocity in [m/s] and v_0, $s0$ the initial speed and initial displacement, respectively. It can be easily verified that if the acceleration is negative i.e. the train is braking, that v_0, s_0 are the final speed and final position of the curve. When the acceleration is positive, v_0 and s_0 are the initial speed and distance.
The switching points $\mathrm{sp}_{ns} \in \mathbb{R}^{\mathrm{Ns}}$ with Ns the number of switching points, and at what distance the hybrid automaton switch from one mode to another must be determined in order to simulate the behaviour of the train. The switching points for the example in Figure 4-15 for the minimal running time are given by ($\mathrm{sp}_0 ... \mathrm{sp}_7$). With the switching points and knowledge of the speed limit on the track, the switching points can be determined with (4-24).

When considering the example in Figure 4-15, the switching points (sp_0,sp_3,sp_4,sp_7) are already fixed by the mapping of the three different speed limits ($v_{\max,1} ... v_{\max,3}$). Then we have two types that determines a switching point. With the first type, a switching point is reached from a fixed switching point by acceleration from this point, with the second type a fixed switching point is reached after braking. In the proposed example, (sp_1,sp_5) belongs to the first and (sp_3,sp_6) belongs to the second type. With this information, we can conclude:

$$\mathrm{sp}_1 = \frac{v_{\max,1}^2}{2a_{ac}} + \mathrm{sp}_0 \tag{4-25}$$

$$\mathrm{sp}_5 = \frac{v_{\max,3}^2 - v_{\max,2}^2}{2a_{ac}} + \mathrm{sp}_4 \tag{4-26}$$

$$\mathrm{sp}_2 = \frac{v_{\max,1}^2 - v_{\max,2}^2}{2a_{br}} + \mathrm{sp}_3 \tag{4-27}$$

$$\mathrm{sp}_6 = \frac{v_{\max,3}^2}{2a_{br}} + \mathrm{sp}_7 \tag{4-28}$$

Now, the time equivalence classes T can be calculated for all invariant sets by solving (4-22) for t by completing the squares:

$$t = \frac{-v_0 \pm \sqrt{v_0^2 - 2a(s_0 - s)}}{a} \tag{4-29}$$

If we again consider the example from Figure 4-15 the trajectory is divided into the following time equivalence classes:

$$T_1 = t_1 + t_2 = \frac{\sqrt{-2a_{ac}(\mathrm{sp}_0 - \mathrm{sp}_1)}}{a_{ac}} + \frac{(\mathrm{sp}_2 - \mathrm{sp}_1)}{v_{\max,1}} \tag{4-30}$$

$$T_2 = t_3 = \frac{-v_{\max,2} - \sqrt{v_{\max,2}^2 - 2a_{br}(\mathrm{sp}_3 - \mathrm{sp}_2)}}{a_{br}} \tag{4-31}$$

$$T_3 = t_4 = \frac{(\mathrm{sp}_4 - \mathrm{sp}_3)}{v_{\max,2}} \tag{4-32}$$

$$T_4 = t_5 + t_6 = \frac{-v_{\max,2} + \sqrt{v_{\max,2}^2 - 2a_{ac}(\mathrm{sp}_4 - \mathrm{sp}_5)}}{a_{ac}} + \frac{(\mathrm{sp}_6 - \mathrm{sp}_5)}{v_{\max,3}} \tag{4-33}$$

$$T_5 = t_7 = \frac{-\sqrt{-2a_{br}(\mathrm{sp}_7 - \mathrm{sp}_6)}}{a_{br}} \tag{4-34}$$

by filling in ((4-26) - (4-28)) into ((4-31) - (4-34)), the T equivalence classes become:

$$T_1 = \frac{v_{\max,1}}{a_{ac}} + \frac{(\mathrm{sp}_2 - \mathrm{sp}_1)}{v_{\max,1}} \tag{4-35}$$

$$T_2 = \frac{v_{\max,2} - v_{\max,1}}{a_{br}} \tag{4-36}$$

$$T_3 = \frac{(\mathrm{sp}_4 - \mathrm{sp}_3)}{v_{\max,2}} \tag{4-37}$$

$$T_4 = \frac{v_{\max,3} - v_{\max,2}}{a_{ac}} + \frac{(\mathrm{sp}_6 - \mathrm{sp}_5)}{v_{\max,3}} \tag{4-38}$$

$$T_5 = \frac{-v_{\max,3}}{a_{br}} \tag{4-39}$$

Every set $ni \in I$ has a time equivalence class associated to it. With the information about the number of invariant sets Ni, a timed automaton can be formed that is an abstract version of the hybrid automaton of Figure 4-2. The timed automaton is as follows:

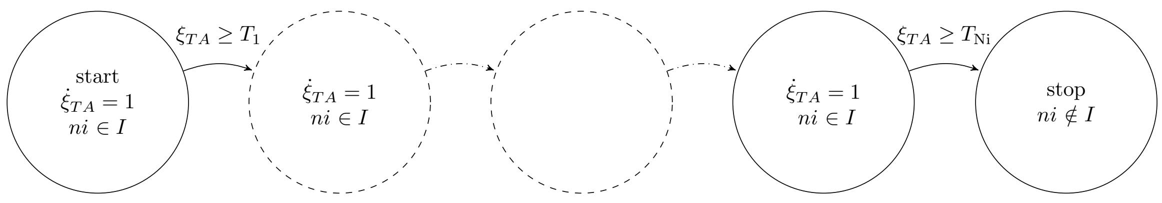


Figure 4-18: general timed automaton for a train run

where $\xi_{TA} \in \mathbb{R}_+$ is the state of the timed automaton and the evolution of the state is equal to one since it is a clock variable. The start mode lies within the invariant set I as do all the modes that are involved in the minimal running time, but the stop mode is modeled outside the invariant set I because the continuous time dynamics of the hybrid automaton are zero for every state variable. The goal is to reach the next station, because from there the train can travel further through the network. Because it is not assumed that a train stops on the track before reaching the station area, this stop mode only appears once in the hybrid state trajectory and can stay there as long as desired. Because this waiting time does not belong to a class where the state of the hybrid system evolves over time, this part does not contribute to the running time between station and is therefore excluded from invariant set I.

Note that we aim for the minimal time to traverse an invariant set. The calculation of the equivalence classes therefore, takes into account only the minimal time that is needed for the trajectories to reach the guard. In doing so, all trajectories that represent a slower path to the destination should be forgotten.

4-4 Determine switching points of the hybrid automaton

In order to determine the total minimal running time of a train as described in section 4-3, the switching points for which the train has to brake or accelerate must first be known. We consider three types of switching points following from acceleration or braking or both. First, we consider the case where a braking or acceleration curve intersects with the maximum speed. The maximal speed is known for the whole train line and the restriction of the speed includes that if a section with a lower maximum speed follows after a section with a higher maximum speed, the train has to brake so that at the entrance of the section with the lower speed limit, the train has an entrance speed of that speed limit. A speed less than the speed limit is allowed only in cases where the block sections are so short that the maximum speed cannot be reached before braking again for the next speed limit.

With the next three cases, all switching points can be obtained so that the constraints for the speed restrictions are satisfied. Since no coasting is present in the minimal running time trajectory, only three different curves can be applied. The acceleration and braking curves are derived from (4-23) and the maximum speed is assumed to be constant. The section in

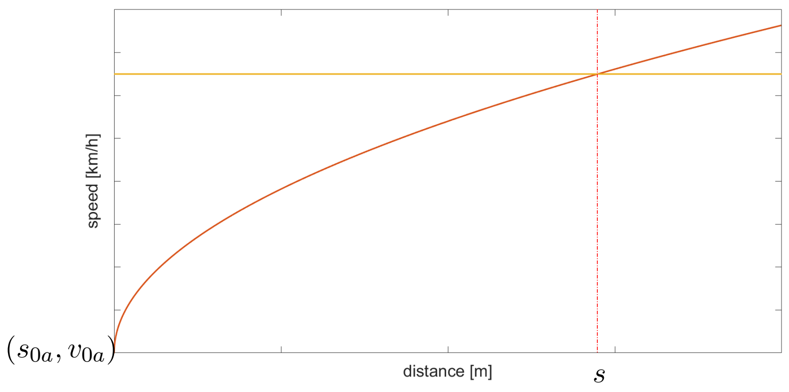


Figure 4-19: intersection, acceleration curve with maximum speed

Figure 4-19 has a maximal constant speed $v_{\max}$ associated to it. When a train is accelerating towards this maximum speed from an initial known begin speed v_{0a} and begin position s_{0a}

the point for which the train reaches this maximum speed is obtained with:

$$v_{\max} = \sqrt{2a_{acc}(s - s_{0a}) + v_{0a}^2} \longrightarrow s = \frac{v_{\max}^2 - v_{0a}^2}{2a_{acc}} + s_{0a} \tag{4-40}$$

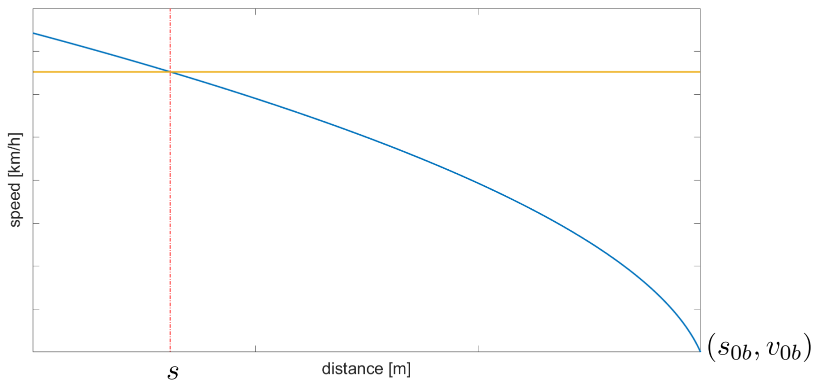


Figure 4-20: intersection, braking curve with maximum speed

The section in Figure 4-20 also has a constant maximum speed, but this time a train has to brake to reach the desired end speed v_{0b} at position s_{0b}. The switching point is determined with:

$$v_{\max} = \sqrt{2a_{br}(s - s_{0b}) + v_{0b}^2} \longrightarrow s = \frac{v_{\max}^2 - v_{0b}^2}{2a_{br}} + s_{0b} \tag{4-41}$$

Note that the initial state of the train is taken differently by braking than with the acceleration curve. Whereas it is logical for determining the switching points for every section associated to the acceleration curve in ascending order, for braking it is more convenient to start with the last section and work backwards in a descending order through the sections with different speed limits. This is because the way the braking curve is calculated is also in the reverse direction of the train movement.

The last case is where the train does not have time to reach the maximum speed since it intersects before the speed limit with the braking curve for the next section.

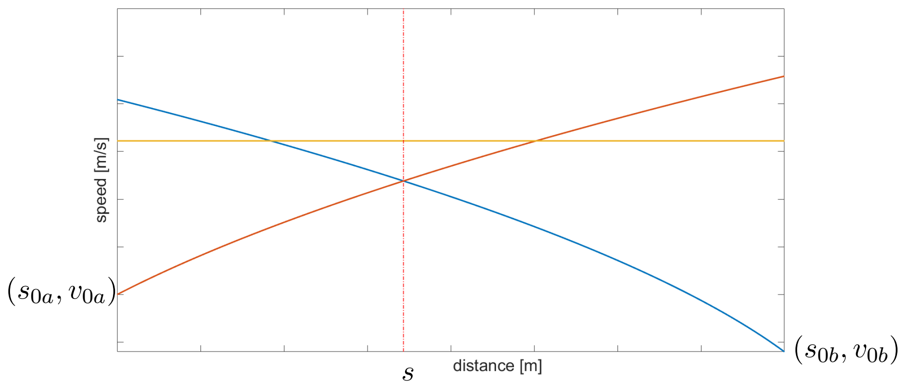


Figure 4-21: intersection acceleration and braking curve

The switching point is obtained by:

$$\sqrt{2a_{acc}(s - s_{0a}) + v_{0a}^2} = \sqrt{2a_{br}(s - s_{0b}) + v_{0b}^2} \longrightarrow s = \frac{2a_{acc}s_{0a} - 2a_{br}s_{0b} + v_{0b}^2 - v_{0a}^2}{2(a_{acc} - a_{br})} \tag{4-42}$$

With these three cases, all switching points can be obtained for an arbitrary number of different maximum speed sections over a track. These switching points can then be used for determining the minimal running time.

Chapter 5

Model Predictive Control used for scheduling

The abstraction level provides predictions for the running times of all trains in the network to the scheduling level. The scheduler schedules the trains according to a timetable that minimize delays and optimizes performance of the railway network. The predictions of the different trains are not coupled and are sent individually to the scheduler. To stimulate safe distances between trains a headway is introduced. As a result, the scheduler can schedule the macroscopic railway network only up to the precision of the prediction of this headway. In this chapter, we propose an iterative scheduling process that starts with the prediction of the running times and comes up with a feasible schedule. The next step is to implement this schedule into a trajectory planner where the resulting arrival and departure times are again fed back to the scheduler in order to check whether improvements to the control decisions made by the scheduler are possible. The iterations stop if one of the predefined stop conditions is satisfied. This concept allows inclusion of microscopic network actions to be considered by the macroscopic network scheduler.

5-1 Constraints for modeling a macroscopic railway network

The evolution of a macroscopic model for a railway network can be modeled as a discrete-event system [25]. Constraints relate the discrete-events, departure and arrival times of trains, at stations. In nominal operation, the trains in the network have a fixed synchronization and ordering between the discrete-events according to the timetable. The constraints that are necessary for nominal train operations are running time constraints, dwell time constraints, headway constraints and timetable constraints. We adopt the definitions for these constraints from [24]:

- Running time constraints relates arrival and departure of a train run in the same cycle with each other by separating these two events by a minimum running time:

$$a_i(k) \geq d_i(k) + \tau_{r,i}(k) \tag{5-1}$$

with a_i, d_i arrival and departure times for train run i in cycle k and $\tau_{r,i}$ the minimal running time for train run i in cycle k.

- At some stations, a transfer between trains is given by the schedule. The dwell time constraint therefore relates departure and arrival events of different train runs to each other by:

$$d_i(k) \geq a_c(k - \mu_{i,c}) + \tau_{dw,i,c}(k) \tag{5-2}$$

with a_c as the arrival time of the connecting train for the transfer, $\tau_{dw,i,c}$ as the dwell time for the transfer and $k - \mu_{i,c}$ indicating which cycle the connecting train is in. When the connecting train is the same as train run i and $\mu_{i,c} = 0$, then the dwell time is the time for the passengers to board and alight the train at the station.

- With the method described in subsection 4-3-1, we are able to abstract the trajectory of a single train into a running time that predicts when a train is able to arrive at the next station. Since the abstraction predicts only times between stations, the scheduler can only consider control decisions at the station. This means that in the macro network the only nodes where trains can interact with each other are at the stations in the network. The scheduler cannot handle control actions that require interaction between two trains somewhere on the track. For example, the scheduler cannot directly control the synchronization of two consecutive trains to maintain a safe distance between the two, or if a line has a junction, merging or splitting lines, the scheduler cannot directly synchronize trains at this point, because it lays on the track and not at a station. We are considering a railway network where trains can only interact at stations. So, a railway network consists of stations that are connected by times between those stations that indicate how fast the stations can be reached by a particular train. However, because the scheduler cannot directly handle synchronization between trains, indirectly the scheduler has to be able to cope with these control decisions in order to end up with a feasible schedule for the whole network. Including those constraints is thus necessary for good performance of the network. Indirectly the constraints must be taken into account by the scheduler. For the synchronization between consecutive trains a general constraint is introduced which, rather than synchronize the trains for every block section, finds a minimal time that holds for every block section and implements that at as a start constraint for a consecutive train. In this way, the headway between two trains is not directly controlled by synchronization for every block section but handled as a single constraint, the minimal headway. This constraint is written as:

$$d_i(k) \geq d_l(k - \mu_{i,l}) + \tau_{h,d,i,l}(k) \tag{5-3}$$

$$a_i(k) \geq a_l(k - \mu_{i,l}) + \tau_{h,a,i,l}(k) \tag{5-4}$$

with d_i, a_i, d_l, a_l the departure and arrival times for train run i and l, k the cycle counter and $\mu_{i,l}$ establishes the link between train run i in cycle k and train run i in cycle $k - \mu_{i,l}$ and $\tau_{h,d,i,l}$, $\tau_{h,a,i,l}$ is the minimum departure and arrival headway between train run i and l.

- A railway network has to deal with multiple objectives and passenger satisfaction is an important one. Keeping up with the timetable is of great importance to avoid delays, it is therefore tempting to leave earlier than the scheduled departure time at the station. However, this is as bad as a delay in terms of passenger satisfaction and can easily be avoided by adding the timetable constraint for departures. Furthermore, sometimes a train is not allowed to arrive earlier than scheduled, this leads to the constraints:

$$d_i(k) \geq r_{d,i}(k) \tag{5-5}$$
$$a_i(k) \geq r_{a,i}(k) \tag{5-6}$$

  with $r_{d,i}$, $r_{a,i}$ the scheduled departure and arrival times.

With the above constraints a railway network with nominal conflict-free operation for all trains can be described if we assume that the given timetable is feasible and conflict-free. The event driven dynamical model for this network has a state that consists of the arrival and departure events and the input to this network is the timetable. The constraints can be recasted as an Max-plus linear (MPL) model when grouping together (5-1) - (5-1) as:

$$d_i(k) \geq \max\Big(a_c(k-\mu_{i,c}) + \tau_{dw,i,c}(k), d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k), r_{d,i}(k)\Big) \tag{5-7}$$
$$a_i(k) \geq \max\Big(d_i(k) + \tau_{r,i}(k), a_l(k-\mu_{i,l}) + \tau_{a,i,l}(k), r_{a,i}(k)\Big) \tag{5-8}$$

We assume that the frequency of the departures and arrivals is as high as possible. This assumption leads to a model where the trains depart and arrive as soon as the constraints are satisfied. Therefore, the inequality constraints can be replaced by equality constraints and the following model:

$$d_i(k) = \max\Big(a_c(k-\mu_{i,c}) + \tau_{dw,i,c}(k), d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k), r_{d,i}(k)\Big) \tag{5-9}$$
$$a_i(k) = \max\Big(d_i(k) + \tau_{r,i}(k), a_l(k-\mu_{i,l}) + \tau_{a,i,l}(k), r_{a,i}(k)\Big) \tag{5-10}$$

Using the rules for max-plus algebra described in (2-14) and (2-15), the equality constraints of (5-9) and (5-10) can be recasted as a MPL model, just as (2-25) by:

$$x(k) = \bigoplus_{\mu=0}^{\mu_{\max}} A_\mu(k) \otimes (k-\mu) \oplus r(k) \tag{5-11}$$

with μ describing the relation between the current cycle k and another cycle μ cycles away.

we define the state vector as:

$$x(k) = \begin{bmatrix} d_1(k) \\ \vdots \\ d_n(k) \\ a_1(k) \\ \vdots \\ a_n(k) \end{bmatrix} \in \mathbb{R}_{\max}^{2n} \tag{5-12}$$

as a consequence the reference vector looks like:

$$r(k) = \begin{bmatrix} r_{d,1}(k) \\ \vdots \\ r_{d,n}(k) \\ r_{a,1}(k) \\ \vdots \\ r_{a,n}(k) \end{bmatrix} \in \mathbb{R}_{\max}^{2n} \tag{5-13}$$

The max-plus model of (5-11) is written in a compact form where $A_\mu \in \mathbb{R}_{\max}^{2n \times 2n}$.

5-1-1 Binary control variable

The model we obtained in (5-11) has a fixed model structure and can therefore not be changed and thus not be rescheduled. The model as described in (2-26) has a structure where the structure is not fixed anymore but the dynamics of the model can now be switched and give room for control. Therefore, the model of (5-11) is extended with a binary switching variable such that the fixed MPL model can be transformed into a Switching max-plus linear (SMPL) model structure.

First of all, the max-plus binary variable $s(k) \in \{0, \varepsilon\}$ is introduced. This variable can switch a constraint on and off as follows.
Consider for example the variables c,d $\in \mathbb{R}_{\max}$ and the case that:

$$d \geq c \otimes s \tag{5-14}$$

The max-plus binary variable ensures that a constraint can become active or inactive. When a constraint is active, the max-plus binary variable is equal to 0, and the constraint becomes:

$$d \geq c \text{ if } s = 0 \tag{5-15}$$

However, when the variable is equal to $\varepsilon = -\infty$ the constraint becomes,

$$d \geq \varepsilon \text{ if } s = \varepsilon \tag{5-16}$$

Which is inactive because the variable d is always larger than ε and therefore the constraint is trivially satisfied for any value of c.

This control variable $s(k)$ becomes an important tool in changing the order of constraints by introducing the adjoint max-plus binary variable $\bar{s}(k) \in \mathbb{R}_{\max}$. The adjoint is the exact opposite of $s(k)$ such that:

$$\text{if } s(k) = 0,\ \bar{s}(k) = \varepsilon \tag{5-17}$$

$$\text{if } s(k) = \varepsilon,\ \bar{s}(k) = 0 \tag{5-18}$$

This way the constraints can be re-synchronized because:

$$\begin{aligned} d &\geq c \otimes s \\ c &\geq d \otimes \bar{s} \end{aligned}$$

In this way, this constraint set can only be active for one possible ordering in synchronization.

Since $-\infty$ cannot be directly implemented, the max-plus binary control variable is adjusted with a large negative number $\beta \ll 0$ instead of ε. A normal binary variable $u(k) \in \{0, 1\}$ is therefore introduced with its adjoint variable $\bar{u}(k) \in \{0, 1\}$. The new binary variable is implemented as:

$$\begin{aligned} s &= \beta u \\ \bar{s} &= \beta(1 - u) \end{aligned}$$

with β large enough so that the constraints with the max-plus binary variable remain unchanged compared to the new binary variable.

5-2 Optimization

The constraints for the departure and arrival events in a MPL model can be transformed into the constraints of a SMPL model [10] with the help of a binary control variable. With the ability of the SMPL model to brake synchronization, the model can be controlled in order to come up with a new optimized timetable in case a delay is present. The optimization problem for MPL systems can be solved as a Model predictive control (MPC) problem [26]. With the cost function defined for a certain objective and mixed-integer linear constraints. The optimization problem is as follows:

$$\begin{aligned} &\min\ c^\top z \\ &s.t.\ Az \leq b \end{aligned} \tag{5-19}$$

with

$$z = \begin{bmatrix} d_1 \dots d_n\ a_1 \dots a_n\ u_1 \dots u_m \end{bmatrix}^\top \tag{5-20}$$

and

$$c = [\underbrace{1 \dots 1}_{n}\ \underbrace{1 \dots 1}_{n}\ \underbrace{0 \dots 0}_{m}] \tag{5-21}$$

where d,a are the continuous departure and arrival variables and u is the binary control variable, variable c is a weighting vector, n the number of trains and m determines the maximum number of control variables depending on the prediction horizon and the number of control actions i.e. connecting different trains or ordering of trains.

5-2-1 Constraints for ordering

We only consider the control action for breaking the synchronization between trains, in order to allow a reordering control actions and obtain an optimized schedule by solving a Mixed-integer linear programming (MILP) problem [27]. Reordering between two train runs i and l is done by rewriting the headway constraints of (5-3) and (5-4) as a set of mixed-integer linear constraints as follows:

$$d_i(k) \geq d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k) \tag{5-22}$$
$$d_l(k-\mu_{i,l}) \geq d_i(k) + \tau_{h,d,l,i}(k-\mu_{i,l}) \tag{5-23}$$
$$a_i(k) \geq a_l(k-\mu_{i,l}) + \tau_{h,a,i,l}(k) \tag{5-24}$$
$$a_l(k-\mu_{i,l}) \geq a_i(k) + \tau_{h,a,l,i}(k-\mu_{i,l}) \tag{5-25}$$

With a binary variable, (5-22), (5-24) and (5-23), (5-25) can be switched on or off at the same time and able to change the order of the trains. The constraint for the arrival and departure times d_i, a_i and d_l, a_l for both options, i before l and l before i are switched by a single max-plus variable s and its complement $\bar{s}$ that determines the order as follows:

$$d_i(k) \geq d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k) + s(k) \tag{5-26}$$
$$d_l(k-\mu_{i,l}) \geq d_i(k) + \tau_{h,d,l,i}(k-\mu_{i,l}) + \bar{s}(k-\mu_{i,l}) \tag{5-27}$$
$$a_i(k) \geq a_l(k-\mu_{i,l}) + \tau_{h,a,i,l}(k) + s(k) \tag{5-28}$$
$$a_l(k-\mu_{i,l}) \geq a_i(k) + \tau_{h,a,l,i}(k-\mu_{i,l}) + \bar{s}(k-\mu_{i,l}) \tag{5-29}$$

Implementing the max-pus variable with $\beta \in \mathbb{R}$ and $u \in \{0, 1\}$, this leads to:

$$d_i(k) \geq d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k) + \beta u(k) \tag{5-30}$$
$$d_l(k-\mu_{i,l}) \geq d_i(k) + \tau_{h,d,l,i}(k-\mu_{i,l}) + \beta(1-u(k-\mu_{i,l})) \tag{5-31}$$
$$a_i(k) \geq a_l(k-\mu_{i,l}) + \tau_{h,a,i,l}(k) + \beta u(k) \tag{5-32}$$
$$a_l(k-\mu_{i,l}) \geq a_i(k) + \tau_{h,a,l,i}(k-\mu_{i,l}) + \beta(1-u(k-\mu_{i,l})) \tag{5-33}$$

If $\beta \ll 0$ is chosen correctly, equation (5-34) - (5-37) is manipulated by u such that the ordering between train run i and l is changed accordingly:

$$d_i(k) \geq \begin{cases} d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k) & \text{for u=0 'active'} \\ d_l(k-\mu_{i,l}) + \tau_{h,d,i,l}(k) + \beta & \text{for u=1 'inactive'} \end{cases} \tag{5-34}$$

$$d_l(k-\mu_{i,l}) \geq \begin{cases} d_i(k) + \tau_{h,d,l,i}(k-\mu_{i,l}) + \beta & \text{for u=0 'inactive'} \\ d_i(k) + \tau_{h,d,l,i}(k-\mu_{i,l}) & \text{for u=1 'active'} \end{cases} \tag{5-35}$$

$$a_i(k) \geq \begin{cases} a_l(k-\mu_{i,l}) + \tau_{h,a,i,l}(k) & \text{for u=0 'active'} \\ a_l(k-\mu_{i,l}) + \tau_{h,a,i,l}(k) + \beta & \text{for u=1 'inactive'} \end{cases} \tag{5-36}$$

$$a_l(k-\mu_{i,l}) \geq \begin{cases} a_i(k) + \tau_{h,a,l,i}(k-\mu_{i,l}) + \beta & \text{for u=0 'inactive'} \\ a_i(k) + \tau_{h,a,l,i}(k-\mu_{i,l}) & \text{for u=1 'active'} \end{cases} \tag{5-37}$$

5-3 Iteration between the microscopic and macroscopic railway network

The infrastructure of the network that is implemented in the scheduler in section 5-2 is a macroscopic version of the network. Since the MILP formulation in (5-19) is an optimization problem that belongs to the class of NP-hard problems [28] it is therefore necessary to model the network with less detail, because the computational complexity grows exponentially with the number of variables. The macroscopic network includes stations and junctions modeled by nodes and train lines modeled by single links. Microscopic models are more detailed and contain more details about the infrastructure of the network. This detailed model typically includes speed limits, block sections, signals and all tracks. Because the microscopic model is detailed and takes into account the most important characteristics of the railway network, this model is used for conflict detection of the general schedule given by the macroscopic scheduler. This results in a conflict-free schedule. However, this conflict-free schedule does not necessarily represent the most optimal schedule possible and therefore the main properties of this schedule are fed back to the macroscopic scheduler to check for improvements in the control variables. Stopping conditions are necessary to apply to the iterations because otherwise a solution might not be found. The stopping conditions should at least contain a maximum computations time to prevent that the iterations form a bottleneck for the whole control system. Detailed stopping conditions for the iterations in the case study are defined in Section 6-6-1.

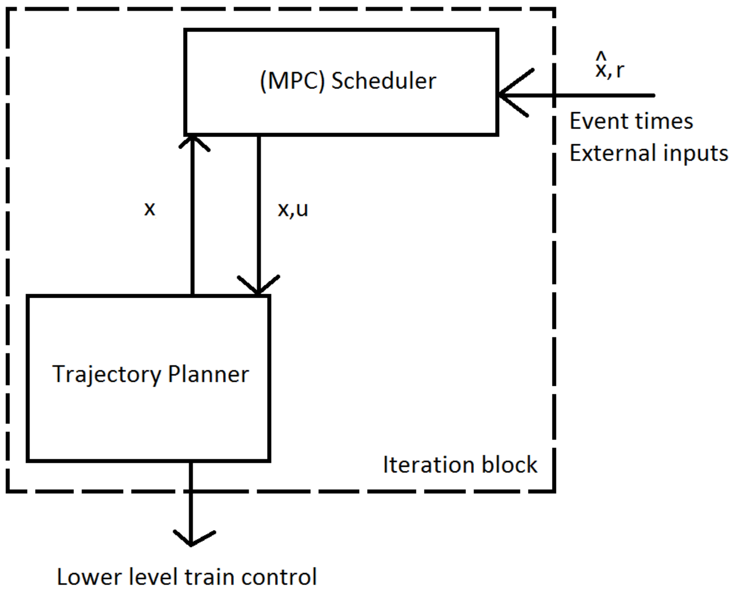


Figure 5-1: Iterations between the macroscopic and microscopic network

Figure 5-1 shows the iterative behaviour between the scheduler which contains the macroscopic characteristics of the network and the trajectory planner which takes into account the details at the microscopic network level. The input for the scheduler is the predicted arrival and departure times at the stations given by $\hat{x}$ at time $t(k)$ and the main timetable given as a reference r. The scheduler outputs a rough schedule to the trajectory planner and the trajectory planner refines this schedule. In case of a conflict, the departure and arrival times of the relevant trains are shifted into a feasible direction. This results in a new schedule, but note that no actual scheduling control decisions are being made by the trajectory planner. The planner is only allowed to turn the rough schedule into a feasible schedule with the fixed control variables. So, no re-ordering, rerouting and re-synchronization can be done by the planner, however their only task is to fit a feasible trajectory for all the trains according to the microscopic constraints.

5-3-1 Safety control

The trajectory planner deals with the interaction between multiple trains and track constraints. The latter is mostly controlled by the signaling systems at the track and speed limits for the sections of the track. After the optimal schedule is calculated, it is transmitted to a lower level controller that controls the movement of the trains. This controller should, in normal operations, only keep track of the trajectories put forward by the planner as a reference. However, in real-time operations there are extra constraints for this controller in order to guarantee safe operation. The safety of a railway network is mainly because all trains keep a safe distance from each other so that a train can always brake if a preceding train makes an unscheduled emergency stop. The safety distances between the trains are maintained by block sections where each block section is equipped with light signals to indicate a free entrance to that sections, or if a train is still on that block section, the latter train is blocked with a red signal to prevent that train from entering the block section and therefore keeping a safe distance between the trains. This practice is described in section 4-2. This means that independent of the conflict-free reference trajectories given by the trajectory planner, an event can still happen so that a block section cannot be released for an upcoming train and this train must reduce its speed or even make an unscheduled stop before the entrance of this block section. So, the line side signals for safety are an extra constraint for the lower level controller and are independent of the predicted optimal schedule.

5-4 Summary

In this chapter, we laid out how the macroscopic model can be rescheduled when translating a set of inequality constraints that form the MPL network into mixed-integer linear constraints so that a SMPL network is obtained which can be solved by a MILP method for optimization. A basic feasible schedule is obtained from the real-time abstracted railway network. Furthermore, we have introduced iterations between the macroscopic and microscopic network in order to further optimize the schedule and end up not only with an optimized schedule according to macroscopic constraints but take also into account the microscopic constraints, thereby optimizing the interaction between trains.

Chapter 6

Case study: Scheduling of a real-time abstracted railway network

In this chapter, we outline a scenario of multiple trains running over a single train line. This small network is abstracted at regular time intervals. The abstracted network contains the predicted arrival times of the trains that are already on the track. All the predicted and known process times are then modeled as a Switching max-plus linear (SMPL) framework, subsequently this framework is optimized by Model predictive control (MPC) and the SMPL framework is translated into a set of linear constraints which are solved by Mixed-integer linear programming (MILP). The control actions considered are only to change the order of the trains.

6-1 Case study: part of a railway network

In this case study we consider a small part of a large railway network. The corridor Rotterdam CS - Delft CS [22] served as an example for our case study (number of stations in this corridor and approximately the total line length). This corridor consists of three stations (Rotterdam CS,Schiedam CS,Delft CS). These stations are renamed in the case study as (A,B,C) respectively. We assume that at stations A and C, all trains must make a stop and at station B only stop trains make a scheduled stop. Furthermore, we assume that the track between the stations consists of a single line per direction. Also it is assumed that no trains can be overtaken once on the track. Trains from different locations in the network arrive at station A and eventually leave the track section at station C. Since it is assumed that at station B no train can be overtaken, this station is abstracted away and only the running times between station A and C are used in the optimization model. However, note that the stop at station B is contained in the total running time between station A and C. Furthermore, the track is divided into block sections to guarantee a safe distance between all trains. The length of the block sections is equal divided over the whole track. The track in numbers: the total track length between station A and C is 15000 meters and the block sections have a length of 1500 meters, with station B between A and C. The length of the block sections is chosen so that it is larger than the longest worst case braking distance of all trains running over over the track [19], also described in Section 4-2.

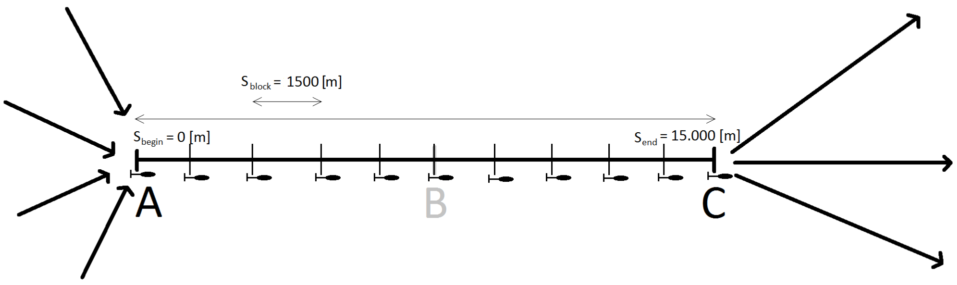


Figure 6-1: Track section considered for case study

The track section of Figure 6-1 is part of a larger railroad network and to place this in perspective the following distributed optimization network [24] is shown for an entire railway network. Note that [24] described this distributed network so that the MPC problem can be handled efficiently.

Figure 6-2 shows how an entire railway network can be optimized in a distributed way. The network is divided in sections and these sections are divided into micro sections. The track section of Figure 6-1 is such a micro section. So, this case study is just a small part of a large network.

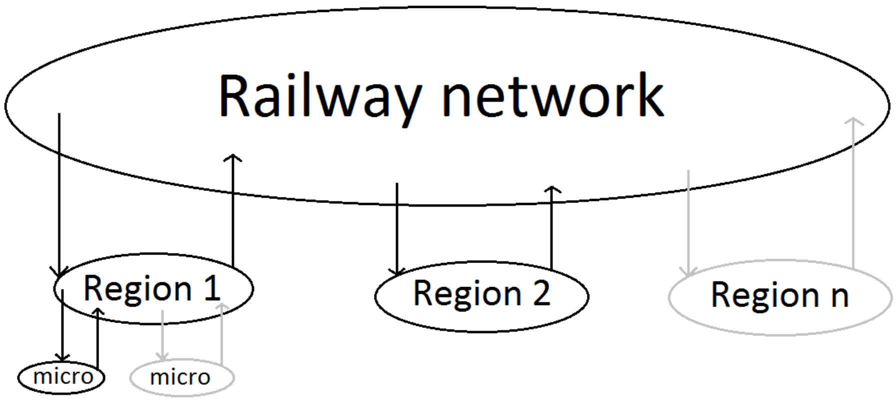


Figure 6-2: Distributed representation of an entire railway network

6-2 Rolling stock

We assume three different train types on the track with different characteristics in maximum speed, acceleration and braking behaviour. These characteristics are given in Table 6-1 and comes from the typical values for the different train types in [29]. Furthermore, one of the trains is a stop train and has an extra stop at station B. At this stop the train cannot be overtaken by other trains since we assume a single train line. In this case study, we study only the train movements in one direction so that all trains run towards the same station and not in the opposite direction. The parameters of the rolling stock are given in Table 6-1.

train type	freight train	stop train	intercity	symbol
train mass [kg]	$2 \cdot 10^6$	$1.8 \cdot 10^5$	$3.2 \cdot 10^5$	m
pull up force [N]	$3.8 \cdot 10^5$	$9.3 \cdot 10^4$	$1.4 \cdot 10^5$	F_a
friction force [N]	$-4.8 \cdot 10^3$	$-1.44 \cdot 10^4$	$-1.44 \cdot 10^4$	F_c
braking force [N]	$-3.8 \cdot 10^5$	$-9.3 \cdot 10^4$	$-1.4 \cdot 10^5$	F_b
maximum speed [m/s]	16.7	36.1	36.1	V_{max}

Table 6-1: parameters per train type

The maximum speed is equal for both the stop train and intercity while the freight train has a lower maximum speed. This is because a freight train has a much longer braking trajectory than the lighter intercity's and stop trains. Since we do not have access to realized train data for the train types we have chosen, the forces and train masses for each train type follow from an educated guess for the acceleration of each train type. We can make these assumptions since the simulation is not compared to real train data and only to show how the control decisions are being made in the case of a disturbance scenario.

6-3 Obtaining the timed automaton used for abstraction

For this case study, we want to get rid of the continuous dynamics present in the modes of the hybrid automaton of Figure 4-11 and replace them with timing information in a timed automaton.

First of all, keep in mind that we are not aiming at a full reproduction of all the trajectories that can actually be made by the automaton in Figure 4-11, but only the minimal time trajectories since these minimal process times are modeled as a Max-plus linear (MPL) model. The assumptions and limitations made on the minimum run time of a train are:

- Between stations, only a single maximum speed limit is allowed.
- If a train has started with coasting or braking, the trajectory can no longer be changed.
- The track length is sufficiently long enough that it is always possible for a train to reach the maximum speed from a starting station to the next station.

The first item is introduced to reduce the number of discrete transitions between the modes of the hybrid automaton. The second item prevents the train from accelerating or cruising again if that train already has started with coasting. There are only two paths left in acquiring a minimum running time. One path includes coasting if the train for which a minimal path has

mode of operation at time t	include in minimal running time q_1	include in minimal running time q_2	include in minimal running time q_3	include in minimal running time q_4	include in minimal running time q_5
q_5(start)	x	x		x	x
q_1	x	x		x	x
q_2		x		x	x
q_3			x	x	x
q_4				x	x
q_5(stop)					x

Table 6-2: mode sequence for minimal running time and different starting modes

been determined has already started with coasting and another path excludes coasting if the real train on the track has not yet started with coasting.

It can be seen that mode q_3, coasting, is always avoided except if the train already runs in that mode. The hybrid automaton that is only valid for the minimal running time reads as:

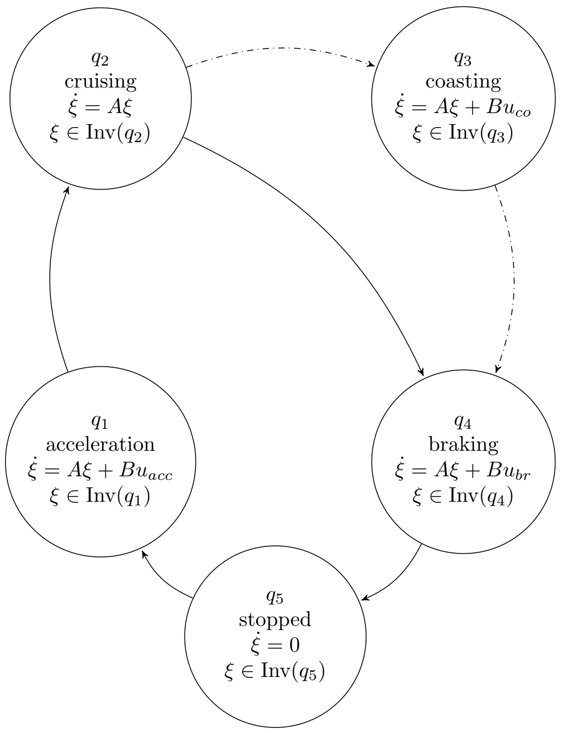

q_2
cruising
$\dot{\xi} = A\xi$
$\xi \in \mathrm{Inv}(q_2)$
q_3
coasting
$\dot{\xi} = A\xi + Bu_{co}$
$\xi \in \mathrm{Inv}(q_3)$
q_1
acceleration
$\dot{\xi} = A\xi + Bu_{acc}$
$\xi \in \mathrm{Inv}(q_1)$
q_4
braking
$\dot{\xi} = A\xi + Bu_{br}$
$\xi \in \mathrm{Inv}(q_4)$
q_5
stopped
$\dot{\xi} = 0$
$\xi \in \mathrm{Inv}(q_5)$

Figure 6-3: hybrid automaton for minimal train run

Finally, for the timed automaton, the connection between q_5 and q_1 is also broken because we make use of an initialized timed automaton with final state q_5. This allows for unique trajectories where the timed automaton provides the minimal running time for a particular initialized trajectory.

We assume that the guard set coincides with the invariant set and therefore we trivially satisfy the condition in (4-19). Furthermore, the system has no equilibrium points that prevent switching (and therefore preventing reaching the destination) until the stop mode is reached. The speed of the train is by definition zero in q_5 and the train can therefore stay there in theory until infinity, ruining the actual running time between stations. To prevent this from happening, the time in q_5 of the timed automaton is set to zero by definition.

Furthermore, since we have fixed the maximum speed of the train, and have not allowed cruising at a speed lower than the speed limit, we have a fixed $t_G = t_I$. The guard does not depend on the speed and the distance anymore, but purely on the distance. Therefore, we are able to construct rectangular constraints for the timed automaton, which makes it decidable.

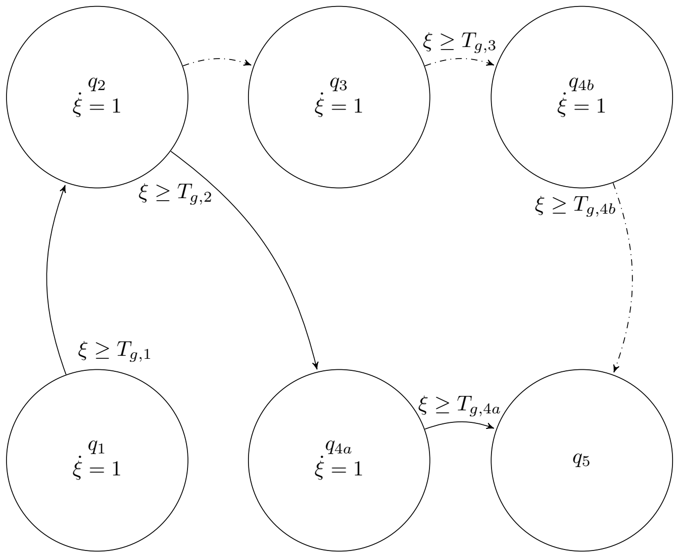


Figure 6-4: timed automaton

An extra mode is introduced, because the hybrid automaton of Figure 6-3 has two incoming edges, from q_2 and from q_3, therefore this mode is split into two modes q_{4a} and q_{4b} in order to obtain a single guard constraint and therefore a unique trajectory per mode. The trajectory is unique, since modes q_3 and q_4b can only be accessed if a train on the track is in coasting mode during optimization of the timetable.

6-3-1 Modeling a railway network

This case study uses the method described in [24], for modeling a railway network as a SMPL model and how MPC can be applied to obtain to this model at each optimization instant. The set-up of this case study is that there will be trains running over a single track, where the track length is 15000 meter. The timetable repeats itself after 30 minutes and the ordering is assumed as follows; the first train is an intercity, the second a stop train, then an intercity again, a freight train and finally an intercity again in the nominal schedule as given in the table:

The timetable is periodic with 30 minutes. With the autonomous behaviour it will be shown that this nominal timetable is stable.

Train	Departure times [min]	Arrival times [min]	Minimal running times [min]
1	0	9	8.5
2	2	13	10.3
3	6.5	15.5	8.5
4	8.5	25	16.5
5	23	32	8.5

Table 6-3: Timetable for 5 trains over the same track

Departure Headway [min]	Arrival Headway [min]
2	2
2	2
1.5	2
3.5	2
2	2

Table 6-4: headway constraints for the nominal order

$$r(k) = r(0) \otimes 30^{\otimes^k} \tag{6-1}$$

The autonomous behaviour of this example can be obtained with the equations in Section 2-2-2 so that:

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) = A_0^\star \otimes A_1 \otimes x(k-1) \tag{6-2}$$

The event times for the first four cycles, are obtained as follows:

$$x(0) = \begin{bmatrix} 0 \\ 2 \\ 6.5 \\ 8.5 \\ 23 \\ 9 \\ 13 \\ 15 \\ 25 \\ 32 \end{bmatrix}, \; x(1) = \begin{bmatrix} 25 \\ 27 \\ 29 \\ 31 \\ 35 \\ 34 \\ 37 \\ 39.5 \\ 47.5 \\ 49.5 \end{bmatrix}, \; x(2) = \begin{bmatrix} 37 \\ 39 \\ 41 \\ 43 \\ 47 \\ 51.5 \\ 53.5 \\ 55.5 \\ 59.5 \\ 61.5 \end{bmatrix}, \; x(3) = \begin{bmatrix} 49 \\ 51 \\ 53 \\ 55 \\ 59 \\ 63.5 \\ 65.5 \\ 67.5 \\ 71.5 \\ 73.5 \end{bmatrix}$$

By subtracting event times from consecutive cycles the cycle time behaviour is obtained:

$$x(1) - x(0) = \begin{bmatrix} 25 \\ 25 \\ 22.5 \\ 22.5 \\ 12 \\ 25 \\ 24.5 \\ 24.5 \\ 22.5 \\ 17.5 \end{bmatrix}, \; x(2) - x(1) = \begin{bmatrix} 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 17 \\ 16 \\ 16 \\ 12 \\ 12 \end{bmatrix}, \; x(3) - x(2) = \begin{bmatrix} 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 12 \end{bmatrix} \tag{6-3}$$

What can be observed from (6-3) is that after some initial transient behaviour, the cycles repeat within 12 minutes. This means that with a periodic timetable of 30 minutes, the timetable is stable.

If the trains are allowed to switch in the cycle $(k-1), (k)$ and $(k+1)$. The max-plus model for this system is [24]

$$x(k) = A_0(k, u(k)) \otimes x(k) \oplus A_1(k, u(k-1)) \otimes x(k-1) \oplus A_{-1}(k, u(k)) \otimes x(k+1) \oplus r(k) \quad (6\text{-}4)$$

$$x(k) = [d_1(k) \quad d_2(k) \quad d_3(k) \quad d_4(k) \quad d_5(k) \quad a_1(k) \quad a_2(k) \quad a_3(k) \quad a_4(k) \quad a_5(k)]^\top \quad (6\text{-}5)$$

$$A_{0,d}(k, u(k))) = \begin{bmatrix} \varepsilon & 2 \otimes \bar{u}_{2,1} & 2 \otimes \bar{u}_{3,1} & 3.5 \otimes \bar{u}_{4,1} & 2 \otimes \bar{u}_{5,1} \\ 2 \otimes u_{2,1} & \varepsilon & 2 \otimes \bar{u}_{3,2} & 3.5 \otimes \bar{u}_{4,2} & 2 \otimes \bar{u}_{5,2} \\ 2 \otimes u_{3,1} & 2 \otimes u_{3,2} & \varepsilon & 3.5 \otimes \bar{u}_{4,3} & 2 \otimes \bar{u}_{5,3} \\ 1.5 \otimes u_{4,1} & 1.5 \otimes u_{4,2} & 1.5 \otimes u_{4,3} & \varepsilon & 1.5 \otimes \bar{u}_{5,4} \\ 2 \otimes u_{5,1} & 2 \otimes u_{5,2} & 2 \otimes u_{5,3} & 3.5 \otimes u_{5,4} & \varepsilon \end{bmatrix} \quad (6\text{-}6)$$

$$A_{0,r}(k, u(k))) = \begin{bmatrix} \tau_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_4 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_5 \end{bmatrix} \quad (6\text{-}7)$$

$$A_{0,a}(k, u(k))) = \begin{bmatrix} \varepsilon & 2 \otimes \bar{u}_{2,1} & 2 \otimes \bar{u}_{3,1} & 2 \otimes \bar{u}_{4,1} & 2 \otimes \bar{u}_{5,1} \\ 2 \otimes u_{2,1} & \varepsilon & 2 \otimes \bar{u}_{3,2} & 2 \otimes \bar{u}_{4,2} & 2 \otimes \bar{u}_{5,2} \\ 2 \otimes u_{3,1} & 2 \otimes u_{3,2} & \varepsilon & 2 \otimes \bar{u}_{4,3} & 2 \otimes \bar{u}_{5,3} \\ 2 \otimes u_{4,1} & 2 \otimes u_{4,2} & 2 \otimes u_{4,3} & \varepsilon & 2 \otimes \bar{u}_{5,4} \\ 2 \otimes u_{5,1} & 2 \otimes u_{5,2} & 2 \otimes u_{5,3} & 2 \otimes u_{5,4} & \varepsilon \end{bmatrix} \quad (6\text{-}8)$$

$$A_0(k, u(k)) = \begin{bmatrix} A_{0,d}(k, u(k)) & \mathcal{E} \\ A_{0,r}(k, u(k))) & A_{0,a}(k, u(k)) \end{bmatrix} \quad (6\text{-}9)$$

With $A_{0,d}$, $A_{0,a}$ contains the respective departure and arrival headways and $A_{0,r}$ the minimal total running times from begin station A till end station C.

Headway constraints between departure events of trains from the current cycle and next cycle.

$$A_{-1,d}(k, u(k)) = \begin{bmatrix} 2 \otimes \bar{u}_{16}(k) & 2 \otimes \bar{u}_{12}(k) & 2 \otimes \bar{u}_{9}(k) & 3.5 \otimes \bar{u}_{7}(k) & 2 \otimes \bar{u}_{6}(k) \\ 2 \otimes \bar{u}_{21}(k) & 2 \otimes \bar{u}_{17}(k) & 2 \otimes \bar{u}_{13}(k) & 3.5 \otimes \bar{u}_{10}(k) & 2 \otimes \bar{u}_{8}(k) \\ 2 \otimes \bar{u}_{25}(k) & 2 \otimes \bar{u}_{22}(k) & 2 \otimes \bar{u}_{18}(k) & 3.5 \otimes \bar{u}_{14}(k) & 2 \otimes \bar{u}_{11}(k) \\ 1.5 \otimes \bar{u}_{28}(k) & 1.5 \otimes \bar{u}_{26}(k) & 1.5 \otimes \bar{u}_{23}(k) & 2 \otimes \bar{u}_{19}(k) & 1.5 \otimes \bar{u}_{15}(k) \\ 2 \otimes \bar{u}_{30}(k) & 2 \otimes \bar{u}_{29}(k) & 2 \otimes \bar{u}_{27}(k) & 3.5 \otimes \bar{u}_{24}(k) & 2 \otimes \bar{u}_{20}(k) \end{bmatrix} \quad (6\text{-}10)$$

Headway constraints between arrival events of trains from the current cycle and next cycle.

$$A_{-1,a}(k,u(k)) = \begin{bmatrix} 2\otimes \bar{u}_{16}(k) & 2\otimes \bar{u}_{12}(k) & 2\otimes \bar{u}_{9}(k) & 2\otimes \bar{u}_{7}(k) & 2\otimes \bar{u}_{6}(k) \\ 2\otimes \bar{u}_{21}(k) & 2\otimes \bar{u}_{17}(k) & 2\otimes \bar{u}_{13}(k) & 2\otimes \bar{u}_{10}(k) & 2\otimes \bar{u}_{8}(k) \\ 2\otimes \bar{u}_{25}(k) & 2\otimes \bar{u}_{22}(k) & 2\otimes \bar{u}_{18}(k) & 2\otimes \bar{u}_{14}(k) & 2\otimes \bar{u}_{11}(k) \\ 2\otimes \bar{u}_{28}(k) & 2\otimes \bar{u}_{26}(k) & 2\otimes \bar{u}_{23}(k) & 2\otimes \bar{u}_{19}(k) & 2\otimes \bar{u}_{15}(k) \\ 2\otimes \bar{u}_{30}(k) & 2\otimes \bar{u}_{29}(k) & 2\otimes \bar{u}_{27}(k) & 2\otimes \bar{u}_{24}(k) & 2\otimes \bar{u}_{20}(k) \end{bmatrix} \tag{6-11}$$

$$A_1(k,u(k)) = \begin{bmatrix} A_{-1,d}(k,u(k)) & \mathcal{E} \\ \mathcal{E} & A_{-1,a}(k,u(k)) \end{bmatrix} \tag{6-12}$$

Headway constraints between the departure events of trains from the current and previous cycle.

$$A_{1,d}(k,u(k-1)) =$$
$$\begin{bmatrix} 2\otimes u_{16}(k-1) & 2\otimes u_{12}(k-1) & 2\otimes u_{9}(k-1) & 3.5\otimes u_{7}(k-1) & 2\otimes u_{6}(k-1) \\ 2\otimes u_{21}(k-1) & 2\otimes u_{17}(k-1) & 2\otimes u_{13}(k-1) & 3.5\otimes u_{10}(k-1) & 2\otimes u_{8}(k-1) \\ 2\otimes u_{25}(k-1) & 2\otimes u_{22}(k-1) & 2\otimes u_{18}(k-1) & 3.5\otimes u_{14}(k-1) & 2\otimes u_{11}(k-1) \\ 1.5\otimes u_{28}(k-1) & 1.5\otimes u_{26}(k-1) & 1.5\otimes u_{23}(k-1) & 2\otimes u_{19}(k-1) & 1.5\otimes u_{15}(k-1) \\ 2\otimes u_{30}(k-1) & 2\otimes u_{29}(k-1) & 2\otimes u_{27}(k-1) & 3.5\otimes u_{24}(k-1) & 2\otimes u_{20}(k-1) \end{bmatrix} \tag{6-13}$$

Headway constraints between arrival events of trains from the current cycle and previous cycle.

$$A_{1,a}(k,u(k-1)) =$$
$$\begin{bmatrix} 2\otimes u_{16}(k-1) & 2\otimes u_{12}(k-1) & 2\otimes u_{9}(k-1) & 2\otimes u_{7}(k-1) & 2\otimes u_{6}(k-1) \\ 2\otimes u_{21}(k-1) & 2\otimes u_{17}(k-1) & 2\otimes u_{13}(k-1) & 2\otimes u_{10}(k-1) & 2\otimes u_{8}(k-1) \\ 2\otimes u_{25}(k-1) & 2\otimes u_{22}(k-1) & 2\otimes u_{18}(k-1) & 2\otimes u_{14}(k-1) & 2\otimes u_{11}(k-1) \\ 2\otimes u_{28}(k-1) & 2\otimes u_{26}(k-1) & 2\otimes u_{23}(k-1) & 2\otimes u_{19}(k-1) & 2\otimes u_{15}(k-1) \\ 2\otimes u_{30}(k-1) & 2\otimes u_{29}(k-1) & 2\otimes u_{27}(k-1) & 2\otimes u_{24}(k-1) & 2\otimes u_{20}(k-1) \end{bmatrix} \tag{6-14}$$

$$A_1(k,u(k-1)) = \begin{bmatrix} A_{1,d}(k,u(k-1)) & \mathcal{E} \\ \mathcal{E} & A_{1,a}(k,u(k-1)) \end{bmatrix} \tag{6-15}$$

No known delays from the past, so the control inputs in $A_1(k,u(k-1))$ are defined as $u_i(k-1)=0,\ \forall i$.

The current cycle is cycle 1. The control horizon will be set to be 40 minutes. So at time instance $\mathcal{X}(25)$ the following events are happening:

$$\begin{aligned} \mathcal{X}(25) = \ & [a_4(0) \quad a_5(0) \quad d_1(1) \quad d_2(1) \quad d_3(1) \quad d_4(1) \\ & \quad d_5(1) \quad a_1(1) \quad a_2(1) \quad a_3(1) \quad a_4(1) \quad a_5(1) \quad d_1(2) \quad d_2(2)]^\top \end{aligned} \tag{6-16}$$

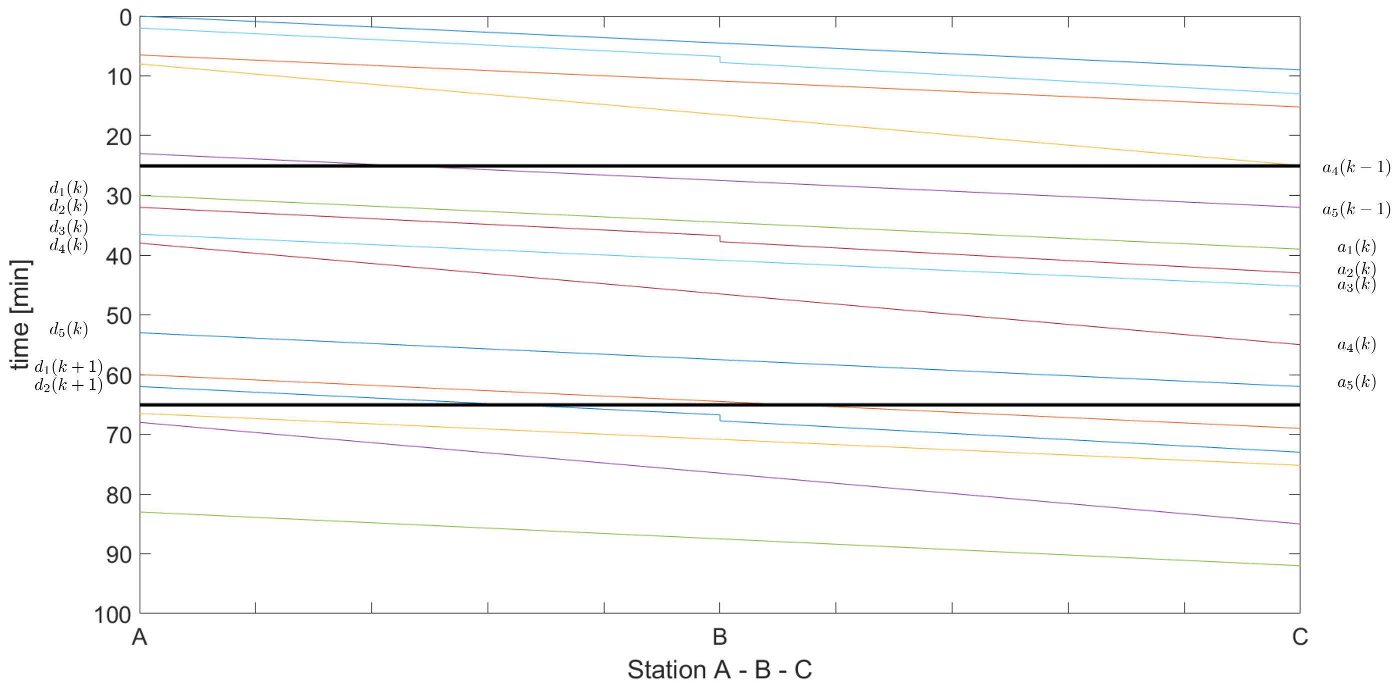


Figure 6-5: Time distance path of trains who are in the prediction horizon for the nominal timetable

As can be seen, all event times from x(1) are scheduled in the interval $(25, 65]$ and a part of $x(0)$ and a part of $x(2)$. From Figure 6-5 can be seen that train $a_{5_{-1}}$ should be on the track at t=25, however, what is not shown in this figure is the delay scenario created. The delay scenario implies that train $a_{4_{-1}}$ stands still for a red sign, two blocks ahead of station A. Due to this scenario is the train with arrival time $a_{5_{-1}}$ still waiting at station A for a red sign, because the block section ahead is not cleared by the freight train. This scenario is pictured in Figure 6-6.

Since the current cycle is defined as cycle 1, the event times from the previous cycle that did not finish yet ($a_4(0)$ and $a_5(0)$) are taken into account in the constraints because there order cannot change anymore.

Since the max-plus binary variables u_i associated to $d_3(2) - d_5(2)$ and $a_1(2) - a_5(2)$ are outside the prediction horizon, they will be set to zero. Simplifying matrices $A_{-1,d}(k, u(k))$ and $A_{-1,a}(k, u(k))$ to:

$$A_{-1,d}(k, u(k)) = \begin{bmatrix} 2 \otimes u_{16}(k) & 2 \otimes u_{12}(k) & \varepsilon & \varepsilon & \varepsilon \\ 2 \otimes u_{21}(k) & 2 \otimes u_{17}(k) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \tag{6-17}$$

and,

$$A_{-1,a}(k, u(k)) = \mathcal{E} \tag{6-18}$$

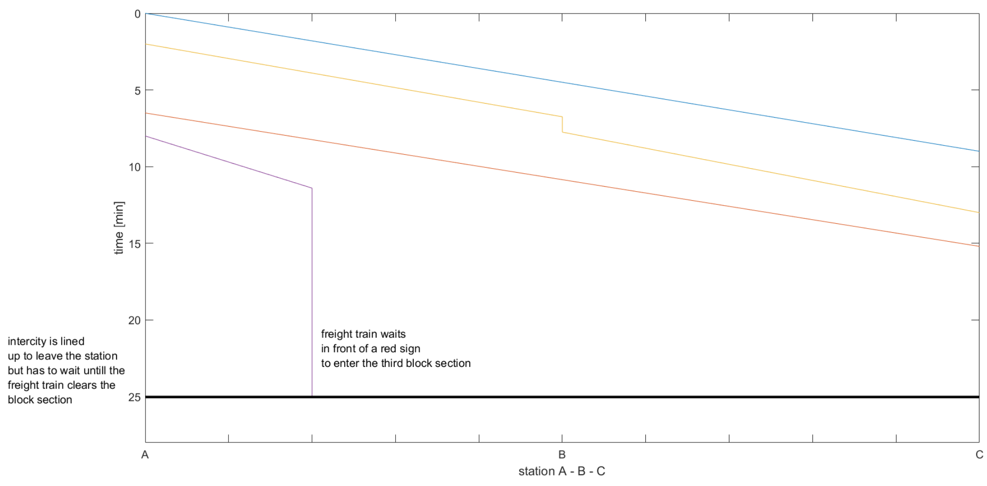


Figure 6-6: delay scenario for trains on the track

The control variables that can be changed are:

$$\psi(25) = \begin{bmatrix} u_{2,1}(1) & u_{3,1}(1) & u_{4,1}(1) & u_{5,1}(1) & u_{3,2}(1) & u_{4,2}(1) & u_{5,2}(1) \\ & u_{4,3}(1) & u_{5,3}(1) & u_{5,4}(1) & u_{16}(1) & u_{12}(1) & u_{21}(1) & u_{17}(1) \end{bmatrix}^\top \tag{6-19}$$

the SMPL model becomes:

$$\begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \\ d_4(1) \\ d_5(1) \\ a_1(1) \\ a_2(1) \\ a_3(1) \\ a_4(1) \\ a_5(1) \\ d_1(2) \\ d_2(2) \end{bmatrix} = \begin{bmatrix} A_0(1,u(1)) & \mathcal{E} \\ \mathcal{E} & A_0(2,u(2)) \end{bmatrix} \otimes \begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \\ d_4(1) \\ d_5(1) \\ a_1(1) \\ a_2(1) \\ a_3(1) \\ a_4(1) \\ a_5(1) \\ d_1(2) \\ d_2(2) \end{bmatrix} \oplus$$
$$\oplus \begin{bmatrix} A_1(1,u(0)) & \mathcal{E} \\ \mathcal{E} & A_{1,a}(2,u(1)) \end{bmatrix} \otimes \begin{bmatrix} x(0) \\ x(1) \end{bmatrix} \oplus \begin{bmatrix} A_{-1}(1,u(1)) & \mathcal{E} \\ \mathcal{E} & \mathcal{E} \end{bmatrix} \otimes \begin{bmatrix} x(2) \\ x(3) \end{bmatrix} \oplus r_{\text{timetable}} \tag{6-20}$$

Note that the A matrices are not all full sized, as defined for all event times, but scaled accordingly so that only the relevant headway constraints are taken into account valid for the state defined for the control interval.

6-3-2 Model simplification and include abstraction times

The model put forward in (6-4) is valid for all cycles k and a schedule can be obtained for a whole day, week or more. This is, however, not the aim of this case study, where we want to optimize a part of the schedule for all current and a few upcoming trains. It is therefore desired to simplify the model of (6-4) so that we end up with a model that correspond to the model in (6-20) but without the dependency of cycle counter k. Furthermore, instead of expressing the prediction and control horizon in a time frame, we would rather express them as the number of trains we are optimizing at a time instance t. We do this because then all relevant information for rescheduling is actually present in the model and a weighted decision can be equally made for all trains in the optimization. The goal is a model that is only valid for the events in the prediction and control horizon, however with the benefit that we can model the whole control and prediction horizon as one cycle.
Instead of having a control horizon of 40 minutes as shown in Figure 6-5, we want to control the order of the first 7 trains that can be reordered. Since at time $t = 25$, there are already two trains on the track that can no longer be reordered (since the IC, after the stopped freight train, is already at the track waiting to leave this IC is considered to be 'on the track'), the control variables are already fixed, but are still included in the constraints. As can be seen in (6-21), all events are virtually assumed to be in cycle 1 thereby simplifying the model of (6-4) so that only the matrix of $A_0(k, u(k))$ is preserved for the events that can be rescheduled. The events that are included in the control horizon are given by:

$$\begin{aligned} \mathcal{X}(25) = \ [a_{4_{-1}}(1) \quad a_{5_{-1}}(1) \quad d_1(1) \quad d_2(1) \quad d_3(1) \quad d_4(1) \quad d_5(1) \\ d_6(1) \quad d_7(1) \quad a_1(1) \quad a_2(1) \quad a_3(1) \quad a_4(1) \quad a_5(1) \quad a_6(1) \quad a_7(1)]^\top \end{aligned} \tag{6-21}$$

The new SMPL model with all events virtually in the same cycle becomes:

$$\begin{bmatrix} a_{4_{-1}}(1) \\ a_{5_{-1}}(1) \\ \hline d_1(1) \\ d_2(1) \\ d_3(1) \\ d_4(1) \\ d_5(1) \\ d_6(1) \\ d_7(1) \\ a_1(1) \\ a_2(1) \\ a_3(1) \\ a_4(1) \\ a_5(1) \\ a_6(1) \\ a_7(1) \end{bmatrix} = \left[\begin{array}{c|c} A_0(1,1) & \mathcal{E} \\ A_{c0}(1,1) & A_0(1,u(1)) \end{array}\right] \otimes \begin{bmatrix} a_{4_{-1}}(1) \\ a_{5_{-1}}(1) \\ \hline d_1(1) \\ d_2(1) \\ d_3(1) \\ d_4(1) \\ d_5(1) \\ d_6(1) \\ d_7(1) \\ a_1(1) \\ a_2(1) \\ a_3(1) \\ a_4(1) \\ a_5(1) \\ a_6(1) \\ a_7(1) \end{bmatrix} \oplus r_{\text{timetable}} \oplus A_{\text{abstr}_r} \tag{6-22}$$

where $A_0(1,u(1)) \in \mathbb{R}_{\max}^{3\times3}$ from (6-9) is adjusted and extended with the two new trains. The information about these trains can be extracted from Table 6-3 and Table 6-4 since the timetable is periodic with 30 minutes. Matrix $A_0(1,1)$ contains the arrival headways for the trains on the track, but no control variables are associated with it since these trains cannot be rescheduled anymore at time $t = 25$. The arrival headway between the two trains that are on the track is 3, and since the order is fixed, $A_0(1,1)$ becomes:

$$A_0(1,1) = \begin{bmatrix} \varepsilon & \varepsilon \\ 2 & \varepsilon \end{bmatrix} \tag{6-23}$$

Matrix $A_{c0} \in \mathbb{R}_{\max}^{16\times3}$ is the connection matrix that connects the headways of the trains that are going to be scheduled and the trains which cannot be rescheduled anymore. Since this hold for trains that are already on the track, only an arrival event can still take place for which a headway should be defined. Matrix $A_0(1,1)$ fixes the order of the trains on the track. Since all trains are modeled into one cycle, the arrival headway of the last train on the track (in this case train $a_{5_{-1}}$ is the last one) is coupled via the minimal running time with the departures of the trains to be rescheduled as follows:

$$A_{c0}(1,1) = \begin{bmatrix} (A_{1,a,\text{last, row}} \otimes A_{0,-r})^\top & \mathcal{E} \\ \mathcal{E} & \end{bmatrix} \tag{6-24}$$

where $A_{1,a,\text{last, row}}$ contains only the arrival headway's of the last train on the track and all other trains that come next, in this case that is the fifth row of $A_{1,a}$:

$$A_{1,a,\text{last, row}} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \end{bmatrix} \tag{6-25}$$

Matrix $A_{0,-r}$ is the same as (6-7) but with the slight difference being that all τ's are replace with $-\tau$'s, this is necessary because only the arrival headways are included. Since if a train is delayed, the arrival time related to this train shifts forward, but not necessarily the departure time, therefore the delay is spread out through the arrival times of the trains on the track.

$$A_{\text{abstr}_r} = \left[\begin{array}{c} r_{a_{4\_1}\text{minimal}} \\ r_{a_{5\_1}\text{minimal}} \\ \hline r_{d_1\ \text{minimal}} \\ \vdots \\ r_{d_7\ \text{minimal}} \\ \mathcal{E} \end{array}\right] \tag{6-26}$$

Matrix A_{abstr_r} represents the values that follow from the actual conditions on the track. These values are the fastest time for which the trains could arrive at their end station. The values in the upper part of the vector $r_{a_{4\_1}\text{minimal}}$ and $r_{a_{5\_1}\text{minimal}}$ follow direct from the abstraction for the trains on the track. The event times in the lower part follow from the abstraction of another zone in the railway network. In this case study, it is assumed that these trains do not encounter a delay and they could depart at station A at their scheduled departure times. Note that we assume a buffer area at station A where the trains could wait for departure without blocking other trains.

6-4 Constraints

Writing out the constraints for (6-22) results in the mixed-integer linear constraints for which the SMPL model is solved. These constraints are given as:

$$
\begin{aligned}
a_{4_{-1}}(1) &= 29.5 \oplus t \otimes \tau_{abs,4}\\
a_{5_{-1}}(1) &= 33.5 \oplus t \otimes \tau_{abs,5} \oplus 4 \otimes a_{4_{-1}}\\
d_1(1) =& 2 \otimes \bar{u}_{2,1}(1) \otimes d_2 \oplus 2 \otimes \bar{u}_{3,1}(1) \otimes d_3 \oplus 3.5 \otimes \bar{u}_{4,1}(1) \otimes d_4 \oplus 2 \otimes \bar{u}_{5,1}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,1}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,1}(1) \otimes d_7 \oplus -\tau_1 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 30\\
d_2(1) =& 2 \otimes u_{2,1}(1) \otimes d_1 \oplus 2 \otimes \bar{u}_{3,2}(1) \otimes d_3 \oplus 3.5 \otimes \bar{u}_{4,2}(1) \otimes d_4 \oplus 2 \otimes \bar{u}_{5,2}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,2}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,2}(1) \otimes d_7 \oplus -\tau_2 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 34\\
d_3(1) =& 2 \otimes u_{3,1}(1) \otimes d_1 \oplus 2 \otimes u_{3,2}(1) \otimes d_2 \oplus 3.5 \otimes \bar{u}_{4,3}(1) \otimes d_4 \oplus 2 \otimes \bar{u}_{5,3}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,3}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,3}(1) \otimes d_7 \oplus -\tau_3 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 39\\
d_4(1) =& 1.5 \otimes u_{4,1}(1) \otimes d_1 \oplus 1.5 \otimes u_{4,2}(1) \otimes d_2 \oplus 1.5 \otimes u_{4,3}(1) \otimes d_3 \oplus 1.5 \otimes \bar{u}_{5,4}(1) \otimes d_5\\
&\oplus 1.5 \otimes \bar{u}_{6,4}(1) \otimes d_6 \oplus 1.5 \otimes \bar{u}_{7,4}(1) \otimes d_7 \oplus -\tau_4 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 43\\
d_5(1) =& 2 \otimes u_{5,1}(1) \otimes d_1 \oplus 2 \otimes u_{5,2}(1) \otimes d_2 \oplus 2 \otimes u_{5,3}(1) \otimes d_3 \oplus 2 \otimes u_{5,4}(1) \otimes d_4\\
&\oplus 3.5 \otimes \bar{u}_{6,5}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,5}(1) \otimes d_7 \oplus -\tau_5 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 55\\
d_6(1) =& 2 \otimes u_{6,1}(1) \otimes d_1 \oplus 2 \otimes u_{6,2}(1) \otimes d_2 \oplus 2 \otimes u_{6,3}(1) \otimes d_3 \oplus 2 \otimes u_{6,4}(1) \otimes d_4\\
&\oplus 3.5 \otimes u_{6,5}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,6}(1) \otimes d_7 \oplus -\tau_6 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 60\\
d_7(1) =& 2 \otimes u_{7,1}(1) \otimes d_1 \oplus 2 \otimes u_{7,2}(1) \otimes d_2 \oplus 2 \otimes u_{7,3}(1) \otimes d_3 \oplus 3.5 \otimes u_{7,4}(1) \otimes d_4\\
&\oplus 2 \otimes \bar{u}_{7,5}(1) \otimes d_6 \oplus 2 \otimes u_{7,6}(1) \otimes d_7 \oplus -\tau_7 \otimes \oplus 2 \otimes a_{5_{-1}} \oplus 64\\
a_1(1) =& \tau_1 \otimes d_1 \oplus 2 \otimes \bar{u}_{2,1}(1) \otimes d_2 \oplus 2 \otimes \bar{u}_{3,1}(1) \otimes d_3 \oplus 2 \otimes \bar{u}_{4,1}(1) \otimes d_4 \oplus 2 \otimes \bar{u}_{5,1}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,1}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,1}(1) \otimes d_7 \oplus 39\\
a_2(1) =& \tau_2 \otimes d_2 \oplus 2 \otimes u_{2,1}(1) \otimes d_1 \oplus 2 \otimes \bar{u}_{3,2}(1) \otimes d_3 \oplus 2 \otimes \bar{u}_{4,2}(1) \otimes d_4 \oplus 2 \otimes \bar{u}_{5,2}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,2}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,2}(1) \otimes d_7 \oplus 45\\
a_3(1) =& \tau_4 \otimes d_3 \oplus 2 \otimes u_{3,1}(1) \otimes d_1 \oplus 2 \otimes u_{3,2}(1) \otimes d_2 \oplus 2 \otimes \bar{u}_{4,3}(1) \otimes d_4 \oplus 2 \otimes \bar{u}_{5,3}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,3}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,3}(1) \otimes d_7 \oplus 49\\
a_4(1) =& \tau_4 \otimes d_4 \oplus 2 \otimes u_{4,1}(1) \otimes d_1 \oplus 2 \otimes u_{4,2}(1) \otimes d_2 \oplus 2 \otimes u_{4,3}(1) \otimes d_3 \oplus 2 \otimes \bar{u}_{5,4}(1) \otimes d_5\\
&\oplus 2 \otimes \bar{u}_{6,4}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,4}(1) \otimes d_7 \oplus 59.5\\
a_5(1) =& \tau_5 \otimes d_5 \oplus 2 \otimes u_{5,1}(1) \otimes d_1 \oplus 2 \otimes u_{5,2}(1) \otimes d_2 \oplus 2 \otimes u_{5,3}(1) \otimes d_3 \oplus 2 \otimes u_{5,4}(1) \otimes d_4\\
&\oplus 2 \otimes \bar{u}_{6,5}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,5}(1) \otimes d_7 \oplus 63.5\\
a_6(1) =& \tau_5 \otimes d_5 \oplus 2 \otimes u_{5,1}(1) \otimes d_1 \oplus 2 \otimes u_{5,2}(1) \otimes d_2 \oplus 2 \otimes u_{5,3}(1) \otimes d_3 \oplus 2 \otimes u_{5,4}(1) \otimes d_4\\
&\oplus 2 \otimes u_{5,5}(1) \otimes d_6 \oplus 2 \otimes \bar{u}_{7,6}(1) \otimes d_7 \oplus 69\\
a_7(1) =& \tau_5 \otimes d_5 \oplus 2 \otimes u_{5,1}(1) \otimes d_1 \oplus 2 \otimes u_{5,2}(1) \otimes d_2 \oplus 2 \otimes u_{5,3}(1) \otimes d_3 \oplus 2 \otimes u_{5,4}(1) \otimes d_4\\
&\oplus 2 \otimes u_{5,5}(1) \otimes d_6 \oplus 2 \otimes u_{6,5}(1) \otimes d_7 \oplus 75
\end{aligned}
\tag{6-27}
$$

With the max-plus binary variables transformed into conventional binary variables and max-plus algebra transformed into conventional algebra, this leads to the following set of constraints of the SMPL model:

$$d_1 \geq \begin{cases} 2+(1-v_{2,1})\beta+d_2 \\ 2+(1-v_{3,1})\beta+d_3 \\ 3.5+(1-v_{4,1})\beta+d_4 \\ 2+(1-v_{5,1})\beta+d_5 \\ 2+(1-v_{6,1})\beta+d_6 \\ 2+(1-v_{7,1})\beta+d_7 \\ a_{5_{-1}}-\tau_1+2 \\ 30 \end{cases} \quad d_2 \geq \begin{cases} 2+v_{2,1}\beta+d_1 \\ 2+(1-v_{3,2})\beta+d_3 \\ 3.5+(1-v_{4,2})\beta+d_4 \\ 2+(1-v_{5,2})\beta+d_5 \\ 2+(1-v_{6,2})\beta+d_6 \\ 2+(1-v_{7,2})\beta+d_7 \\ a_{5_{-1}}-\tau_2+2 \\ 32 \end{cases} \quad d_3 \geq \begin{cases} 2+v_{3,1}\beta+d_1 \\ 2+v_{3,2}\beta+d_2 \\ 3.5+(1-v_{4,3})\beta+d_4 \\ 2+(1-v_{5,3})\beta+d_5 \\ 2+(1-v_{6,3})\beta+d_6 \\ 2+(1-v_{7,3})\beta+d_7 \\ a_{5_{-1}}-\tau_3+2 \\ 36.5 \end{cases} \tag{6-28}$$

$$d_4 \geq \begin{cases} 1.5+v_{4,1}\beta+d_1 \\ 1.5+v_{4,2}\beta+d_2 \\ 1.5+v_{4,3}\beta+d_3 \\ 1.5+(1-v_{5,4}\beta+d_5 \\ 1.5+(1-v_{6,4})\beta+d_6 \\ 1.5+(1-v_{7,4})\beta+d_7 \\ a_{5_{-1}}-\tau_4+2 \\ 38.5 \end{cases} \quad d_5 \geq \begin{cases} 2+v_{5,1}\beta+d_1 \\ 2+v_{5,2}\beta+d_2 \\ 2+v_{5,3}\beta+d_3 \\ 3.5+v_{5,4}\beta+d_4 \\ 2+(1-v_{6,5})\beta+d_6 \\ 2+(1-v_{7,5})\beta+d_7 \\ a_{5_{-1}}-\tau_5+2 \\ 53 \end{cases} \quad d_6 \geq \begin{cases} 2+v_{6,1}\beta+d_1 \\ 2+v_{6,2}\beta+d_2 \\ 2+v_{6,3}\beta+d_3 \\ 3.5+v_{6,4}\beta+d_4 \\ 2+v_{6,5}\beta+d_6 \\ 2+(1-v_{7,6})\beta+d_7 \\ a_{5_{-1}}-\tau_6+2 \\ 60 \end{cases} \tag{6-29}$$

$$d_7 \geq \begin{cases} 2+v_{7,1}\beta+d_1 \\ 2+v_{7,2}\beta+d_2 \\ 2+v_{7,3}\beta+d_3 \\ 3.5+v_{7,4}\beta+d_4 \\ 2+v_{7,5}\beta+d_6 \\ 2+v_{7,6}\beta+d_7 \\ a_{5_{-1}}-\tau_7+2 \\ 62 \end{cases} \quad a_1 \geq \begin{cases} \tau_1+d_1 \\ 2+(1-v_{2,1})\beta+d_2 \\ 2+(1-v_{3,1})\beta+d_3 \\ 2+(1-v_{4,1})\beta+d_4 \\ 2+(1-v_{5,1})\beta+d_5 \\ 2+(1-v_{6,1})\beta+d_6 \\ 2+(1-v_{7,1})\beta+d_7 \\ 39 \end{cases} \quad a_2 \geq \begin{cases} \tau_2+d_2 \\ 2+v_{2,1}\beta+d_1 \\ 2+(1-v_{3,2})\beta+d_3 \\ 2+(1-v_{4,2})\beta+d_4 \\ 2+(1-v_{5,2})\beta+d_5 \\ 2+(1-v_{6,2})\beta+d_6 \\ 2+(1-v_{7,2})\beta+d_7 \\ 43.5 \end{cases} \tag{6-30}$$

$$a_3 \geq \begin{cases} \tau_3+d_3 \\ 2+v_{3,1}\beta+d_1 \\ 2+v_{3,2}\beta+d_2 \\ 2+(1-v_{4,3})\beta+d_4 \\ 2+(1-v_{5,3})\beta+d_5 \\ 2+(1-v_{6,3})\beta+d_6 \\ 2+(1-v_{7,3})\beta+d_7 \\ 45.5 \end{cases} \quad a_4 \geq \begin{cases} \tau_4+d_4 \\ 2+v_{4,1}\beta+d_1 \\ 2+v_{4,2}\beta+d_2 \\ 2+v_{4,3}\beta+d_3 \\ 2+(1-v_{5,4}\beta+d_5 \\ 2+(1-v_{6,4})\beta+d_6 \\ 2+(1-v_{7,4})\beta+d_7 \\ 55 \end{cases} \quad a_5 \geq \begin{cases} \tau_5+d_5 \\ 2+v_{5,1}\beta+d_1 \\ 2+v_{5,2}\beta+d_2 \\ 2+v_{5,3}\beta+d_3 \\ 2+v_{5,4}\beta+d_4 \\ 2+(1-v_{6,5})\beta+d_6 \\ 2+(1-v_{7,5})\beta+d_7 \\ 62 \end{cases} \tag{6-31}$$

$$a_6 \geq \begin{cases} \tau_6 + d_6 \\ 2 + v_{6,1}\beta + d_1 \\ 2 + v_{6,2}\beta + d_2 \\ 2 + v_{6,3}\beta + d_3 \\ 2 + v_{6,4}\beta + d_4 \\ 2 + v_{6,5}\beta + d_6 \\ 2 + (1 - v_{7,6})\beta + d_7 \\ 69 \end{cases} \qquad a_7 \geq \begin{cases} \tau_7 + d_7 \\ 2 + v_{7,1}\beta + d_1 \\ 2 + v_{7,2}\beta + d_2 \\ 2 + v_{7,3}\beta + d_3 \\ 2 + v_{7,4}\beta + d_4 \\ 2 + v_{7,5}\beta + d_6 \\ 2 + v_{7,6}\beta + d_7 \\ 73 \end{cases} \qquad \begin{aligned} a_{4_{-1}} &= \begin{cases} 38.5 \end{cases} \\ a_{5_{-1}} &= \begin{cases} 33.5 \\ 2 + a_{4_{-1}} \end{cases} \end{aligned} \tag{6-32}$$

Now we can formulate the optimization problem where the objective is to minimize the total delay, the total departure and arrival delays at the station. To order the constraints in a structured way, the vector z contains all events that are in the control horizon and all the control variables as:

$$z = [\underbrace{a_{4_{-1}} \quad a_{5_{-1}} \quad d_1 \dots d_7 \quad a_1 \dots a_7}_{x} \quad \underbrace{v_{2,1} \dots v_{7,6}}_{v}]^\top \tag{6-33}$$

For the seven trains, the objective function that is minimized is the following:

$$\min_z \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^\top \cdot \begin{bmatrix} z_1 \\ z_2 \\ z_3 - r_3 \\ z_4 - r_4 \\ \vdots \\ z_{16} - r_{16} \\ z_{17} \\ \vdots \\ z_{35} \end{bmatrix} \tag{6-34}$$

s.t. the constraints defined above for the SMPL model hold. When this model is optimized for z, the resulting vector shows that no reordering takes place because no delays are introduced in the case study yet i.e. the binary control variables $z_{17} \dots z_{35}$ are all equal to 1. The first three events of z do not show up in the objective function because they cannot be controlled and are only present in the constraints.

6-5 delays

In this section, we are going to study the scheduling actions when a delay is introduced in the system. We have one train on the track that cannot be rescheduled. The delay is introduced as follows, the freight train on the track is assumed to come to standstill before the entrance of the third block section due to a disruption that causes a red signal. From there the train has to wait until the signal turns green after which the train is allowed to travel to station C. Time $t = 25$ is chosen to be the start of the optimization, furthermore at this time instant, the traffic light turns green for the freight train on the track.

6-5-1 Obtaining the minimal running times for trains on the track section

Since the train has come to standstill, the timed automaton starts with q_1, then switches to q_2, q_{4_a} and to q_5 since the coasting phase is not involved. We know that the block sections are 1500 m each, so the train has 3000 m traveled and 12000 m to cover since the total distance of the track is 15000 m. With a single maximum speed limit, the switching points are obtained just as in Section 4-3-1. With the switching points for the hybrid system known, the minimal trajectory is obtained via abstraction of the hybrid automaton. The minimal running time at $t = 25$ for the train on the track becomes:

conditions at $t = 25$	train $a_{4_{-1}}$	train $a_{5_{-1}}$
minimal running time to reach station C	13.5 minutes	8.5 minutes
initial conditions (s, v)	(3000,0)	(0,0)

Table 6-5: minimal running times

where $\tau'_{\min}$ in Table 6-5 represents not the total minimal running time from station A to C, but the remaining running time from time point $t = 25$.

The minimal running time of the train is implemented in the mixed-integer linear constraints of (6-22) where event time $a_{5_{-1}}$ is adjusted with this information.

This results in a model that is heavily delayed. The optimization showed that reordering takes place by changing the order of trains 1,2 and 3 as can be observed in the second part of vector z where the control variables are located.

$$v = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{6-35}$$

The control variables in (6-35) indicate that all seven trains are reordered so that the order of the trains is now [2 3 1 4 5 6 7] instead of [1 2 3 4 5 6 7].

The total arrival delay for the seven trains is for this configuration as follows:

train ordering	2	3	1	4	5	6	7
delay [min]	0	3.1	11.9	5.2	0.3	0	0

This adds up to a total arrival delay of 20.5 minutes.
To set this total arrival delay in perspective, we need to obtain the delay propagation over time in the uncontrolled case i.e. the trajectory planner plans the trajectories according to the nominal timetable. This results in a total arrival delay of 34.2 minutes for the seven trains in the control horizon.

6-5-2 Include energy consumption

The objective function in (6-34) assumes that the weights for all trains are equal. However, the trajectory planner deals with making the trajectories of the trains energy efficient given

the track constraints and the constraints given by the scheduler. It is therefore desired to include a sense of energy efficient driving into the scheduling level. The more energy a train consumes when it needs rescheduling, the higher the weight on the delays in the objective function. The weight is the resistance to rescheduling or delaying a train. The train type that consumes the most energy is obviously the freight train. This train does not stop at the stations in the control horizon of the scheduler. If this train should make an unscheduled stop or change speed this would cost a lot of energy. The stop train already stops at every station so it does not cost extra energy to reschedule this train and therefore this weight is not adjusted. In this case study, the intercity and the freight train make scheduled stops. A non-linear weight is proposed for trains that does not make scheduled stops at the control stations. This weight starts initially high and remains constant until the delay for that train equals the breaking time since then the extra energy is already at its maximum and cannot increase further. If a delay is larger than the stopping time for a nonstopping train, the train must make an unscheduled stop, the weight is dropped and made equal with the other trains that make a scheduled stop since the train is already taking up its extra energy consumption and extra delay which does not add up to the extra energy consumption. So, the weight is a constant up until the delay is equal to the braking time of this train after which the weight drops to a weight that is equal to the other trains.
A continuous function is desired and therefore the weight function has the following form:

$$w_f = w_{f_0} + \frac{-w_{f_0} + 1}{1 + e^{t_{br} - d}} \tag{6-36}$$

where w_{f_0} is the approximate initial weight, delay $d = z - r$ is zero, t_{br} is the braking time for the freight train to come to a stop.

However, this weight function is not ideal. Since the weight for this train is no longer linear, the MILP is turned into a mixed-integer nonlinear programming (MINLP) problem which is not desired since only one train benefits from this complex weight and when this weight is implemented the optimization can no longer benefit from the relaxation to the MINLP formulation. Instead of using this complex weight, a linear weight is proposed that is considered valid for all delays and therefore the weight w_{f_0} is lowered by a factor so that the weight can be implemented linearly, but now valid for the entire region such that the MILP formulation is still valid and we can still take into account some energy consumption.

$$w_{fl} = \frac{w_{f_0}}{f_1} \tag{6-37}$$

where w_{fl} is the linear weight function for the ongoing trains and f_1 a weighting factor.

However, since all trains in this case study are said to make a scheduled stop at stations A and C due to the trajectory planner limitations, we cannot directly alter the weights for a train. To simulate the energy efficient behaviour and circumvent the trajectory planner limitations, we propose a higher weight for the freight train despite the fact that this trains already makes a scheduled stop. By showing that this train is more resistant to rescheduling, possibly another optimum can be found for the rescheduling control variables.
With the new weight for the freight train (set to 5), another optimization is done and indeed a new optimal ordering is obtained with the new control vector:

$$v = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{6-38}$$

indicating a new ordering, now with the first four trains are reordered by the scheduler [2 3 4 1 5 6 7]. We observe that in contrast to the objective with normalized weights that the freight train has more resistance to being delayed and therefore the order is different than with the normalized weights. It can clearly be observed from Figure 6-10 that the freight train has less delay according to the main timetable, however the total delay of all trains is higher compared to the order obtained with equalized weights, shown in Figure 6-9.

The total arrival delay for the seven trains is for this configuration as follows:

train ordering	2	3	4	1	5	6	7
delay [min]	0	3.1	3.1	21.2	0.5	0	0

This adds up to a total arrival delay of 27.9 minutes. As expected the total delay is larger compared to the optimization with equal weights for for all trains. The freight train has much less delay. However, by inspecting this delay we noticed that it is actually still to high, because 3 minutes is larger than the stopping time for this train. It is possible to completely remove the delay of the freight train at the cost of a total arrival delay of 32.2 minutes and the freight train is in this situation shifted forward to the second position. Although from an energy efficient point of view this is the best solution, it is not from a time optimal point of view. For this case study we are satisfied with the solution found in (6-38), since it showed that we can actually increase the weights in the objective function to shift the, optimized for time schedule, towards a more energy efficient schedule. Note that we are not able within the limits of this case study to specify exactly what the weights should be for in the energy efficient case.

6-6 Trajectory planner

The trajectory planner designed for this case study is built from basic functions to provide just enough performance to complete the task. We stress that we did not focus on coming up with the most ideal planner since that is not the purpose of this thesis. We wanted to show that performance of the total system can be improved by performing a rescheduling action at regular time intervals based on real-time information. This planner can calculate a trajectory for a train with a single maximum speed. The track is divided into block sections that provide safe distances between all trains as described before. The planner fits feasible trajectories for the trains with information about the departure and arrival events from the scheduling level. The rules for the planner given are:

- minimize coasting points.
- minimize the total run time.
- trains cannot depart or arrive before the time given by the normal schedule.
- the scheduler is only allowed to shift the arrival and departure times if needed, but does not perform any control actions.

- the track is divided into fixed block sections.
- a train is not allowed to cruise at a speed lower than the speed limit.
- trains are delayed at the departure station until it 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 block distance.

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 part of the objective function to ensure that a train leaves the departure station as soon as its allowed by the constraints. Both items are contained in the objective function of the trajectory planner. The objective function can minimize the running time of the trains as well as extend the running time by minimizing the coasting point due to the constraints for safety and departure and arrival events. When no schedule is provided, the planner plans the trains as early as possible. This can be viewed in an example where four trains are running over the same track, the first is an intercity, second a freight train and the last two trains are also intercity's. For safety reasons due to signaling, the intercity's 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). Also, the distance between a freight train preceding a (faster) intercity is required to be at least two block sections due to safety and signaling

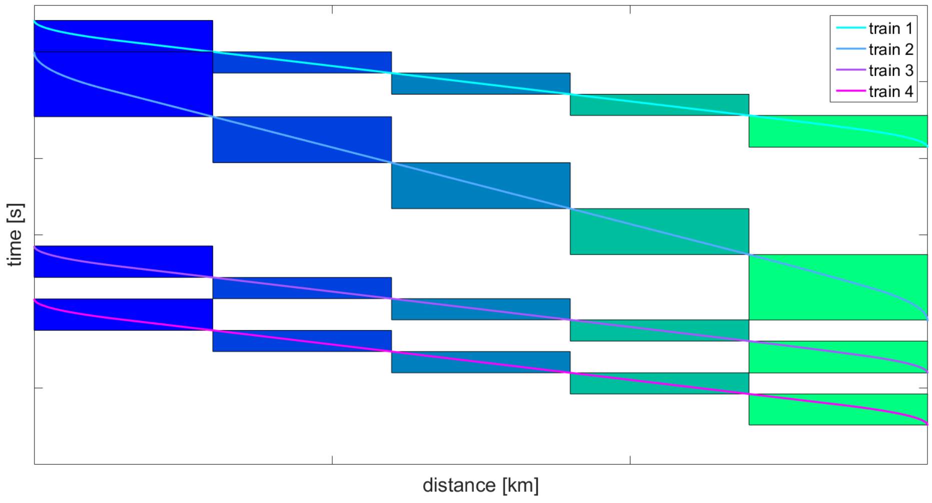


Figure 6-7: Planning as fast as possible without timetable

The objective function has a penalty function for when the trains are planned with a delay to the main schedule. This means that the weight for minimal running time is more important than the weight for energy efficient driving.

Now back to the case study where the seven trains are rescheduled for a disturbance scenario. The most important part of the scheduler is the control decisions that are made. These control decisions capture the behaviour of the trains in the railway network. In addition to the control decisions, the scheduler obviously outputs a schedule, however since the trajectory planner refines this schedule based on the control decisions and the main schedule. If no iterations were planned for the schedule put forward by the scheduler, then the planner could use this interim schedule from the scheduler for the trajectories of the trains. However, this interim schedule is, in this framework, not used since the main results of the scheduler is the control decisions that are made and not the schedule. The planner is much better equipped to obtain the correct event times with the optimal control decisions. In this way only the information where the control blocks are dedicated is collected and communicated. The headway times contains information about the microscopic behaviour of the network and the control decisions contain information about the macroscopic behaviour of the railway network. As can be seen in Figure 6-7, the headway times for this particular order of the trains can be directly obtained.

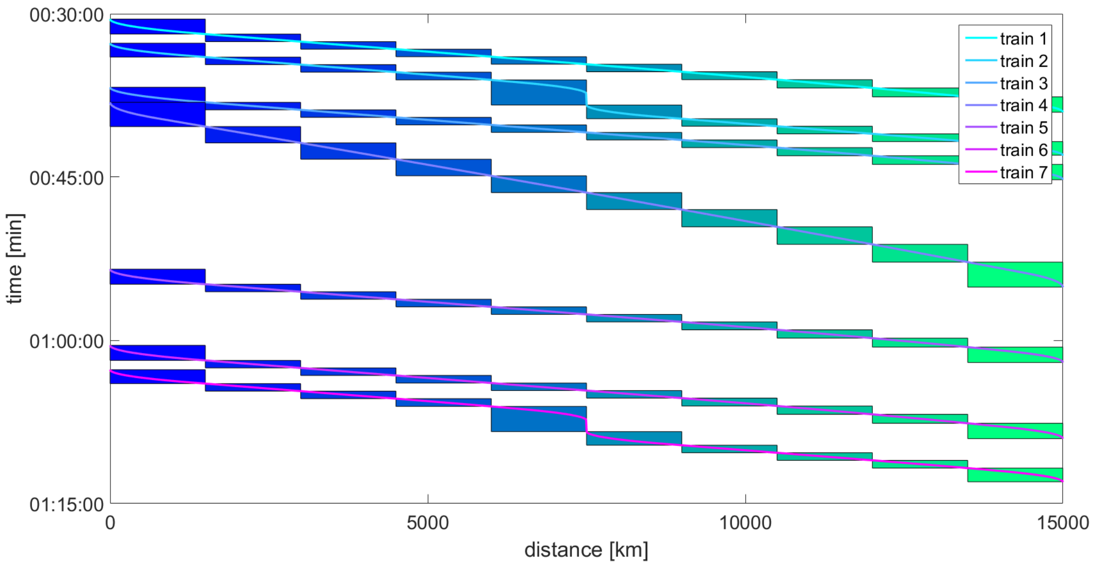


Figure 6-8: nominal train runs

Now, first the trajectories are shown in Figure 6-8 for the nominal schedule for the 7 trains that are later going to be rescheduled. This figure shows that the nominal schedule gives a feasible planning and the nominal timetable is thus valid. In Section 6-5-1, we obtained an optimal order of the 7 trains for the known delay on the track. This optimal order weights all trains with the same weight and therefore the total delay is minimized as much as possible for each train. The planning corresponding to the control decisions made is given by Figure 6-9

In Section 6-5-2, we obtained an optimal order of the trains with the delay introduced by the train on the track. The information of the new order and the delay on the track is sent to the trajectory planner where an optimal schedule is outputted back to the scheduler in order to validate the trajectories and check if the control decisions made earlier still hold.

The planning, that is obtained by the planner, at time instance $t = 24$, is shown in Figure 6-10. The weight of the objective functions are all equal so that the total delay is minimized in this figure.

It can be seen directly from the figure that the delay does effect the schedule of the all trains where it does have an effect on the first four. In contrast to the planning with the minimal delay the planning of Figure 6-10, it does have a larger total delay, but since the weights have

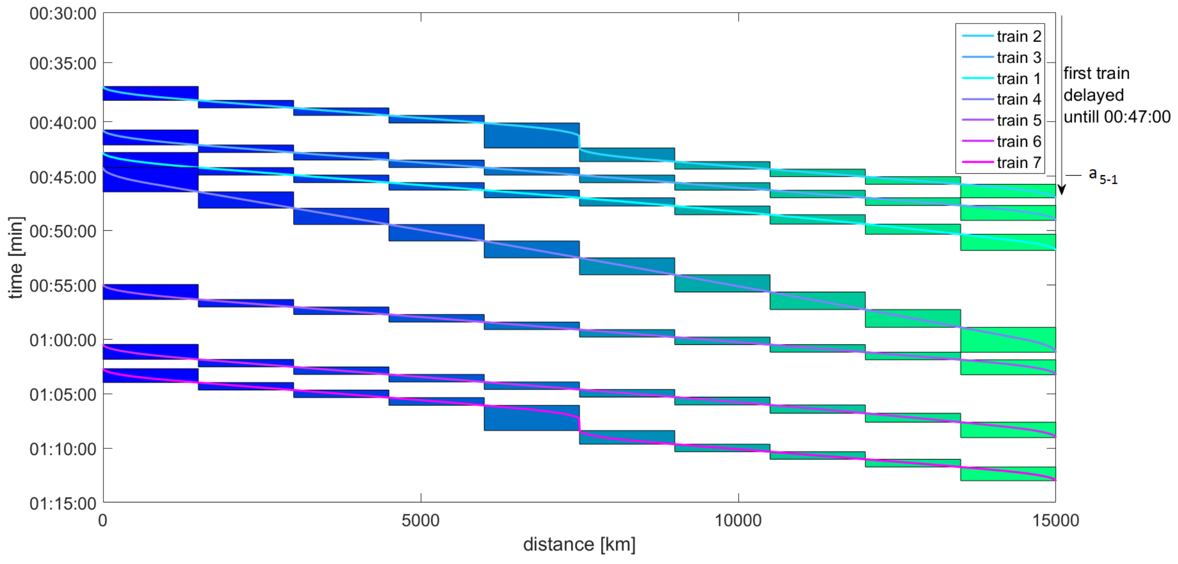


Figure 6-9: controlled time optimal planning for delayed trains

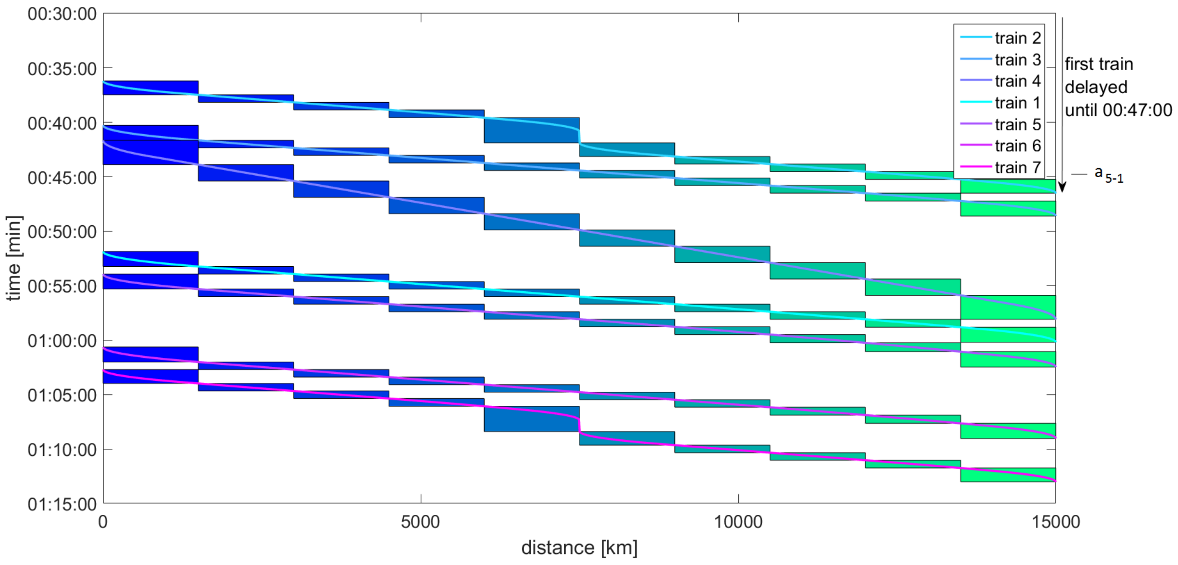


Figure 6-10: energy efficient planning for delayed trains

been adjusted towards energy saving, it has a lower energy consumption. It can be seen from the figure that the delay of the freight train is reduced by half with respect to the minimal delay optimization and is ordered now before train 1, so train 1 has been given a slightly larger delay since the freight train weighs more.

In comparison, we show the planning for the delayed train in the uncontrolled case. Since the nominal timetable of this part of the railway network is stable i.e. the eigenvalue of the system matrix is smaller than the periodic cycle time.

At the end of this chapter we can summarize the information streams needed for iterating to an optimal order and schedule.

From scheduler to planner	From planner to scheduler
- actual delays from abstraction - optimized order	- delay introduced by microscopic constraints

Table 6-6: information exchange

In Table 6-6, the actual delays mentioned are the delays that come either from known delays, such as the predicted extra waiting time for a train in the control horizon, or follow from the calculated minimum running time if this minimum running time indicates a delay in the main timetable.

The event times obtained by the planner are the actual optimal event times that take into account the microscopic network constraints. This could mean that an extra delay is introduced due to microscopic constraints and the current control decisions. This is implemented in the scheduler as an extra set of constraints as follows.
These extra constraint during iteration between the planner and scheduler have the following form:

$$r \geq \left(N_{ac} - \left(\sum_{i=1}^{N_{ac}} v_{n_{ac},i}\right)\right)\beta + \left(\sum_{j=1}^{N_{in}} v_{n_{in},j}\right)\beta + r + \text{extra}_{\text{delay}} \tag{6-39}$$

where $v \in \mathbb{R}^N$ and N is the number of control variables, N_{ac} is the number of control variables that are active and change the order of trains, let $v_{n_{ac}}\mathbb{R}^{N_{ac}}$ be a vector containing the indices of the control variables that are active with $v_{n_{ac},i}$ the ith element of vector $v_{n_{ac}}$, $N_{in} = N - N_{ac}$ is the number of inactive control variables and let $v_{n_{in}}\mathbb{R}^{N_{in}}$ be a vector containing the indices of the control variables that are inactive with $v_{n_{in},j}$ the jth element of vector $v_{n_{in}}$.
Let us consider the control vector v as in (6-38). Then, for every event time that has an extra delay due to microscopic constraints:

$$\begin{aligned} d &\geq (4 - (v_1 + v_2 + v_3))\beta + (v_4 + ... + v_{21})\beta + r_{\text{planner}} \\ a &\geq (4 - (v_1 + v_2 + v_3))\beta + (v_4 + ... + v_{21})\beta + r_{\text{planner}} \end{aligned} \tag{6-40}$$

So only the arrival and departure events that introduce an extra delay, with respect to the schedule obtained by the planner, are given an extra constraint as given in (6-39). In this way, the constraint is active only for the exact combination that was sent to the planner.

In this case study, we do not consider the extra delays introduced at the departure events because it was assumed that the departure station A has an infinitely large buffer for the trains to wait for their departure so an extra delay introduced at the departure events does not influence the connected railway network. Furthermore to simplify the implementation of the extra arrival constraints, it is chosen to only add only one arrival delay per iteration. The delay that introduces the largest mismatch between the scheduler and planner is picked for this constraint in this case study.

6-6-1 Iterations

An important part of the iterations are the stopping conditions. This is because no proof of convergence to the optimal solution is given in this thesis, but with this defined stopping conditions we are still able to tell something about the resulting schedule after that is send to the lower level controller of the trains. The stopping conditions are defined as, stop if:

- a previous set of control variables matches the current set.
- the total delay introduced by the scheduler for the current iteration is higher than the delay of the previous iteration.
- 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 the current found schedule is always a better overall schedule that the previous iteration. If not, the previous iteration is said to be most optimal in this set of iterations. This condition is weak in the sense that a better solution might be found after the solution with an increasing total delay. The reason for still using this condition is that the trajectory planner is far from optimized and computationally expensive. The last condition is obvious since we can control indirectly the computation time.

arrival events	iteration 1	iteration 2	iteration 3
train 1	60.2	48.6	57.1
train 2	46.5	46.5	46.5
train 3	48.6	60.2	59.4
train 4	58.1	58.1	55
train 5	62.5	62.5	62
train 6	69	69	69
train 7	73	73	73
ordering	2-3-4-1-5-6-7	2-1-4-3-5-6-7	2-4-1-3-5-6-7
total delay [min]	27.9	27.9	32.2

Table 6-7: arrival event times per iteration

The arrival events obtained by the planner for the energy optimal case are given in Table 6-7. The iterations are aborted during the third iteration after it comes clear that the total delay, obtained by the planner, increases. For the first two iterations, the total delay is exactly the same. As can be observed train 1 and train 3 are exchanged, but both these trains are of the same type and therefore their characteristics are exactly the same. From an engineering point of view, these two iterations cannot be distinguished, but it seems logical for passenger satisfaction to pick the ordering what looks most like the main timetable, and therefore iteration 2 is considered the most optimal schedule.

6-7 Summary

In this chapter, we showed that an optimal schedule can be obtained by first getting a prediction of the running times for all trains that are on the track and including these in the constraints of the MILP formulation. We then performed optimization on the upcoming trains with a MPC scheme. The outcome of this optimization was then used as a starting point for the planner and, via iterations between planner and scheduler, subsequently an optimal was schedule obtained.

Chapter 7

Conclusions and Future research

This chapter is a summary of the contributions made by this thesis. The main focus of this research is to abstract a hybrid dynamical system into a system where the dynamics were completely removed and only timing information is preserved. Hereafter, this timing information must be recasted into a Max-plus linear (MPL) system since max-plus algebra is especially suited for describing nonlinear timing behaviour of semi-cyclic systems in a linear fashion. Furthermore a feasible and especially optimal schedule must be obtained via optimization where the optimization framework makes use of the Switching max-plus linear (SMPL) model that is derived from the MPL model. The method of optimization chosen was Model predictive control (MPC) for its constraint handling. However due to the complexity of hybrid systems as a general class, it was decided that focus should be laid on a railway network for which a case study is elaborated on.

7-1 Conclusions

Abstraction of hybrid systems

- We have derived an abstraction method with sufficient conditions for a limited class of hybrid dynamical systems. This abstraction is then applied to a hybrid dynamical model of a train so that we are able to extract the timing information from any moment in time if the initial conditions for the trains on the track are known.
- The abstraction method is only valid for obtaining the minimal running times to complete the event. The hybrid dynamical model of a train is adjusted with information about the current speed limits for the block sections in which the track is divided for safety reasons. This hybrid model of trains running over a track is modeled in the case study as a Piecewise affine (PWA) model and because of the constraints set for the hybrid modes and switching points, the optimal i.e. minimal running times are automatically obtained.

Scheduling and planning a railway network

- The resulting minimal arrival times, for the trains on the track section that is considered for optimization, are compared to the scheduled arrival times so that, in case a delay happens, rescheduling actions can be applied. This provides an optimal schedule at regular intervals for which optimization is done.
- We showed how the schedule could be improved by adding extra constraints to the optimization of the scheduler that results from planning operations by the trajectory planner. The improvement of the schedule is done by iterating between scheduling and planning level until the control variables match a previous solution by the scheduler.

7-2 Future research

- The application in this research has been focused on railway management, but sequential interactive optimization can be applied to and can be beneficial for many more industrial processes as has already mentioned in Chapter 3. The hybrid models for these systems each have their own characteristics and the rules for abstraction may be revisited for each process.
- The trajectory planner is represented only with basic functions in this case study since it was not our main goal to find the most efficient trajectories but to find a framework that is suitable to finding an optimal schedule at regular times based on real-time information. The optimization that finds the trajectories is slow and the trajectories have a limited behaviour due to this basic implementation. For further research, it is advised to design a more sophisticated trajectory planner such as the one found in [21].
- This research has only focused on a very small part of the entire railway network, as shown in Chapter 6. At this point of the thesis, we cannot draw conclusions for the entire network based on only one sub-region of one of the many regions that form the total railway network. Therefore, this research should be extended so that an optimal schedule for all sub-regions can be obtained. It is known from the literature [24] that distributed optimization can efficiently handle large scale MPC optimization schemes.
- As is already mentioned, the optimized schedule found after iterations is not necessarily the global optimum. It would be interesting to show under what conditions, a proof of convergence could be derived.
- Finally, the hybrid model used for abstraction makes use of a linear model. The error introduced by this linearization is not further researched, but it is interesting to show how this error influences rescheduling operations on the timetable and the effect on the performance of the real railway network. Instead, another approach is to apply advanced nonlinear dynamics to the hybrid model of a train and abstract that model to obtain more accurate minimal running times.

Appendix A

Matlab program case study

Overview

The matlab file **case_study.m** is the main file from where all functions are called by there function syntax. This main file is structured as follows:

---

case_study.m

- Adjustable_Variables.m ← (require external inputs)
- Index_Matrices.m
- Abstraction.m
- Timetable.m ← (require external inputs)
- Constraints.m
- Constr_Pos_Delay.m
- Iterations.m
  - intlinprog.m (function of the matlab Optimization Toolbox™)
  - Init_opti.m
    - Nonlinear_opt.m
      - fmincon.m (function of the matlab Optimization Toolbox™)
        - fun.m
        - nonlcon.m
- Plot_Rescheduled_Train_Runs.m

---

every function will now be explained

Adjustable_Variables

The main variables for the trains used in the case study are defined in this file.

synopsis

[beta, number_of_trains, tpc,trot,type, t_opt, weights, number_of_iterations, tau_r, headway_0d, headway_0a] = Adjustable_Variables();

description

The outputs in this function are loaded to the workspace and used in the subsequent functions. This function does not require input arguments.

beta	Large negative value for activating or deactivating the binary variables
number_of_trains	number of trains considered in the control horizon
tpc	trains per cycle, where a cycle is the periodicity of the timetable
trot	trains on the track which cannot be rescheduled
type	native order of trains in the main timetable sorted per type
t_opt	the time the optimization is started
weights	weights for the objective function in the MPC problem
number_of_iterations	maximum number of iterations between the scheduling and planning level
tau_r	minimum running times between the end and begin station
headway_0d	departure headway matrix
headway_0a	arrival headway matrix

Index_Matrices

This function finds indexing variables used in the constraints.

synopsis

[U_index, UUt] = Index_Matrices(number_of_trains);

description

Given the number of trains, the switching variables are assigned to the headway's between the trains. Furthermore a matrix is build for indexing correct signs for the constraints.

U_index	index matrix for building the MILP constraints
UUt	matrix keeps track of correct + and - signs in the MILP constraint matrix
number_of_trains	number of trains in the control horizon

Abstraction

Finding the minimal running times.

synopsis

[L] = Abstraction(type, tpc, trot);

description

This function abstract the train dynamics of the trains on the track into minimal running times to reach the end station. The output is used extend the arrival times of the trains on the track with this real time information.

L	minimum running times for the trains on the track to reach the end station
type	native order of all trains expressed in train type
tpc	number of trains per cycle
trot	number of trains that are actually on the track

Timetable

Timetable adjusted with real time information by abstraction.

synopsis

[r_main, r_on_track_main, r, r_abstr_on_track] = Timetable(t_opt, L, tpc, trot, headway_0a, beta, (r_abstr_on_track));

description

This function outputs the nominal timetable enriched with real time track information. Optional is r_abstr_on_track as an input. In a distributed network, this is an input from another optimized part of the railway network. In this case study, this input is omitted and fixed in the function itself.

r_main	main timetable for trains to be considered for rescheduling
r_on_track_main	main timetable for the trains on the track
r	timetable for trains to be rescheduled
r_abstr_on_track	resulting timetable for trains on the track
t_opt	time for which the optimization starts
L	minimum running times for the trains on the track to reach the end station
tpc	number of trains per cycle of the nominal schedule
trot	number of trains on the track
headway_0a	headway matrix used to separate arrival events of the trains on the track
beta	large negative number
r_abstr	possible timetable adjustment for trains to be rescheduled (optional)

Constraints

Building the constraints for reordering

synopsis

[A1, V1, p1, p2, q1, s2] = Constraints(number_of_trains, UUt, U_index, beta, headway_0d, headway_0a, r, tau_r);

description

This function builds the constraints for the MILP problem. The constraints are written as $Axx \leq V$.

A1	constraints ordering matrix, left side inequality for controllable trains
V1	constraints ordering vector, right side inequality for controllable trains
p1	number of departure constraints
p2	number of arrival constraints
q1	number of switching variables
s2	number of optimization variables
number_of_trains	number of trains for optimizations
UUt	matrix keeps track of correct + and - signs in the MILP constraint matrix
U_index	index matrix for building the MILP constraints
beta	large negative number for modeling max-plus binary variables
headway_0d	departure headway matrix for trains to be rescheduled
headway_0a	arrival headway matrix for trains to be rescheduled
r	timetable for trains to be rescheduled
tau_r	minimum running times for trains to be rescheduled

Constr_Pos_Delay

Adding constraints that result from the abstraction.

synopsis

[A, V, sa2] = Constr_Pos_Delay(number_of_trains, tau_r, trot, r_abstr_on_track, A1, V1, p1, p2, q1, s2);

description

Extra constraints are introduced that adds a possible delay scenario based on the outcome of the abstraction.

A	extended ordering constraints matrix, left side inequality
V	extended ordering constraints vector, right side inequality
sa2	size of objective function MPC
number_of_trains	number of trains considered for rescheduling
tau_r	minimum running times for trains to be rescheduled
trot	number of trains that are actually on the track and cannot be rescheduled
r_abstr_on_track	actual arrival times for trains on track
A1	constraints ordering matrix, left side inequality for controllable trains
V1	constraints ordering vector, right side inequality for controllable trains
p1	number of departure constraints
p2	number of arrival constraints
q1	number of switching variables
s2	number of optimization variables

Iterations

Iterate optimnized schedule between dynamic scheduler an trajectory planner.

synopsis

[v, x_ideal, times_ideal, times_all, I_ideal, pblock, T_ideal, Y_ideal] = Iterations(number_of_iterations, weights, number_of_trains, trot, A, V, sa2, s2, r, r_main, type, beta);

description

This function generates a loop over the interaction between the scheduler and trajectory planner. For each iteration, the optimized schedule and control variables are stored in in the workspace.

cpid	matrix containing optimized coasting point and departure time
v	stacked optimal control vector for every iteration
x_ideal	optimized event times with the corresponding control vector
times_ideal	optimized departure and arrival times
times_all	optimized departure and arrival times for every iteration
I_ideal	optimized ordering of trains
pblock	vector containing blocking length information
T_ideal	time vector per train result from simulation
Y_ideal	distance vector per train result from simulation
number_of_iterations	maximum number of iterations allowed
weights	weights for the objective function of the MPC problem
number_of_trains	number of trains in the control horizon
trot	number of trains on the track which cannot be rescheduled anymore
A	extended ordering constraints matrix, left side inequality
V	extended ordering constraints vector, right side inequality
sa2	size of objective function MPC
s2	number of optimization variables
r	timetable for trains to be rescheduled
r_main	nominal schedule for the non delayed network
type	native order of trains in the main timetable sorted per type
beta	large negative number for modeling max-plus binary variables

intlinprog

Solving the MPC problem.

synopsis

[xx, Fval] = intlinprog(w_tot, index_int_con, A, V, [], [], lower_b, upper_b)

description

This function solves the MPC optimization problem by a MILP formulation. The result is an optimal ordering of the trains with their corresponding departure and arrival times.

xx	optimal control vector containing event times and switching variables
Fval	value of the objective function
w_tot	extended weight vector
index_int_con	vector containing the indexes of the integer variables
A	matrix left side inequality
V	vector right side inequality
lower_b	lower bound of the control variables
upper_b	upper bound of the control variables

Init_opti

Initialization of the nonlinear optimization.

synopsis

[cpid, times, pblock, T_all, Y_all] = Init_opti(I, r_sched, type, number_of_trains);

description

This function collects the optimal ordering and minimal delay for the first train from the scheduler and in this function are the track characteristics defined such as track length, block lengths and dwell time.

cpid	matrix containing optimized coasting point and departure time
times	all event times for trains in the control horizon
pblock	vector containing blocking length information
T_ideal	time vector per train result from simulation
Y_ideal	distance vector per train result from simulation
I	optimized ordering of trains
r_sched	timetable extended with delay information due to the abstraction
type	assign train type to the train order
number_of_trains	number of trains in the control horizon

Nonlinear_opt

Trajectory planner.

synopsis

[cpid, times, delay, tau, headway_a, headway_d, T_all, Y_all] = Nonlinear_opt(schedule, pblock, s_end, number_of_trains, type, s_interm_station, dwell_time, order_of_trains);

description

This function finds the trajectories for the trains via a nonlinear optimization function fmincon.m and includes microscopic constraints such headway constraints described by blocking times.

cpid	matrix containing optimized coasting point and departure time
times	all event times for trains in the control horizon
delay	arrival delay vector
tau	actual running time according to the planner
headway_a	arrival headway matrix
headway_d	departure headway matrix
T_all	time vectors for simulation of all trains ans all iterations
Y_all	distance vectors for simulation of all trains and all iterations
schedule	matrix of departure and arrival events according to delays and timetable
pblock	vector containing blocking length information
s_end	length of the track section
number_of_trains	number of trains in the control horizon
type	assign train type to the train order
s_interm_station	location of the extra station for stop trains
dwell_time	passenger transfer time for stop trains
order_of_trains	order of trains according to the main timetable

fmincon

Nonlinear optimization function, Matlab Toolbox™.

synopsis

[x, fval, exitflag, output] = fmincon(@(x)fun(x, times(n,:), type, s_end, n, s_interm_station, order_of_trains, schedule, v_max, a_acc, a_des, a_co),x0,a_ineq_fmin,b_ineq_fmin, [], [], l_b_fmin,u_b_fmin,[1e5 1e5], @(x)nonlcon(x, times(n,:), pblock, type, s_end, n, s_interm_station, dwell_time, order_of_trains,v_max, a_acc, a_des, a_co), options);

description

This function minimizes the objective function such that the nonlinear constraints are satisfied.

x	vector containing departure time and start of coasting per train
fval	objective function value
exitflag	integer for evaluating the found optimum
output	summary of optimization algorithm outcome
x0	initial point of x that satisfies all the constraints
a_ineq_fmin	left side inequality constraints on x
b_ineq_fmin	right side inequality constraints on x
l_b_fmin	lower bounds on x
u_b_fmin	upper bounds on x

fun

Objective function of fmincon.m.

synopsis

[Opt_traj, dOpt_traj] = fun(x, times, type, s_end, n, s_interm_station, order_of_trains, schedule, v_max, a_acc, a_des, a_co);

description

This function contains the objective for the trajectory optimization. For this function is given that a trajectory is given by $f(cp, id)$ that is, an optimal trajectory is a function of *cp*, which is the time interval when a train starts to coast from the begin station and *id* which is the departure time of that train.

Opt_traj	
dOpt_traj	
x	vector containing departure time and start of coasting per train
times	all event times for trains in the control horizon
type	assign train type to the train ordering
s_end	length of the track
n	train number for which a trajectory is planned
s_interm_station	location of the extra station for stop trains
order_of_trains	order of trains according to the main timetable
schedule	matrix of departure and arrival events according to delays and timetable
v_max	speed limit for every train
a_acc	maximum acceleration of a train
a_dec	maximum deceleration of a train due to breaking
a_co	deceleration of a train due to coasting

nonlcon

Nonlinear constraints of fmincon.m.

synopsis

[c, ceq, dc, dceq] = nonlcon(x, times, pblock, type, s_end, n, s_interm_station, dwell_time, order_of_trains, v_max, a_acc, a_des, a_co);

description

The nonlinear constraints are build where a trajectory is checked for each block section if a safe distance is maintained. A safe distance is one or two block sections depending of the train type.

c	inequality constraint
ceq	equality constraint
dc	derivative of the inequality constraint
dceq	derivative of the equality constraint
x	vector containing departure time and start of coasting per train
times	all event times for trains in the control horizon
pblock	vector containing blocking length information
type	assign train type to the train ordering
s_end	length of the track
n	train number for which a trajectory is planned
s_interm_station	location of the extra station for stop trains
dwell_time	passenger transfer time for stop trains
order_of_trains	order of trains according to the main timetable
v_max	speed limit for every train
a_acc	maximum acceleration of a train
a_des	maximum deceleration of a train due to breaking
a_co	deceleration of a train due to coasting

Plot_Rescheduled_Train_Runs

Plot the results of the iterated schedule.

synopsis

Plot_Rescheduled_Train_Runs(times_ideal, number_of_trains, pblock, I_ideal, T_ideal, Y_ideal);

description

This function requires only input arguments from Adjustable_Variables.m and Iterations.m to show the results of the optimized schedule found by the interactive framework.

times_ideal	optimized timetable for the trains in the control horizon
number_of_trains	number of trains in the control horizon
pblock	vector containing blocking length information
I_ideal	optimized ordering of trains
T_ideal	time vector per train result from simulation
Y_ideal	distance vector per train result from simulation

Bibliography

[1] L. Özkan, T. van den Boom, and J. Ludlage, "Real time scheduling and model based control for improved process operation." An idea for STW Open Call, 2015.

[2] A. J. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*. No. 251 in Lecture Notes in Control and Information Sciences, Springer-Verlag, 2000.

[3] P. A. Fishwick, ed., *Handbook of Dynamic System Modeling*, ch. 15. Chapman and Hall/CRC, 2007.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "Hybrid systems the algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[5] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[6] E. D. Sontag, "Nonlinear regulation: the piecewise linear approach," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 346–357, 1981.

[7] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[8] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat, *Synchronization and linearity: an algebra for discrete event systems*. John Wiley & Sons Ltd, 1992.

[9] B. Heidergott, G. J. Olsder, and J. van der Woude, *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press, 2006.

[10] T. 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, pp. 1199–1211, oct 2006.

[11] J. Lygeros, "Lecture notes on hybrid systems," 2004.

[12] J. L. Piovesan, H. G. Tanner, and C. T. Abdallah, "Discrete asymptotic abstractions of hybrid systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 917–922, 2006.

[13] H. K. Khalil, *Nonlinear Systems.* Prentice Hall, second ed., 1996.

[14] E. Asarin, V. P. Mysore, A. Pnueli, and G. Schneider, "Low dimensional hybrid systems - decidable, undecidable, don′t know," *Information and Computation*, vol. 211, pp. 138–159, 2012.

[15] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?," *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998.

[16] H. G. Tanner, J. L. Piovesan, and C. T. Abdallah, "Finite asymptotic abstractions for hybrid systems with stable continuous dynamics." submitted, 2009.

[17] V. I. Zubov, *Mathematical Methods for the Study of Automatic Control Systems.* Pergamon Press/Macmillan, 1963.

[18] M. Alirezaei, T. van den Boom, and R. Babuska, "Max-plus algebra for optimal scheduling of multiple sheets in a printer," in *2012 American Control Conference (ACC)*, pp. 1973–1978, June 2012.

[19] I. A. Hansen and J. Pachl, eds., *Railway Timetable & Traffic.* Hamburg: Eurailpress, 2008.

[20] R. Liu and I. M. Golovicher, "Energy-efficient operation of rail vehicles," *Transportation Research Part A: Policy and Practice*, vol. 37, no. 10, pp. 917–932, 2003.

[21] Y. Wang, *Optimal Trajectory Planning and Train Scheduling for Railway Systems.* PhD thesis, Delft University of Technology, Delft, The Netherlands, November 2014.

[22] N. Besinovic, E. Quaglietta, and R. M. P. Goverde, "A simulation-based optimization approach for the calibration of dynamic train speed profiles," *Journal of Rail Transport Planning & Management*, vol. 3, pp. 126–136, 2013.

[23] J. Pachl, *Railway operation and control.* Mountlake Terrace (USA): Vtd Rail Pub, januari 2002.

[24] B. Kersbergen, *Modeling and Control of Switching Max-plus-linear Systems: Rescheduling of Railway Traffic and Changing Gaits in Legged Locomotion.* PhD thesis, Delft University of Technology, Delft, The Netherlands, October 2015.

[25] J. G. Braker, "Max-algebra modelling and analysis of time-dependent transportation networks," in *Proceedings of the 1st European Control Conference*, (Grenoble, France), pp. 1831–1836, july 1991.

[26] T. van den Boom and B. De Schutter, "Modeling and control of switching max-plus-linear systems with random and deterministic switching," *Discrete Event Dynamic Systems*, vol. 22, no. 3, pp. 293–332, 2012.

[27] T. van den Boom, N. Weiss, W. Leune, R. M. P. Goverde, and B. De Schutter, "A permutation-based algorithm to optimally reschedule trains in a railway traffic network," in *Proceedings of the 18th IFAC World Congress*, (Milan, Italy), pp. 9537–9542, Aug-Sept 2011.

[28] A. Schrijver, *Theory of Linear and Integer Programming.* Chichester UK: John Wiley & Sons Ltd, 1986.

[29] V. A. Profillidis, *Railway management and engineering.* Ashgate, 4th ed., 2014.

Glossary

List of Acronyms

SMPL	Switching max-plus linear
MPL	Max-plus linear
PWA	Piecewise affine
MLD	Mixed-logical dynamical
MPC	Model predictive control
DES	Discrete-event system
MILP	Mixed-integer linear programming

List of Symbols

$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{\max}$	Set of $\mathbb{R} \cup \{-\infty\}$
$\tau_{\max}$	Minimum process time
$\tau_{\max}$	Nominal process time
ξ	State variable for the continuous dynamics
G	Guard set of a hybrid automaton
I	Intervals of the hybrid process time τ
Q	Set of modes of a hybrid automaton
q	Hybrid mode
r	Schedule as reference for the hybrid system
x	State variables for the discrete-event system
Inv	Invariant set of a hybrid automaton