FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Energy-efficient Optimization of Railway Operation

## An Algorithm Based on Kronecker Algebra

DISSERTATION

zur Erlangung des akademischen Grades

**Doktor der technischen Wissenschaften**

eingereicht von

**Dipl.-Ing. Mark Volcic, BSc**
Matrikelnummer 0425294

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Ao. Univ. Prof. Dipl.-Ing. Dr.techn. Johann Blieberger
Priv. Doz. Dipl.-Ing. Dr.techn. Andreas Schöbel

Diese Dissertation haben begutachtet:

(Ao. Univ. Prof. Dipl.-Ing. Dr.techn. Johann Blieberger)

(Prof. Dr.-Ing. habil. Jürgen Siegmann)

Wien, 29.12.2014

(Dipl.-Ing. Mark Volcic, BSc)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Energy-efficient Optimization of Railway Operation

## An Algorithm Based on Kronecker Algebra

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

**Doktor der technischen Wissenschaften**

by

**Dipl.-Ing. Mark Volcic, BSc**
Registration Number 0425294

to the Faculty of Informatics
at the Vienna University of Technology

Advisor:
Ao. Univ. Prof. Dipl.-Ing. Dr.techn. Johann Blieberger
Priv. Doz. Dipl.-Ing. Dr.techn. Andreas Schöbel

The dissertation has been reviewed by:

(Ao. Univ. Prof. Dipl.-Ing. Dr.techn. Johann Blieberger)

(Prof. Dr.-Ing. habil. Jürgen Siegmann)

Wien, 29.12.2014

(Dipl.-Ing. Mark Volcic, BSc)

# Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Mark Volcic, BSc
Erich-Fried-Weg 1/6/606, 1220 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

# Acknowledgments

I want to thank Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Johann Blieberger and Priv.Doz. Dipl.-Ing. Dr.techn. Andreas Schöbel from Vienna University of Technology for their great support in the research project EcoRailNet and for supporting my PhD-thesis. Next, I want to thank Prof. Dr.-Ing. habil. Jürgen Siegmann from the Technische Universität Berlin for his attendance to do the assessment of my thesis.

Furthermore, I want to thank the Austrian Ministry of Transport, Innovation and Technology (bmvit) and the Austrian Research Promotion Agency (FFG) for sponsoring the project EcoRail-Net in the frame of the program New Energy 2020 (Project-ID: 834586), which was the basis for this thesis.

Last but not least, I want to say thank you to my family for the patience and the great support in my educational, occupational, and private way.

# Abstract

In general, a railway system consists of several trains which use a large infrastructure. In such complex systems it is very difficult for the train driver and the dispatcher to find a solution where each train will arrive at its destination in time and the energy consumption is minimal. To ensure the requirement of a safe system, few operations are done by automated systems. Often there is less consideration on the energy consumption resulting in a high potential for saving traction energy. This potential can be divided into two parts. On one hand each single trip can be optimized by using an optimal driving strategy, where the demand of traction energy is minimal. On the other hand, the complete railway system is analyzed. In the latter case the schedule, delays, and restriction on the tracks are taken into account for the calculation of an optimal solution. The optimization of such a system must satisfy important criteria, such as safety, punctuality and passenger comfort. Based on these restrictions, the optimization algorithm determines an overall solution where the energy consumption is as small as possible. The results of the algorithm can be used to support train drivers and dispatchers.

The first part of the algorithm consists of finding an optimal solution for a single journey, which consists of four driving types, namely *acceleration-phase*, *speed-hold-phase*, *coasting-phase*, and *braking-phase*. The second part of the algorithm is the optimization of the complete system by applying the so-called Kronecker Algebra and the single-trip optimization.

Kronecker Algebra consists of Kronecker Sum and Kronecker Product and is used to create a matrix, which contains all possible movements of the trains within the system. Each train has a route for its journey which consists of a sequence of track sections. This information is given as matrix and used as input for the algorithm. Each track section is modeled by a semaphore and given as a matrix, too. Kronecker Sum calculates all possible interleavings of the trains, but only if they do not use the same track sections. If several trains use a common track section, Kronecker Product is used to ensure that the access on each section is done sequentially. In particular a train can enter a section only if it has been released by another train. The result of the application of Kronecker Algebra is again a matrix, which can be represented by a graph. The optimal driving strategy for each train is given as a distance-speed-diagram.

This model is used to determine and to avoid conflicts (e.g. deadlocks or headway-conflicts). All conflict-free solutions are calculated and as a result, the best solution in terms of energy consumption is given.

# Kurzfassung

Ein Eisenbahnnetz besteht im Allgemeinen aus einer umfangreichen Infrastruktur, auf der eine Vielzahl an Zügen verkehrt. In einem derart komplexen System ist es für den einzelnen Triebfahrzeugführer oder Disponenten schwierig, die jeweils richtige Entscheidung für den optimalen Betriebsablauf zu treffen. Zur Gewährleistung der Sicherheit wird ein Großteil des Betriebs automatisiert durchgeführt. Allerdings wird in Dispositionsentscheidungen der Energiebedarf nicht oder nur am Rande berücksichtigt, wodurch sich ein großes Potential zur Einsparung ergibt, das auf zwei Ebenen aufgeteilt werden kann. Zum Einen kann jede einzelne Zugfahrt optimiert werden, um das Fahrziel mit möglichst wenig Energiebedarf zu erreichen. Zum Anderen muss das Gesamtsystem betrachtet werden und aufgrund des aktuellen Fahrplans, etwaiger Verspätungen oder auch aufgrund von gesperrten oder nur eingeschränkt nutzbaren Abschnitten eine optimale Strategie zur Minimierung des Energiebedarfs entwickelt werden. Diese beiden Ebenen zur Optimierung müssen sich allerdings wichtigen Kriterien, wie zum Beispiel der Sicherheit, der Pünktlichkeit und auch dem Fahrkomfort unterordnen. Basierend auf diesen Anforderungen berechnet der Optimierungsalgorithmus eine Gesamtlösung mit möglichst geringem Energiebedarf. Dem Disponenten bzw. dem Triebfahrzeugführer können somit Informationen zum optimierten Gesamtsystem bzw. zur optimalen Einzelfahrt zur Verfügung gestellt werden.

Der erste Teil des Algorithmus besteht aus der Optimierung der Einzelfahrten, dessen Ergebnis sich aus den Fahrtypen *Beschleunigung*, *Halten der Geschwindigkeit*, *Ausrollen* und *Bremsen* zusammensetzt. Der zweite Teil optimiert das Gesamtsystem, bei der die sogenannte Kronecker-Algebra und die Einzelfahrt-Optimierung zur Anwendung kommen.

Die Kronecker-Algebra besteht aus der Kronecker-Summe und dem Kronecker-Produkt und dient dazu, eine Matrix zu erzeugen, die alle Zugbewegungen im System beinhaltet. Für die Berechnung werden die Routen der Züge als Abfolge von Streckenabschnitten in einer quadratischen Matrix dargestellt. Die Abschnitte selbst werden als Semaphore modelliert und ebenfalls in Matrizen bereitgestellt. Die Kronecker-Summe berechnet alle möglichen Verschachtelungen der beteiligten Züge, sofern diese keinen Streckenabschnitt gemeinsam nutzen. Andernfalls können über das Kronecker-Produkt alle Varianten berechnet werden, in denen mehrere Züge einen gemeinsamen Abschnitt befahren. In diesem Fall wird sichergestellt, dass der Zugriff sequentiell erfolgt. Das bedeutet, dass ein Zug einen Abschnitt erst dann befahren darf, wenn er von einem anderen Zug freigegeben wurde. Die Anwendung dieser mathematischen Operationen erzeugt als Ergebnis wiederum Matrizen, die zur Visualisierung der Ergebnisse in Form von Graphen dargestellt werden können. Die optimale Fahrkurve der einzelnen Züge wird als Weg-Geschwindigkeit-Diagramm zur Verfügung gestellt.

Die Verwendung dieses mathematischen Modells erlaubt es, Konflikte (z.B. Deadlocks, Auflaufen) zu erkennen und diese nicht mehr als potentielle Lösungen zur Verfügung zu stellen. Für die verbleibenden Zugbewegungen wird mittels der zuvor erwähnten Einzelfahrtoptimierung der Energiebedarf errechnet und anschließend die optimale Strategie zur Verfügung gestellt.

# Contents

# List of Figures

# List of Tables

CHAPTER 1

# Introduction

For each railway company it is a formidable challenge to provide a well-performing railway network in several aspects like punctuality, passenger comfort or energy demand. Due to the great amount of trains, track sections, railway station, and so on, humans cannot overview and handle such complex systems. Therefore automation systems may adopt some tasks and help the user to find good strategies and act in a safe way. One of such challenges is to find optimal driving strategies for each train within the railway system. If the train driver knows the track exactly, he might find a *good* driving strategy in terms of energy demand, but it may not be the best one. Algorithms on fast computers can calculate an optimal driving strategy very fast and this information can be given to the train driver or to an automated driving system. For large systems, where several trains use the same track sections, the problem is more complex, because there must be a synchronization between the trains to guarantee that a train enters a track section only, if it is not occupied by another train. The sequence of entering and leaving track sections depends on the planned travel time and the energy consumption when finding a solution with minimal energy consumption. Finding the best solution is a complex task and needs a lot of computing power. Thus it is impossible for humans to find the best strategy in real-time without the support of automation systems.

## 1.1 Prerequisites to the algorithm

This thesis describes an algorithm to calculate the optimal driving strategy in terms of energy demand for several trains within a railway network. The development of the algorithm was done within the so called *EcoRailNet*-project, an Austrian research project. The project partners were *Österreichische Bundesbahnen (ÖBB)*, *THALES Austria*, and *Vienna University of Technology* (Institute of Computer Aided Automation). Based on several practical restrictions from ÖBB, an algorithm was designed and implemented to optimize a railway system. These restrictions are:

- *Safety*: The algorithm must be safe in a way that not more than one train can be within a track section. Further track restrictions (e.g. maximum allowed track speed) and train restrictions (e.g. maximum train speed) must be taken into account.

- *Punctuality*: The trains must leave and arrive in time.

- *Driving comfort*: The calculated speed profile must be easy to follow by the train driver and the passenger comfort must not be affected in an unpleasant way (e.g. lots of accelerating and braking phases within one trip). In addition, it must be possible to define a speed-step-size (e.g. 5-km/h-steps) due to the restrictions of the cruise control.

- *Energy efficient driving*: The overall energy consumption of all trains must be as small as possible in strict accordance of the previous restrictions.

## 1.2 Outline and state-of-the-art

Based on these restrictions an algorithm was developed. Basically, it is divided into two parts:

1. Single trip optimization: This part of the algorithm calculates the optimal driving strategy for a single train on a track with no influence on other trains. Several train and track parameters are included in the calculation of the driving strategy.

   Brüger *et al.* [6] give an overview of the physical train model and running time estimation, including methods for solving the corresponding equations. The model presented in Chapter 3 does not solve differential equation in this way – the optimal driving strategy is determined by a meta model to find the optimal driving speed and types.

   Albrecht [2] introduces methods for finding the optimal driving strategies for a train, consisting of an acceleration phase, a cruising phase, coasting and a braking phase but only for tracks with constant inclination. Chapter 3 will show methods for arbitrary track inclination.

   Howlett *et al.* explain techniques for finding optimal driving strategies in several scientific papers (cf. [1,13,32–35]). These methods will create speed profiles with several variations of driving types (cf. Chapter 3) within a single trip and thus, they are not feasible for the demand of a comfortable driving strategy for the driver and the passengers.

   In [29] and [49] an algorithm for determining the optimal driving strategy is presented, too, again with several variations of driving types. In addition, the demand of the speed-step-size (cf. Section 3.2) cannot be satisfied.

   The same is valid for [23], [24], and [41] which present algorithms under the assumption that the train is handled as a mathematical point with unit mass.

   In [59], three different methods for finding energy-efficient driving strategies are presented. Summing up it can be said that each method has at least one disadvantage (e.g. calculation time, ease of implementation).

Desprez and Djellab [18] present an algorithm which is similar to the algorithm in Chapter 3 but not practical due to the lack of the possibility to choose the speed-step-size and the minimum travel time for each driving type (cf. Sections 3.2 and 3.4).

Schöbel *et al.* [72] present a method for reducing the energy consumption by optimizing the station dwell time. Due to the reduced hold speed the energy consumption can be reduced by approximately 1 %. This method does not consider a coasting phase which brings another potential for energy savings.

Summing up it can be said that several optimization algorithms do not consider a number of practical aspects like speed-step-size, minimal travel time for each driving type, track inclination, real train length and mass, or the computation time (cf. [4, 8, 14, 21, 47, 48, 82, 84]). Thus, the algorithm presented in Chapter 3 is the first for practical applications due to its low computation time and the number of parameters for practical usage.

2. System optimization: The second part of the algorithm is based on the so-called Kronecker Algebra. The application of the mathematical model calculates all possible movements of all trains within the railway system and creates matrices as results which can be represented by graphs. This graph can be reduced to the relevant nodes, so called synchronizing nodes which are used to guarantee that trains can enter a track section only if it is not occupied by another train. Based on the physical train model, the energy consumption of all possible movements of the trains is calculated and the best result is determined.

   In [53,60,61,70,71] the potential for energy savings by using computer-based train control is shown. The presented methods try to avoid conflicts and thus, additional stops within the trip (cf. [51]). In addition, the potential energy savings in the *Lötschberg Base Tunnel* are given. A similar approach is used in the system optimization (Chapter 5).

   In [39] some methods for conflict resolutions (*Dispatching Based on Asynchronous Simulation* and *Dispatching Based on Synchronous Simulation*) are explained, which may be possible for small railway networks but they do not scale up. In addition, extended stops, extra stops, and increasing running time are mentioned. The optimization algorithm in Chapter 5 does not consider these possibilities, because it tries to find a solution without changing the dispatching parameters.

   The method in [69] uses graph theory to model the railway infrastructure on a microscopic and macroscopic level. A similar model is used in Chapter 4.

   Siefer [73] gives an overview of simulations and explains the advantages of this method in railway systems. In contrast to common simulation tools, the model in this thesis calculates all possible routes and chooses the best one in terms of safety, punctuality, and energy consumption.

   In [16] *Banker's algorithm* [20] has been modified such that it can be used for deadlock analysis in railway systems. Due to the fact that the native algorithm was designed for computer systems, it is not well-suited for the application in railway systems. It might be the case that the usage of a track section is prohibited although it can be used without the danger of deadlocks. In contrast to the model in Chapter 4, [16] models both track sections and switches as resources.

Zarnay [83] uses a colored Petri net model and Banker's algorithm to avoid deadlocks. Due to the previously mentioned restrictions of Banker's algorithm, it turned out that it cannot handle a large number of trains and track sections.

In [22] a colored Petri net model is explained to guarantee safeness and deadlock-freedom but it is not clear how this approach can be extended to find optimal solutions.

In [52] an operations research approach is presented to determine deadlocks in railway systems.

Pachl [63,65] shows two methods for deadlock analysis which deliver false positives. The same restriction applies to the approach presented in [50], [55], and [66].

Fuchsberger [25] presents the train routing problem and the pulsed train scheduling problem in a main station area but it is unclear how the model performs for regions with less tracks and switches or if the model is scalable for larger areas.

Several scientific papers from Switzerland (cf. [9–12]) present new approaches to generate conflict-free train schedules. The basic idea is that the railway network is divided into so-called compensation zones and condensation zones. This segmentation is done due to the assumption that there is a high traffic density at railway stations and a low traffic density between them. Around the railway stations, predefined speed profiles are used to ensure punctuality. Between these nodes optimized speed profiles are generated to reduce the energy consumption. As a result the complexity of the system could be reduced and the driving strategy must not be calculated for the complete railway system. Major disadvantages of the model are that the calculated speed profiles could be more realistic, and that some travel time might be lost due to the predefined profiles. The model was not tested for larger railway networks. Other approaches use conflict-graphs, precomputed blocking-stairways, or a framework to generate periodic timetables respectively.

Jaekel and Albrecht [40] present a model based on non-linear programming for train path envelope determination, including a case study from Sweden. The optimization was done in MATLAB with a predefined toolbox but the calculation time was too high for the usage in real-time systems on online-calculations within train operation.

## 1.3 Contents of this thesis

The thesis is divided into the following parts:

- Chapter 2 describes the train model. In particular, several formulas and a description of the traction force, the braking force, and the resistance values are given. At the end of the chapter, some important details about the implementation of the train model are shown. The model is used for the optimization strategies, presented in the succeeding chapters.

- Chapter 3 describes the single trip optimization. The driving strategy of a single train is calculated, based on the schedule of the train, the given track, and the physical train model, including several configuration parameters.

- Chapter 4 gives an overview and some theoretical background on Kronecker Algebra, a mathematical model using matrices to calculate all possible movements of the trains within a railway network, including feasible deadlocks.
- Chapter 5 describes the Kronecker Algebra based system optimization, including the reduction of the resulting graph, the calculation of all possible routes of the trains and the determination of the optimal driving strategy of each involved train.
- Chapter 6 shows some extensions of the model.
- Chapter 7 contains the conclusion of the work and a prospect.

To illustrate the theoretical elaboration, several examples are given in the following chapters.

The implementation of the train model, the operations of Kronecker Algebra, and the algorithms to calculate the optimal driving strategy for a single train and to determine the optimal solution for the overall system is done in *Ada*. This programming language offers a number of features for complex implementations and real-time systems, e.g. object-oriented programming, protected objects, generics, and a good support for the implementation of concurrent tasks and thus, the parallelism of several computation steps of the whole system which can reduce the computation time.

CHAPTER 2

# Train model

This chapter describes the physical model of the train in detail. In particular, on one hand formulas to calculate the behavior of the train on the track, and on the other hand formulas to calculate the energy demand are given. A similar physical model is implemented in several simulation tools, but for integrity it is explained here in detail. Most of the given formulas are based on [6] and [15] and modified as they are used in *OpenTrack* [36, 37].

The algorithms in Chapter 3 and 5 use this model to calculate the movements of the trains and to calculate the energy consumption.

As it is know from fundamental laws of dynamics, the movement of a train is influenced by three forces, namely:

- The *traction force* $F(v)$ acts in the trains forward motion (Section 2.1).
- The *braking force* $B(v)$ acts against the forward motion (Section 2.2).
- In general, the *resistance force* $R(v, s)$ acts against the forward motion of a train. Several types of resistance forces are explained in Section 2.3.

Based on these types of forces, there are three different situations which influence the motion of a train. They depend on the current train speed $v$ and on the position $s$ of the train on the track.

- $F(v) - B(v) - R(v) > 0 \ldots$ The train will increase its speed.
- $F(v) - B(v) - R(v) = 0 \ldots$ The train will hold the current speed.
- $F(v) - B(v) - R(v) < 0 \ldots$ The train will decrease its speed until it stops.

The algorithms to determine the optimal driving strategy for a train (Chapter 3) and for the optimization of a railway system (Chapter 5) calculate the forces at each position by using the current train speed.

## 2.1 Traction force

The traction force of a traction vehicle is given as a table with two columns – the first one is the speed $v$ in $\mathrm{km/h}$ and the second one the corresponding force on the wheels $F(v)$ in $\mathrm{kN}$. Thus, for each speed value $v$ a corresponding traction force value $F(v)$ is defined.

**Example 2.1.** *Figure 2.1 shows the velocity–traction-force diagram of the* ÖBB 4124 *(Bombardier Talent) [27]. The maximum traction effort of the train is located between* $0\ \mathrm{km/h}$ *and* $47\ \mathrm{km/h}$ *and is* $110.00\ \mathrm{kN}$.

*Between* $47\ \mathrm{km/h}$ *and the maximum speed of the train (*$v = 140\ \mathrm{km/h}$, $F(v) = 32.37\ \mathrm{kN}$*), there is a nearly continuous traction effort which is inversely proportional to the corresponding speed value. Thus, the traction force decreases with higher speed values and as a consequence the acceleration will decrease, too. This effect will be increased by several resistance values which will be explained in section 2.3.*

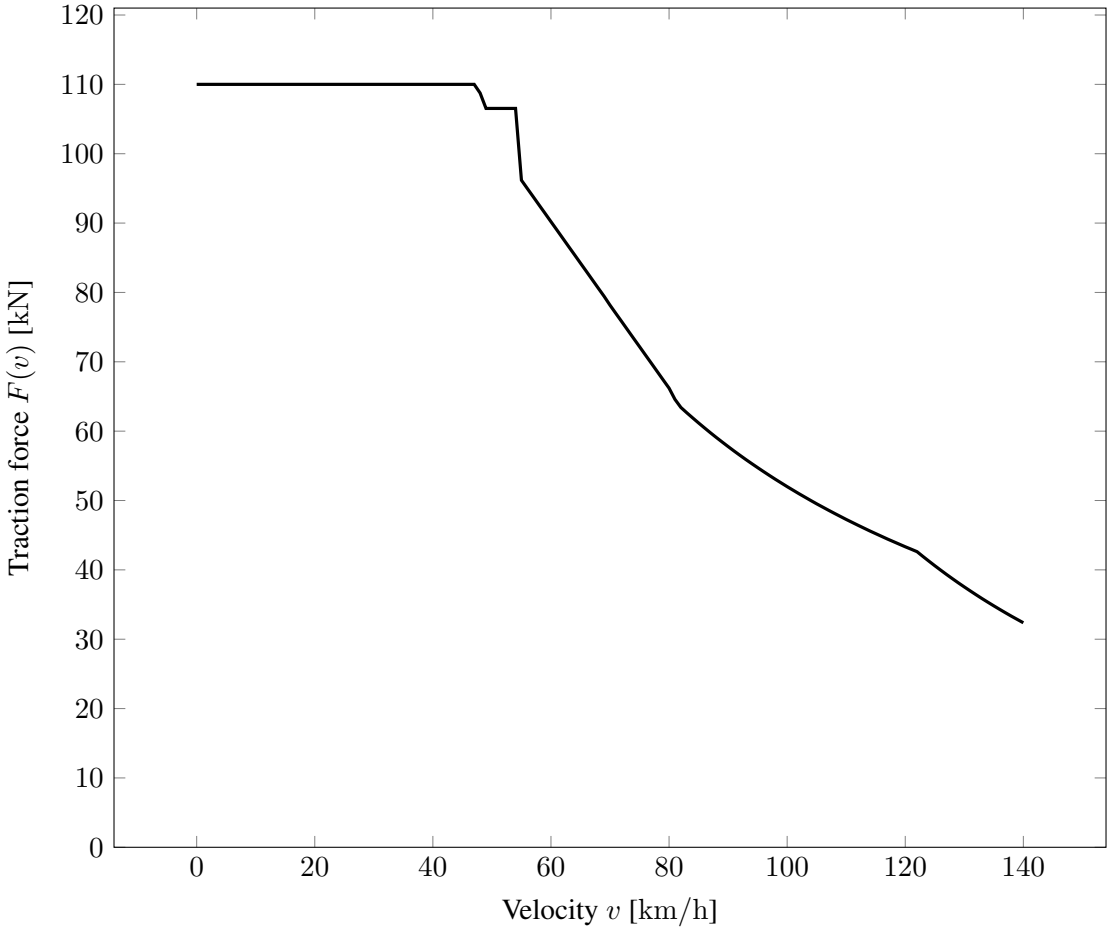



**Figure 2.1:** $vF$-Diagram of ÖBB 4124 (Bombardier Talent, Example 2.1) [27]

## 2.2 Braking force

Similarly to the traction force, the braking force is also given as a table with the speed value $v$ in $\mathrm{km/h}$ in the first column and the braking force $B(v)$ in $\mathrm{kN}$ in the second one. Again, for each speed value $v$ a corresponding braking force $B(v)$ is defined.

**Example 2.2.** *The velocity–braking-force diagram of* ÖBB 4124 *(Bombardier Talent) [26] is shown in Figure 2.2. The maximum braking force is located between* $0\ \mathrm{km/h}$ *and* $80\ \mathrm{km/h}$ *and is* $80\ \mathrm{kN}$.

*Between* $80\ \mathrm{km/h}$ *and the maximum speed of the train (*$v = 140\ \mathrm{km/h}$, $B(v) = 45\ \mathrm{kN}$*), there is a decreasing braking power, again inversely proportional to the corresponding speed value.*

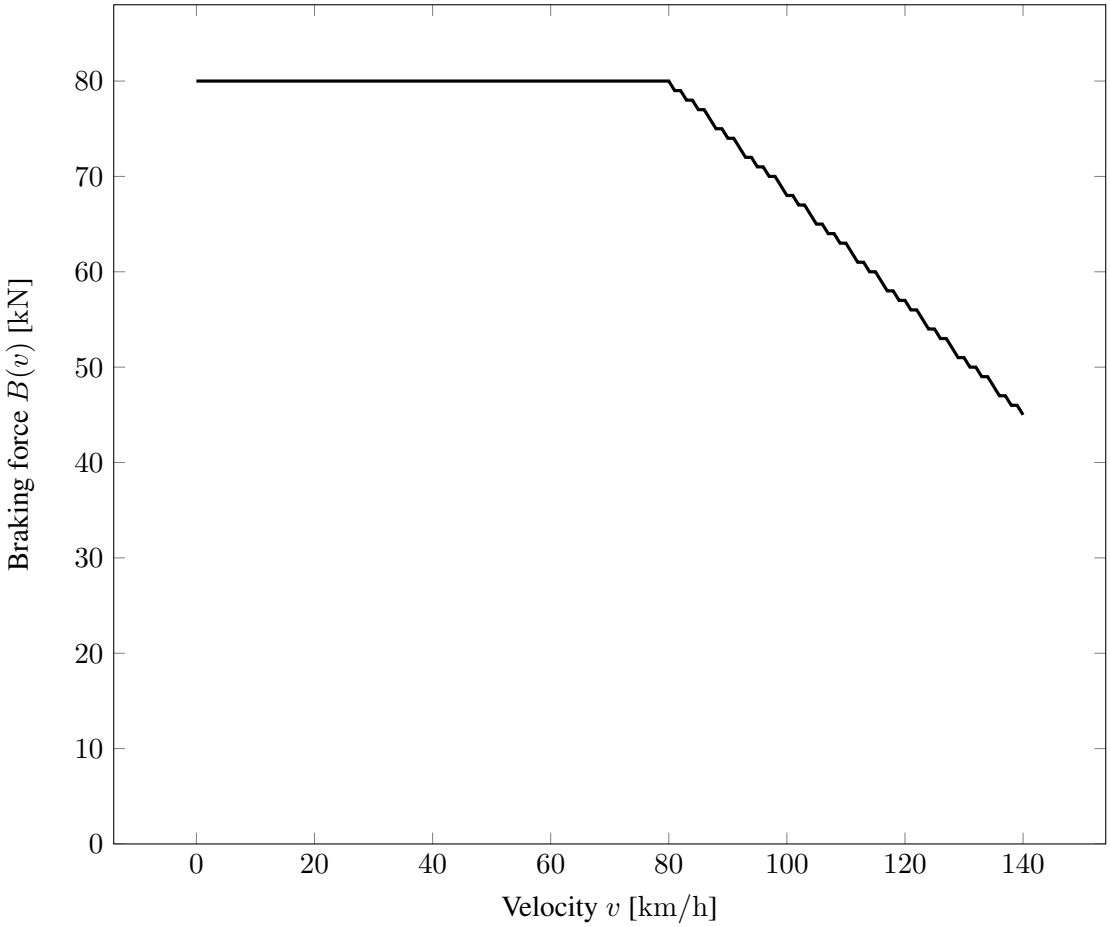



**Figure 2.2:** $vB$-Diagram of ÖBB 4124 (Bombardier Talent, Example 2.2) [26]

## 2.3 Resistance

This section describes the calculation of the resistance which acts in general against the motion of the train. The given formulas are based on [6], [15], [37], and [36].

As already mentioned, if the resistance is greater than the traction force, the train will decrease its speed. The complete resistance of a train is given in Equation 2.1 and is calculated by the sum of its *traction resistance* and its *acceleration resistance*. The used parameters are explained in Table 2.1.

$$R(v, s) = R_{Tr}(v, s) + R_A(v, s) \tag{2.1}$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $R(v, s)$ | Resistance | $[R(v, s)] = \mathrm{N}$ |
| $R_{Tr}(v, s)$ | Traction resistance | $[R_{Tr}(v, s)] = \mathrm{N}$ |
| $R_A(v, s)$ | Acceleration resistance | $[R_A(v, s)] = \mathrm{N}$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.1:** Resistance

### Traction Resistance

The calculation of the traction resistance is given in Equation 2.2. It is calculated by summing up the *rolling resistance* and the *distance resistance*. An overview of the parameters is given in Table 2.2.

$$R_{Tr}(v, s) = R_R(v, s) + R_D(s) \tag{2.2}$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_{Tr}(v, s)$ | Traction resistance | $[R_{Tr}(v, s)] = \mathrm{N}$ |
| $R_R(v, s)$ | Rolling resistance | $[R_R(v, s)] = \mathrm{N}$ |
| $R_D(s)$ | Distance resistance | $[R_D(s)] = \mathrm{N}$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.2:** Traction resistance

**Rolling resistance**

The rolling resistance contains the *train resistance*, the *wagon resistance*, and the *tunnel resistance*. Equation 2.3 shows the formula to calculate the rolling resistance for passenger trains, Table 2.3 gives an overview of the parameters.

$$R_R(v, s) = R_{RT}(v) + R_{RP}(v) + R_T(v, s) \tag{2.3}$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_R(v, s)$ | Rolling resistance | $[R_R(v, s)] = \mathrm{N}$ |
| $R_{RT}(v)$ | Train resistance | $[R_{RT}(v)] = \mathrm{N}$ |
| $R_{RP}(v)$ | Wagon resistance for passenger trains | $[R_{RP}(v)] = \mathrm{N}$ |
| $R_T(v, s)$ | Tunnel resistance | $[R_T(v, s)] = \mathrm{N}$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.3:** Rolling resistance

Instead of $R_{RP}(v)$, $R_{RF}(v)$ can be used for freight trains. In general, the rolling resistance of a train and its wagons is given by the *Formula of Davis* [15]:

$$R(v) = A \cdot v^2 + B \cdot v + C. \tag{2.4}$$

To configure the parameters in a fine grained level, Equation 2.4 is often given in some more detail, e.g. by using the following formulas:

- *Strahl-formula* to calculate the resistance of the traction vehicle ($R_{RT}(v)$).
- *Sauthoff-formula* to calculate the resistance of the person-wagons ($R_{RP}(v)$).
- *Improved Strahl-formula* to calculate the resistance of the freight-wagons ($R_{RF}(v)$).

**Rolling resistance of traction vehicles.** Equation 2.5 shows the *Strahl-formula* to calculate the resistance of a traction vehicle. Table 2.4 gives an overview of the parameters and their units.

$$R_{RT}(v) = g \cdot \left\{ \left[ f_L \cdot \frac{m_T}{1000} \right] + \left[ k_{St1} \cdot ((v + \Delta v) \cdot 3.6)^2 \right] \right\} \tag{2.5}$$

**Rolling resistance of person-wagons.** Equation 2.6 shows the formula for calculating the rolling resistance of person wagons. Table 2.5 gives an overview of the parameters and their units.

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_{RT}(v)$ | Rolling resistance of the traction vehicle | $[R_{RT}(v)] = \mathrm{N}$ |
| $g$ | Constant of gravitation (9.81 m/s$^2$) | $[g] = \mathrm{m/s^2}$ |
| $m_T$ | Mass of the traction vehicle | $[m_T] = \mathrm{kg}$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $\Delta v$ | Supplement wind speed | $[\Delta v] = \mathrm{m/s}$ |
| $f_L$ | Resistance factor | $[f_L] = 1$ |
| $k_{St1}$ | Resistance coefficient | $[k_{St1}] = \mathrm{kg{\cdot}s2/m^2}$ |

**Table 2.4:** Rolling resistance of the traction vehicle

$$R_{RP}(v) = g \cdot \left\{ \left[ 1.9 \cdot \frac{m_W}{1000} \right] + \left[ k_{Sa1} \cdot v \cdot 3.6 \cdot \frac{m_W}{1000} \right] + \left[ k_{Sa2} \cdot (n + 2.7) \cdot ((v + \Delta v) \cdot 3.6)^2 \right] \right\} \tag{2.6}$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_{RP}(v)$ | Rolling resistance of the person-wagons | $[R_{RP}(v)] = \mathrm{N}$ |
| $g$ | Constant of gravitation $[g] = \mathrm{m/s^2}$ | |
| $m_W$ | Mass of the wagons | $[m_W] = \mathrm{kg}$ |
| $n$ | Number of wagons | $[n] = 1$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $\Delta v$ | Supplement wind speed | $[\Delta v] = \mathrm{m/s}$ |
| $k_{Sa1}$ | Resistance coefficient | $[k_{Sa1}] = \mathrm{s/m}$ |
| $k_{Sa2}$ | Resistance coefficient | $[k_{Sa2}] = \mathrm{kg{\cdot}s^2/m^2}$ |

**Table 2.5:** Rolling resistance of person-wagons.

**Rolling resistance of freight-wagons.** Equation 2.7 shows the formula for calculating the rolling resistance of freight wagons. Table 2.6 gives an overview of the parameters and their units.

$$R_{RF}(v) = g \cdot \frac{m}{1000} \cdot \left[ 2.2 - \frac{k_{St2}}{v \cdot 3.6 + k_{St3}} + k_{St4} \cdot (v \cdot 3.6)^2 \right] \tag{2.7}$$

As already mentioned (Equation 2.4), the resistance of the traction vehicle and the wagons is often given by the *Formula of Davis*. Based on Equation 2.4, by summing up all terms of

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_{RF}(v)$ | Rolling resistance of the freight-wagons | $[R_{RF}(v)] = \mathrm{N}$ |
| $g$ | Constant of gravitation (9.81 m/s$^2$) | $[g] = \mathrm{m/s^2}$ |
| $m_W$ | Mass of the wagons | $[m_W] = \mathrm{kg}$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $k_{St2}$ | Resistance coefficient | $[k_{St2}] = \mathrm{m/s}$ |
| $k_{St3}$ | Resistance coefficient | $[k_{St3}] = \mathrm{m/s}$ |
| $k_{St4}$ | Resistance coefficient | $[k_{St4}] = \mathrm{s^2/m^2}$ |

**Table 2.6:** Rolling resistance of freight-wagons.

Equations 2.5, 2.6, and 2.7, the coefficients $A$, $B$, and $C$ can be calculated by

$$\begin{aligned} A &= g \cdot \Big[ f_L \cdot \frac{m_T}{1000} + k_{St1} \cdot 12.96 \cdot \Delta v^2 + 1.9 \cdot \frac{m_W}{1000} \\ &+ k_{Sa2} \cdot (n + 2.7) \cdot 12.96 \cdot \Delta v^2 \Big] \qquad (2.8) \\ B &= g \cdot \Big[ k_{St1} \cdot 25.92 \cdot \Delta v + k_{Sa1} \cdot 3.6 \cdot \frac{m_W}{1000} + k_{Sa2} \cdot (n + 2.7) \cdot 25.92 \cdot \Delta v \Big] \qquad (2.9) \\ C &= g \cdot [k_{St1} \cdot 12.96 + k_{Sa2} \cdot (n + 2.7) \cdot 12.96] \qquad (2.10) \end{aligned}$$

**Tunnel resistance.** The third term of the sum for calculating the rolling resistance (Equation 2.3) contains the tunnel resistance. It is calculated by using Equation 2.11, the parameters, the description and the units are given in Table 2.7.

$$R_T(v, s) = f_t(s) \cdot v^2 \qquad (2.11)$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_T(v, s)$ | Tunnel resistance | $[R_{T(v,s)}] = \mathrm{N}$ |
| $f_T(s)$ | Tunnel factor | $[f_T(s)] = \mathrm{kg/m}$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.7:** Tunnel resistance

### Distance resistance

The distance resistance consist of the *gradient resistance*, the *curve resistance*, and the *switch resistance*. Due to the minimum impact of the switch resistance in big railway systems, it is ignored in the calculation of the distance resistance. The formula is given in Equation 2.12, the parameters are explained in Table 2.8.

$$R_D(s) = R_G(s) + R_C(s) + R_S(s) \tag{2.12}$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_D(s)$ | Distance resistance | $[R_D(s)] = \mathrm{N}$ |
| $R_G(s)$ | Gradient resistance | $[R_G(s)] = \mathrm{N}$ |
| $R_C(s)$ | Curve resistance | $[R_C(s)] = \mathrm{N}$ |
| $R_S(s)$ | Switch resistance | $[R_S(s)] = \mathrm{N}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.8:** Distance resistance

**Gradient resistance.** The gradient resistance is calculated by

$$R_G(s) = m \cdot g \cdot \sin(\alpha(s)) \tag{2.13}$$

For small inclination $\alpha(s)$, $\sin \alpha(s)$ can be substituted by $\tan(\alpha(s))$:

$$R_G(s) = m \cdot g \cdot \tan(\alpha(s)) = m \cdot g \cdot \frac{I(s)}{1000} \tag{2.14}$$

The used parameters are explained in Table 2.9.

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_G(s)$ | Gradient resistance | $[R_G(s)] = \mathrm{N}$ |
| $\alpha(s)$ | Inclination at position $s$ | $[\alpha(s)] = {}^\circ$ |
| $g$ | Constant of gravitation (9.81 m/s$^2$) | $[g] = \mathrm{m/s^2}$ |
| $m$ | Complete mass of the train ($m_T + m_W$) | $[m] = \mathrm{kg}$ |
| $I(s)$ | Track Gradient at position $s$ | $[I(s)] = ‰$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.9:** Gradient resistance

**Curve resistance.** The curve resistance for a curve radius $r \geq 300$ m is calculated by

$$R_C(s) = \frac{6.3}{r(s) - 55} \cdot m \tag{2.15}$$

and for $r(s) < 300$ m by

$$R_C(s) = \frac{4.91}{r(s) - 30} \cdot m. \tag{2.16}$$

The parameters are explained in Table 2.10.

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_C(s)$ | Curve resistance | $[R_C(s)] = \mathrm{N}$ |
| $r(s)$ | Curve radius | $[r(s)] = \mathrm{m}$ |
| $m$ | Complete mass of the train $(m_T + m_W)$ | $[m] = \mathrm{kg}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.10:** Curve resistance

### Acceleration Resistance

While the train is accelerating or decelerating, the so called *acceleration resistance* acts on the train (Equation 2.17).

$$R_A(v, s) = m \cdot a(v, s) \cdot (1 + 0.01 \cdot \rho) \tag{2.17}$$

The parameters of Equation 2.17 are explained in Table 2.11.

| Symbol | Description | Physical Unit |
|---|---|---|
| $R_A(v, s)$ | Acceleration resistance | $[R_A(v, s)] = \mathrm{N}$ |
| $m$ | Complete mass of the train $(m_T + m_W)$ | $[m] = \mathrm{kg}$ |
| $a(v, s)$ | Acceleration | $[a(v, s)] = \mathrm{m/s^2}$ |
| $\rho$ | Rotating mass factor | $[\rho] = 1$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $s$ | Train position | $[s] = \mathrm{m}$ |

**Table 2.11:** Acceleration resistance

As a result, the complete resistance for a person train is calculated by

$$R(v, s) = R_{RT}(v) + R_{RP}(v) + R_T(v, s) + R_G(s) + R_C(s) + R_S(s) + R_A(v, s) \tag{2.18}$$

and for a freight train by

$$R(v, s) = R_{RT}(v) + R_{RF}(v) + R_T(v, s) + R_G(s) + R_C(s) + R_S(s) + R_A(v, s) \tag{2.19}$$

and by using the Formula of Davis

$$R(v, s) = A \cdot v^2 + B \cdot v + C + R_T(v, s) + R_G(s) + R_C(s) + R_S(s) + R_A(v, s) \tag{2.20}$$

whereby $A$, $B$, and $C$ depends on the train type (passenger or freight train) and its characteristics.

## 2.4 Train Dynamics

This section describes the calculation of the acceleration value $a(v, s)$ for a train accelerating, braking, and coasting. Additionally, formulas for speed-holding and the calculation of the current speed value and the travel time of a train are given. The formulas are based on the equations of the previous section (Equations 2.1 - 2.20). To calculate the motion of a train, it is necessary to know the traction force $F(v)$, the braking force $B(v)$ and the resistance values $R(v, s)$. Based on these parameters, it can be determined if the train accelerates, holds its speed, or decelerates. In the following, it is assumed that for both, traction force and braking force, the maximum force will be used. This approach is used typically in the field and has mathematical and physical reasons.

### Acceleration

Assuming that the driver will use maximum traction force ($F(v) > 0$ N) – and thus no braking force ($B(v) = 0$ N) – the acceleration value for passenger trains is be calculated by:

$$a(v, s) = \frac{F(v) - (R_{RT}(v) + R_{RP}(v) + R_T(v, s)) - (R_G(s) + R_C(s) + R_S(s))}{m \cdot (1 + 0.01 \cdot \rho)} \tag{2.21}$$

As already mentioned, the resistance parameters depend on various factors, e.g. the train mass, the current speed, or the inclination of the track. Based on Equation 2.21, the following three different kinds of situations can happen:

- The train will increase its speed if

$$F(v) > R_{RT}(v) + R_{RP}(v) + R_{RP}(v) + R_G(s) + R_C(s) + R_S(s).$$

- The train will hold its speed if

$$F(v) = R_{RT}(v) + R_{RP}(v) + R_{RP}(v) + R_G(s) + R_C(s) + R_S(s).$$

- The train will decrease its speed if

$$F(v) < R_{RT}(v) + R_{RP}(v) + R_{RP}(v) + R_G(s) + R_C(s) + R_S(s).$$

### Deceleration

Assuming that the driver will use the maximum braking power ($B(v) > 0$ N) – and thus no traction force is available ($F(v) = 0$ N) – the deceleration value for passenger trains is calculated by:

$$a(v, s) = -\left[\frac{B(v) + R_{RT}(v) + R_{RP}(v) + R_T(v) + R_G(s) + R_C(s) + R_S(s)}{m \cdot (1 + 0.01 \cdot \rho)}\right] \tag{2.22}$$

Similar to the acceleration, again three different kinds of situation can occur:

- The train will decrease its speed if

$$B(v) > R_{RT}(v) + R_{RP}(v) + R_{RP}(v) + R_G(s) + R_C(s) + R_S(s).$$

- The train will hold its speed if

$$B(v) = R_{RT}(v) + R_{RP}(v) + R_{RP}(v) + R_G(s) + R_C(s) + R_S(s).$$

- The train will increase its speed if

$$B(v) < R_{RT}(v) + R_{RP}(v) + R_{RP}(v) + R_G(s) + R_C(s) + R_S(s).$$

### Coasting

Assuming that the train driver neither uses its traction force nor its acceleration force. As a consequence, the train is in its so-called *coasting phase*. In that case only the resistance values influence the motion of the train. Equation 2.23 shows the formula for calculating the acceleration value of a passenger train.

$$a(v,s) = -\left[\frac{R_{RT}(v) + R_{RP}(v) + R_T(v) + R_G(s) + R_C(s) + R_S(s)}{m \cdot (1 + 0.01 \cdot \rho)}\right] \tag{2.23}$$

Again, three different situation can happen, based on the resistance parameters:

- The train will decrease its speed

$$-R_G(s) < R_{RT}(v) + R_{RP}(v) + R_T(v) + R_C(s) + R_S(s)$$

- The train will hold its speed

$$-R_G(s) = R_{RT}(v) + R_{RP}(v) + R_T(v) + R_C(s) + R_S(s)$$

- The train will increase its speed if

$$-R_G(s) > R_{RT}(v) + R_{RP}(v) + R_T(v) + R_C(s) + R_S(s)$$

### Speed-holding

To hold the current speed value the parameters acting in the direction of the trains motion and the parameters against the motion must be in equality:

$$F(v) - B(v) - (R_{RT}(v) + R_{RP}(v) + R_{RP}(v)) - (R_G(s) + R_C(s) + R_S(s)) = 0. \tag{2.24}$$

| Symbol | Description | Physical Unit |
|---|---|---|
| $s$ | Travel distance | $[s] = \mathrm{m}$ |
| $s_0$ | Starting position | $[s_0] = m$ |
| $t$ | Travel time | $[t] = \mathrm{s}$ |
| $a(v_0, s_0)$ | Acceleration value | $[a(v_0, s_0)] = \mathrm{m/s^2}$ |
| $v_0$ | Starting speed value | $[v_0] = \mathrm{m/s}$ |

**Table 2.12:** Calculation of the speed value

### Calculation of the speed value

After the calculation of the acceleration value for a given speed and a given position, the speed value of the next position can be determined by

$$s = s_0 + v_0 \cdot t + \frac{1}{2} \cdot a(v_0, s_0) \cdot t^2. \tag{2.25}$$

where $v_0$ is the current speed and ans $s_0$ the current position. The parameters of Equation 2.25 are explained in Table 2.12.

### Calculation of the travel time

If a starting position $s_0$, the driving distance $d$, the initial speed $v_0$, and the acceleration value $a$ are known, the required travel time $t$ and the speed value $v$ for a driving distance $d$ can be calculated by

$$v = v_0 + \Delta v = v_0 + a(v_0, s_0) \cdot t. \tag{2.26}$$

These formulas can be used for a train accelerating ($a(v_0, s_0) > 0\ \mathrm{m/s^2}$), decelerating ($a(v_0, s_0) < 0\ \mathrm{m/s^2}$), and speed holding ($a(v_0, s_0) = 0\ \mathrm{m/s^2}$). By using the numerical integration with a step size of $s = 1\ \mathrm{m}$, the resulting speed $v$ and travel time $t$ after distance $d$ can be calculated.

## 2.5 Energy consumption

As it is know from fundamental physical laws (cf. [2]) the overall energy consumption can be calculated by integrating the resulting force $\hat{F}(v, s) = F(v) - B(v) - R(v, s)$ over the covered distance $s$:

$$E(s) = \int \hat{F}(v, s) \mathrm{d}s \tag{2.27}$$

Parameters of Equation 2.27 are explained in Table 2.13.

By using numerical integration with a step size of $1\ \mathrm{m}$ the following three kinds of situations are possible:

| Symbol | Description | Physical Unit |
|---|---|---|
| $E(s)$ | Energy consumption | $[E(s)] = \mathrm{kWh}$ |
| $F(\hat{v}, s)$ | Force | $[F(\hat{v}, s)] = N$ |
| $v$ | Train velocity | $[v] = \mathrm{m/s}$ |
| $s$ | Travel distance | $[s] = \mathrm{m}$ |

**Table 2.13:** Energy consumption

- Energy demand (e.g. while accelerating): $\hat{F}(v, s) > 0$: $E(s) = \sum_{i=0}^{s} \hat{F}_i(v, s) \cdot 1\ \mathrm{m}$.
- No energy demand or recovery (e.g. while coasting): $\hat{F}(v, s) = 0$: $E(s) = 0$.
- Energy recovery (e.g. while braking, if energy recuperation is used): $\hat{F}(v, s) < 0$: $E(s) = r \cdot \sum_{i=0}^{s} \hat{F}_i(v, s) \cdot 1\ \mathrm{m}$.

The resulting unit for an energy value is J. In the field, $\mathrm{kWh}$ is typically used and thus the result has to be transformed by

$$E_{kWh}(s) = E_J(s) \cdot 3\,600\,000 \tag{2.28}$$

## 2.6 Implementation of the train model

This section gives an overview of the most important functions which are used in the algorithm to determine the optimal driving strategy for a train.

### Motion of the train

In general, the following situations must be distinguished while determining the motion of a train:

- When the train driver wants to accelerate, $F(v_0)$ is taken from the speed–traction-force diagram (Section 2.1) and $B(v_0) = 0$ because no braking force is acting against the motion of the train.
- When the train driver uses the brakes, $B(v_0)$ is taken from the speed–braking-force diagram (Section 2.2) and $F(v_0) = 0$ because no traction force is action on the train.
- Assuming that the train driver uses the cruise control of a train and wants to hold the current speed, the following cases must be distinguished, depending on the resistance values of the train, especially the inclination.
  - The train must use a particular traction force ($F(v_0) > 0$, $B(v_0) = 0$) to hold the current speed value. Otherwise the speed of the train will be decreased.
  - The train must use a particular braking force ($B(v_0) > 0$, $F(v_0) = 0$) to hold the current speed value. Otherwise the train will accelerate.

– No additional force will be applied to the motion of the train ($F(v_0) = 0$, $B(v_0) = 0$).

**Algorithm 2.1:** $trainMotion$ (Calculate the motion of a train)

**input** : $v_0$,$s_0$,$t_0$,$e_0$
**output**: $v$, $s$, $t$, $e$

$a \leftarrow getAcceleration(v_0, s_0)$;
$(v, t, s) \leftarrow calcSpeed(v_0, t_0, s_0, a)$;
$e \leftarrow e_0 + calcEnergy(v, s)$;
**return** $(v, t, s, e)$;

Depending on the above restrictions Algorithm 2.1 calculates the motion of the train. In particular, the resulting speed value $v$, the resulting travel time $t$ and the new position $s$ are determined by assuming that the train starts at position $s_0$ with a given speed value $v_0$ at a given instant of time $t_0$ and drives exactly $1\,\mathrm{m}$. Based on the current speed value, the position, the applied traction force $F(v_0)$ and braking force $B(v_0)$ the acceleration value $a$ and the energy consumption $e$ for the trip are calculated.

**Example 2.3.** *Assume that a train starts at a railway station at position $s_0 = 1\,000\,\mathrm{m}$ with $v_0 = 0\,\mathrm{km/h}$ and accelerates until position $s_1 = 1\,400\,\mathrm{m}$ is reached. The train starts its journey at $t_0 = 0\,\mathrm{s}$. Now Algorithm 2.1 is used in a loop.*

**Algorithm 2.2:** Algorithm of Example 2.3

$v_0 \leftarrow 0\,\mathrm{km/h}$;
$s_0 \leftarrow 1\,000\,\mathrm{m}$;
$s_1 \leftarrow 1\,400\,\mathrm{m}$;
$t_0 \leftarrow 0\,\mathrm{s}$;
$e_0 \leftarrow 0\,\mathrm{kWh}$;
$s \leftarrow s_0$;
$v \leftarrow v_0$;
$t \leftarrow t_0$;
$e \leftarrow e_0$;
**while** $s < s_1$ **do**
  $(v, s, t, e) \leftarrow trainMotion(v, s, t, e)$;
**end**

*By applying Algorithm 2.2 the train finally reaches position $s = s_1$. The speed value $v$ holds the speed at this position, $t$ contains the travel time and $e$ the energy consumption.*

## Determining the speed change of a train

As already mentioned the acceleration of deceleration and thus the train speed depends on several parameters, in particular on the traction force $F(v_0)$, the braking force $B(v_0)$ and the resis-

tance values $R(v_0, s_0)$. To calculate the acceleration value $a$ for a given train speed $v_0$ at position $s_0$, Algorithm 2.3 is used.

**Algorithm 2.3:** $getAcceleration$ (Determining the acceleration value)

**input** : $v_0$, $s_0$
**output**: $a$

1 $a \leftarrow \frac{F(v_0)-B(v_0)-R(v_0,s_0)}{m\cdot(1.0+0.01\cdot\rho)}$;
2 **return** $a$;

Now, $a$ can have the following values:

- $a > 0$: The train will accelerate because the applied traction force is bigger than the sum of the braking force $B(v_0)$ and the resistance values $R(v_0, s_0)$. It might be possible that the train increases its speed although no traction force is applied because of a high negative inclination of the track.

- $a < 0$: The train will decelerate because the sum of the resistance values and the applied braking force is bigger than the traction force. If there is a high positive inclination on the track, the train might lose some speed although the maximum traction force is applied.

- $a = 0$: The train will hold the current speed because the used forces balance each other.

Algorithm 2.4 shows the algorithm to calculate the the resulting speed value $v$, the travel time $t$, and the next position $s$ for a given input speed $v_0$, an initial travel time $t_0$, a given position $s_0$, and an acceleration value $a$.

**Algorithm 2.4:** $calcSpeed$ (Calculating the speed and travel time)

**input** : $v_0$, $t_0$, $s_0$, $a$
**output**: $v$, $t$, $s$
1 **if** $a > 0$ **then**
2 $\quad p \leftarrow \frac{2\cdot v_0}{a}$;
3 $\quad q \leftarrow -\frac{2}{a}$;
4 **else**
5 $\quad p \leftarrow -\frac{2\cdot v_0}{a}$;
6 $\quad q \leftarrow \frac{2}{a}$;
7 **end**
8 $D = \left(\frac{p}{2}\right)^2 - q$;
9 $\Delta t \leftarrow -\frac{p}{2} + \sqrt{D}$;
10 $\Delta v \leftarrow a \cdot \Delta t$;
11 $v \leftarrow v_0 + \Delta v$;
12 $t \leftarrow t_0 + \Delta t$;
13 $s \leftarrow s_0 + 1\ \mathrm{m}$;
14 **return** $(v, t, s)$;

### Calculating the energy consumption

Equation 2.27 from Section 2.5 is used to calculate the energy consumption of a train. Algorithm 2.5 describes the algorithm for the calculation, where $getResistance(v, a, s)$ calculates the complete resistance as described in Section 2.3.

**Algorithm 2.5:** $calcEnergy$ (Calculate the energy consumption of a train)

**input** : $v$, $s$
**output**: $e$
1 $r \leftarrow getResistance(v, s)$;
2 $e \leftarrow e + r \cdot 3\,000\,000$;
3 **return** $e$;

Now, the train model is explained in detail and an overview about the implementation is given. The next chapter explains the algorithm to find the optimal driving strategy of a train, based on the restrictions of this chapter.

Further examples showing the impact of the train parameters and the track parameters are given in Section 3.6.

CHAPTER 3

# Single trip optimization

## 3.1 Introduction

As already discussed in several scientific publications, there exist a lot of algorithms to find the optimal driving strategy for the journey of a single train, which cannot be used for real-time-applications, because of practical restrictions (e.g. Chapter 1) like computation time, passenger comfort or drive-ability.

In general, the optimal driving strategy depends on various factors of influence, e.g.

- the gradient of the track and speed restrictions,
- the length and weight of the train,
- the scheduled travel time,
- traction power and braking capability,
- recuperation behavior, and
- various resistance parameters.

It is know from several publications (cf. [2, 46]) that the optimal driving strategy for a train, which starts at a given position $s_s$ and drives within a given travel time $t$ on a specified track to reach its destination (final position) $s_f$, may contain the following *driving modes* in exactly this order:

1. *Acceleration-phase*. The train starts its acceleration-phase with maximum traction force at the start position $s_s$ with a given starting speed $v_s$ (e.g. $0\,\mathrm{km/h}$ at a railway station) and accelerates until the hold speed $v_h$ and the hold position $s_h$ are reached. The resulting travel time for this phase is $t_a$.

2. *Speed-hold-phase*. The train starts its speed-hold-phase at the hold position $s_h$ with hold speed $v_h$ and drives with constant speed until the coasting position $s_c$ is reached. The speed value at this position is called coasting speed $v_c$ and is equal to $v_h$. The travel time for the hold phase is $t_h$.

3. *Coasting-phase*. The coasting-phase starts at the coasting position $s_c$ and ends at the braking position $s_b$. During this phase, the train will not use its traction or braking force. It will just roll until the braking position is reached. The travel time of this phase is $t_c$.

4. *Braking-phase*. The braking-phase starts at the braking position $s_b$ and ends at the final position $s_f$. The speed decreases from $v_b$ to $v_f$ (e.g. to $0\,\mathrm{km/h}$ at a commercial stop which can be station or stop). The required travel time for the braking-phase is $t_b$.

The speed values above are limited by the maximum train speed and the track speed limit. Usually, the start position $s_s$, the corresponding speed $v_s$, the final position $s_f$, and the final speed value $v_f$ of a trip are known. Thus the challenge is to determine the other three tuples $(s_h, v_h), (s_c, v_c), (s_b, v_b)$ with respect to the following restriction:

- A train $T_k$ must reach its destination within the planned travel time $t^k_{\text{⏲}}$.
- The driving-comfort for passengers (e.g. there should be a minimum of transitions between the driving types) and the driver (e.g. each driving type should last at least a given duration in seconds) must be ensured .
- The energy consumption must be minimal. It is calculated by $e = e_d - e_r$ where $e$ is the energy consumption, $e_d$ is the energy demand, and $e_r$ is the energy recovery of the trip.
- The result must be available as fast as possible to use the algorithm for online-systems.

**Definition 3.1** (Maximum train speed)**.** *The maximum speed of train $k$, based on the train characteristics (Chapter 2), is denoted by $\overline{v}^k$.*

**Definition 3.2** (Maximum allowed track speed)**.** *The maximum allowed speed on track section $i$ is denoted by $\overline{v}_i$.*

**Definition 3.3** (Maximum speed)**.** *The maximum allowed speed $\overline{v}$ is defined as $\overline{v} = \min\left(\overline{v}^k, \overline{v}_i\right)$.*

**Definition 3.4** (Regime)**.** *A part of the track with constant maximum speed is called a regime.*

As a consequence, a track with constant track speed consists of exactly one regime. The definition of regime can be extended:

**Definition 3.5** (Regime (extended))**.** *A part of the track with constant maximum and no (significant) gradient changes is called a regime.*

The following sections describe the algorithm to find the optimal driving strategy for a single train with no interaction to other trains. The explanation starts with a track, consisting of a single regime and will be extended to a multi-regime optimization.

## 3.2 Simple Algorithm

This section describes the algorithm to find the optimal driving strategy for a train with a given maximum speed along the complete track and thus, with only one regime, based on the previously mentioned restrictions. A train $k$ starts at a railway station with $v_s = 0\ \mathrm{km/h}$ and should arrive at another railway station after the planned travel time $t^k_{\circledcirc}$. Thus, $v_f = 0\ \mathrm{km/h}$.

The following algorithm is very efficient in terms of calculation time and can be adjusted by several parameters. The first part of the algorithm consists of finding a driving strategy consisting of

- acceleration-phase,
- speed-hold-phase, and
- braking-phase.

To get a valid driving strategy the following equation must be fulfilled:

$$t^k_{\circledcirc} \approx t^k_a + t^k_h + t^k_b \tag{3.1}$$

In general, it is hard to find a driving strategy where the left side of Equation 3.1 conforms exactly the planned travel time. Thus, a result is valid, if it is within a slight time interval around $t^k_{\circledcirc}$ and for this reason the $\approx$-operator is used.

Due to the fact that the speed values at the beginning $v_s$ and at the end $v_f$ of the journey are known and assuming that the maximum force is used for the acceleration and braking-phase, only the hold speed $v_h$ must be determined. This is done by using a binary search algorithm.

### Algorithm

Algorithm 3.1 determines the driving strategy and works as follows:

- If the planned travel time $t^k_{\circledcirc}$ for train $k$ is less than the shortest travel time (maybe also running time), no solution can be found.
- Otherwise, the algorithm loops until an appropriate solution ($t^k_{\circledcirc} \approx t^k_a + t^k_h + t^k_b$) is found:
  - Calculate a new hold speed $v_h = \frac{\underline{v}+\overline{v}}{2}$, where $\underline{v}$ is the lower boundary speed value[1] and $\overline{v}$ is the upper boundary speed value.
  - The train starts at $s_h$ and accelerates until the hold speed $v_h$ is reached.
  - Calculate the braking position for $v_h$[2].
  - Hold the current speed, starting at the hold position $s_h$ until the braking position $s_b$ is reached.

[1] $\underline{v}$ is set to a value slightly greater than $0\ \mathrm{km/h}$ initially.

[2] When the train starts braking at the braking position (with maximum braking force), it will decrease its speed until it stops at the final position.

- Brake until the final position $s_f$ is reached.
- Calculate the complete travel time $t$.
- If an appropriate solution ($t^k_{\clock} \approx t^k_a + t^k_h + t^k_b$) is found, calculate the energy consumption for this driving strategy.
- If no solution is found, the upper boundary speed value or the lower boundary speed value is changed:
  * $\overline{v} = v_h$, if $t < t^k_{\clock}$
  * $\underline{v} = v_h$, if $t > t^k_{\clock}$

**Algorithm 3.1:** $calcHoldSpeed$ (Algorithm to calculate the hold speed)

**input** : $s_s$, $s_f$, $t^k_{\clock}$, $\overline{v}$
**output**: $s_h$, $s_b$, $v_h$, $v_b$, $e$

```
not_finished ← true;
v̲ ← 0;
while not_finished do
    v_h ← (v̲ + v̄) / 2;
    (s_h,t_a,e_a) ← Accelerate (s_s,v_h) ;
    (s_b,t_b) ← GetBrakingPosition (s_f v_h) ;
    (t_h,e_h) ← SpeedHold (s_h,s_b,v_h) ;
    (t_b,e_b) ← Brake (s_b,s_f) ;
    t ← t_a +t_h +t_b;
    if t ≈ t^k_clock then
        not_finished ← false;
        e ← e_a +e_h +e_b;
        v_b ← v_h;
    else
        if t < t^k_clock then
            v̄ ← v_h;
        else
            v̲ ← v_h;
        end
    end
    return (s_h,s_b,v_h,v_b,e);
end
```

**Example 3.1.** *In the following example, the hold-strategy for a single train is calculated. Its journey starts at* $s_s = 19\,584\,\mathrm{m}$ *and ends at* $s_f = 26\,955\,\mathrm{m}$. *The planned travel time is* $t^k_{\clock} = 270\,\mathrm{s}$. *The train-specific parameters can be found in Table 3.1 and the track gradient is illustrated in Figure 3.1.*

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $m_T$ | 55 000 t | $m_W$ | 82 000 t |
| $n$ | 2 | $r$ | 0 |
| $F(v)$ | see Ex. 2.1 | $B(v)$ | see Ex. 2.2 |
| $l$ | 67 m | $f_L$ | 3.3 |
| $k_{St_1}$ | $0.03\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $k_{Sa_1}$ | $0.0025\ \mathrm{s/m}$ |
| $k_{Sa_2}$ | $0.00696\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $\Delta v$ | $4.17 m/s$ |
| $\rho$ | 6.0 | | |

**Table 3.1:** Example

*The resulting driving strategy consisting of an acceleration-phase, a speed-hold-phase, and a braking-phase is illustrated in Figure 3.2. The train will reach its destination after* $t = 269.9$ s. *The calculated positions and speed values are given in Table 3.2. Figure 3.3 shows the distance-time diagram and Figure 3.4 the accumulated energy consumption.*

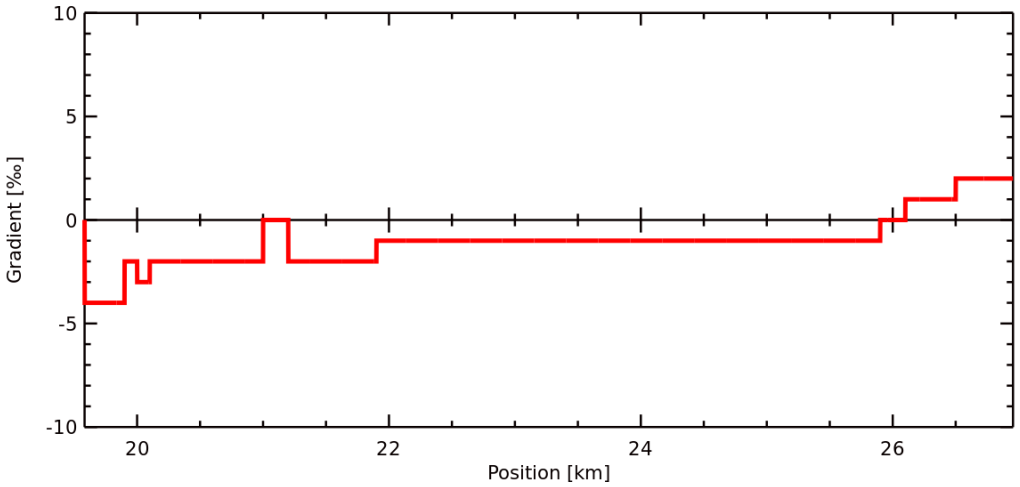



**Figure 3.1:** Gradient (Example 3.1)

| Position | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 19 583 m | $v_s$ | 0 km/h |
| $s_h$ | 21 252 m | $v_h$ | 125.3 km/h |
| $s_b$ | 25 862 m | $v_b$ | 125.3 km/h |
| $s_f$ | 26 955 m | $v_f$ | 0 km/h |

**Table 3.2:** Resulting positions of Example 3.1

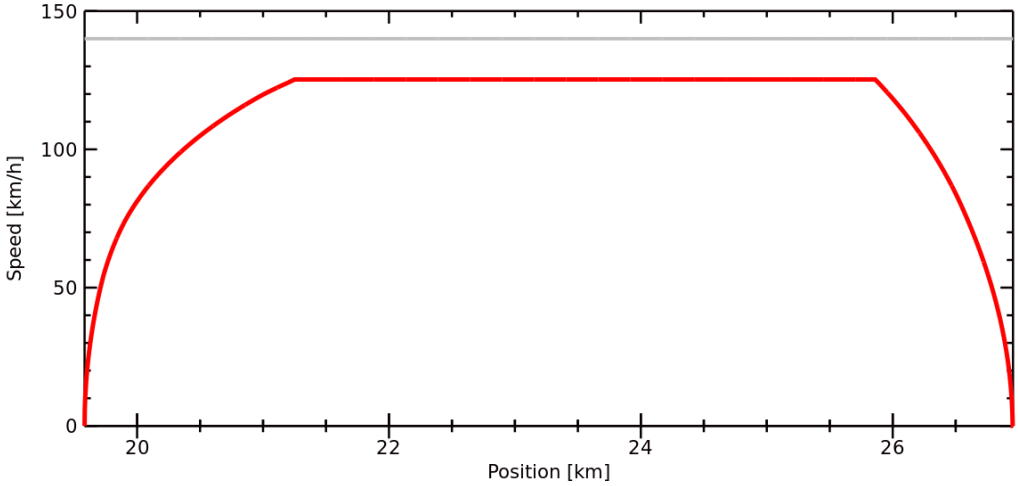



**Figure 3.2:** Driving strategy (Example 3.1)

*Obviously, the energy demand has the strongest growth while accelerating. Within the speed-hold-phase it grows less and while braking, the energy demand will not change in this example because the recovery factor was set to* 0*. In general, the optimal driving strategy depends on the recovery factor.*

Now a driving strategy without a coasting phase was found, which forms the base for the optimization. In general, the optimal driving strategy with minimum energy consumption will have a higher hold speed than the hold strategy and the speed-hold-phase will be divided into a speed-hold-phase and a coasting-phase and thus, the following values must be determined:

- $v_h'$: The hold-speed of the optimal driving strategy is bounded by the hold-speed of the hold-strategy and the maximum allowed speed:

$$v_h \leq v_h' \leq \overline{v} \tag{3.2}$$

- $s_c'$: A coasting position must be found for the optimal driving strategy. It is bounded by the hold position and the braking position of the previously calculated hold-strategy:

$$s_h \leq s_c' \leq s_b \tag{3.3}$$

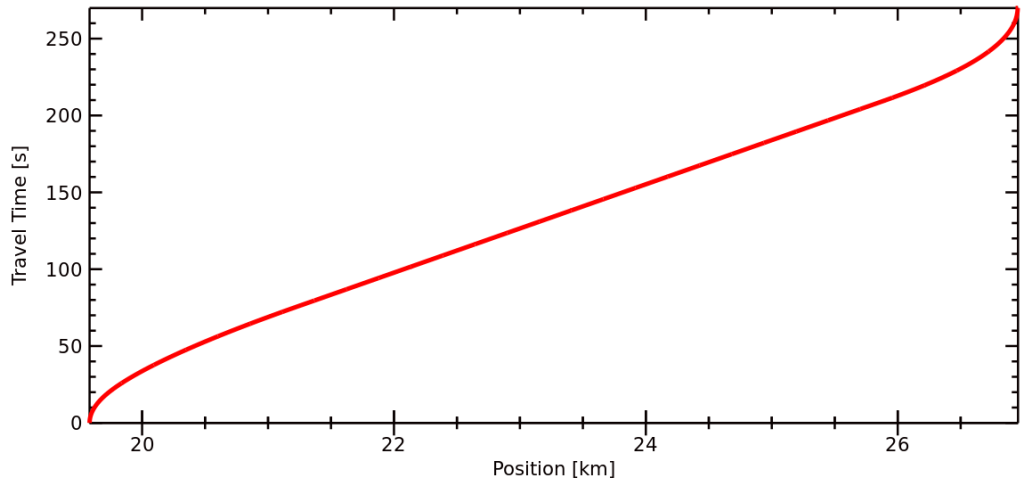



**Figure 3.3:** Distance-time diagram (Example 3.1)

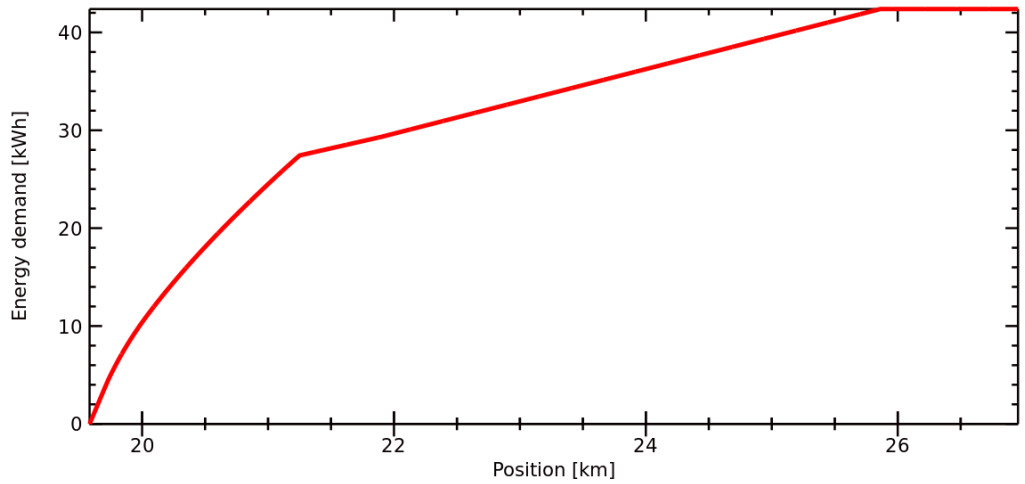



**Figure 3.4:** Energy consumption (Example 3.1)

- Based on these two values, the other speed values and positions will be determined.

As a result, the challenge is to find the hold-speed $v'_h$ and the coasting position $s'_c$ in a way that the energy consumption is as small as possible. Assuming that $v_h = 80\ \mathrm{km/h}$ and $\overline{v} = 140\ \mathrm{km/h}$, the hold-speed for the optimal driving strategy will be $80\ \mathrm{km/h} \leq v'_h \leq 140\ \mathrm{km/h}$. Due to the gradient values, it is not possible to use a binary search algorithm to find the best speed value and thus, every value in this range might be the best solution. Due to restrictions on the calculation

time, a parameter to the algorithm is introduced to adjust the step size within the range: $v_+$. As a result, the following hold-speed values will be used for the determination of the optimal driving strategy:

$$v_h, v_h + v_+, v_h + 2 \cdot v_+, v_h + 3 \cdot v_+ \ldots \overline{v}$$

When using $v_+ = 5\ \mathrm{km/h}$, the following hold-speed values will be used for further calculations:

$$80\ \mathrm{km/h}, 85\ \mathrm{km/h}, 90\ \mathrm{km/h}, 95\ \mathrm{km/h}, 100\ \mathrm{km/h}, 105\ \mathrm{km/h}, 110\ \mathrm{km/h},$$
$$115\ \mathrm{km/h}, 120\ \mathrm{km/h}, 125\ \mathrm{km/h}, 130\ \mathrm{km/h}, 135\ \mathrm{km/h}, 140\ \mathrm{km/h}$$

Due to the fact that some cruise control systems of the trains can be adjusted only in 5-$\mathrm{km/h}$-steps, this parameter can be used to set-up the algorithm in a way that the computed results can be used in practice. Based on this parameter, the hold speed must be adjusted in a way that it is a whole-number multiple of the $v_+$ and less or equal to the calculated value from the hold-strategy:

$$\underline{v}_h \leq x \cdot v_+ \tag{3.4}$$

with

$$x = \left\lfloor \frac{v_h}{v_+} \right\rfloor \tag{3.5}$$

and thus, the following speed-values will be used for the determination of the optimal driving strategy:

$$\underline{v}_h,\ \underline{v}_h + v_+,\ \underline{v}_h + 2 \cdot v_+,\ \underline{v}_h + 3 \cdot v_+,\ \ldots,\ \overline{v}$$

Assuming that the previously calculated hold speed is $v_h = 63.56\ \mathrm{km/h}$ and $v_+ = 10\ \mathrm{km/h}$, the modified hold-speed is set to $\underline{v}_h = 60\ \mathrm{km/h}$.

Similar to this parameter, a second parameter is available to set-up the granularity for finding the optimal coasting position: $s_+$. Here, too, the position cannot be found by a binary search algorithm because of the track gradient. Assume a hold position $s_h = 1\,000\ \mathrm{m}$ and a braking position $s_b = 5\,000\ \mathrm{m}$. As a consequence, there is a range of $4\,000\ \mathrm{m}$ to find the coasting position. By using $s_+$, designated positions are used for the calculation of the driving strategy:

$$s_h,\ s_h + s_+,\ s_h + 2 \cdot s_+,\ s_h + 3 \cdot s_+,\ \ldots,\ s_b$$

With decreasing $v_+$ and decreasing $s_+$ the algorithm will find better results in terms of energy consumption, but the execution time will increase. For online-systems, it is important to find a good trade-off between a good result, the execution time, and the practical feasibility.

## Algorithm

As already mentioned the hold-strategy forms the basis for the optimization. Initially the minimum energy consumption is set to $\underline{e} = \infty$. The algorithm starts by calculating the acceleration curve, starting with the initial speed $v_s$ until $\underline{v}_h$ is reached (see Equation 3.4). Next, the braking curve for this speed value is calculated in a way that the train arrives at its final position with its final speed $v_f$. If the final position is a stop (e.g. at a railway station), the final speed is set

to $v_f = 0\,\mathrm{km/h}$. The coasting position is set to the braking position and the train will hold its speed value until the coasting position is reached. As a result the first calculated driving strategy contains the acceleration phase, the speed-hold phase and the braking phase. Based on the following equations, the travel time $t$ and energy consumption $e$ are calculated.

$$t = t_a + t_h + t_c + t_b \tag{3.6}$$
$$e = e_a + e_h + e_c + e_b \tag{3.7}$$

If the following equations are valid, a new driving strategy with a minimum in terms of energy consumption has been found:

$$t \approx t^k_{\textcircled{\scriptsize}} \tag{3.8}$$
$$e < \underline{e} \tag{3.9}$$

If a new solution is found, the so far minimum energy consumption will be set to $\underline{e} = e$. Next, the coasting position will be decreased by $s_+$ and thus the new driving strategy starts with the acceleration-phase until the maximum speed value is reached, then there is the speed-hold-phase until the coasting position is reached, followed by a coasting-phase until the braking curve is reached. Finally the train will brake until it reaches its the final position. Again, if Equations 3.8 and 3.9 are valid, a new minimum is found.

The coasting position is again decreased by $s_+$ and the calculated values are checked. This procedure is repeated until at least one of the following termination conditions are valid:

- The coasting position is equal to the hold position (Equation 3.10) and thus the driving strategy contains the acceleration-phase, the coasting-phase, and the braking-phase. No further decrease of the coasting position is possible.

$$s_c = s_h \tag{3.10}$$

- The calculated travel time for the driving strategy is greater than the planned travel time (Equation 3.11). All further driving strategies with the same maximum speed value will have a higher travel time, because the hold phase decreases and the coasting phase increases. This assumption is valid only if the train decelerates or holds its speed while coasting.

$$t^k > t^k_{\textcircled{\scriptsize}} \tag{3.11}$$

- The speed value after the coasting phase is smaller than the final speed (Equation 3.12). This means that the train will not be able to reach its destination with the defined final speed value. For instance, a train would stop at the track before it reaches the railway station.

$$v_c \leq v_f \tag{3.12}$$

When there are no further valid solutions for the given hold speed, it is increased by $v_+$, the braking curve for the new speed is calculated and the braking position is determined. Then the procedure to find a new solution for the coasting positions starts. The algorithm is finished, if

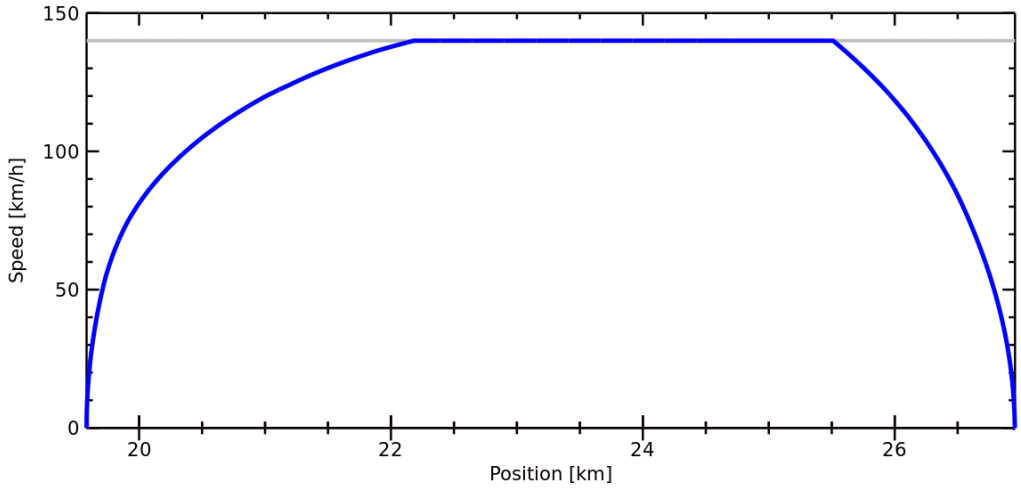



**Figure 3.5:** Interim result after the first step (Example 3.1)

- the maximum allowed speed is reached ($v_h = \overline{v}$) and the coasting position cannot be further decreased ($s_c = s_h$) or
- the maximum allowed speed is not reached ($v_h < \overline{v}$), but the calculated travel time is greater than the planned travel time ($t^k > t^k_{\clock}$) and the coasting position cannot be further decreased ($s_c = s_h$).

An optimal driving strategy is found if appropriate speed values and positions are found by the algorithm. Otherwise, either the planned travel time $t^k_{\clock}$ cannot be achieved or the algorithm parameters must be adjusted to a finer grain (e.g. $v_+ = 5\ \mathrm{km/h}$ instead of $v_+ = 20\ \mathrm{km/h}$ or $s_+ = 10\ \mathrm{m}$ instead of $s_+ = 1000\ \mathrm{m}$).

To avoid several calculations of the acceleration curve and the braking curve, the calculation is done only once and stored in an array, including the energy consumption and the travel time for each positions. Thus, the required travel time and energy consumption can be reused for different driving strategies. A similar approach is employed for the speed-hold-phase, with the restriction that the calculation must be done for each hold speed, but only once. As a result, the travel time and the energy consumption, again, can be reused for each combination of the speed-hold-phase and the coasting phase.

Figures 3.5–3.8 show the interim result for Example 3.1 after the first, the second, the third, and the thirtieth step of the algorithm, when using $s_+ = 100\ \mathrm{m}$. Figures 3.9, 3.10, and 3.11 show the results of the optimal driving strategy, where the red line illustrates the hold strategy and the blue one the optimal driving strategy. The results show the distance-speed diagram, the accumulated energy consumption, and the distance-time diagram.

Algorithm 3.2 describes the algorithm to find the optimal driving strategy.

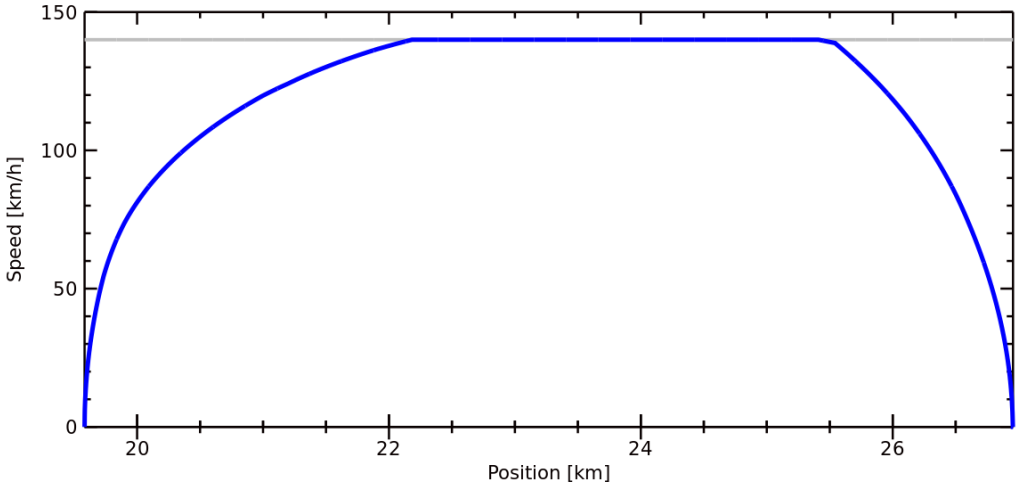



**Figure 3.6:** Interim result after the second step (Example 3.1)

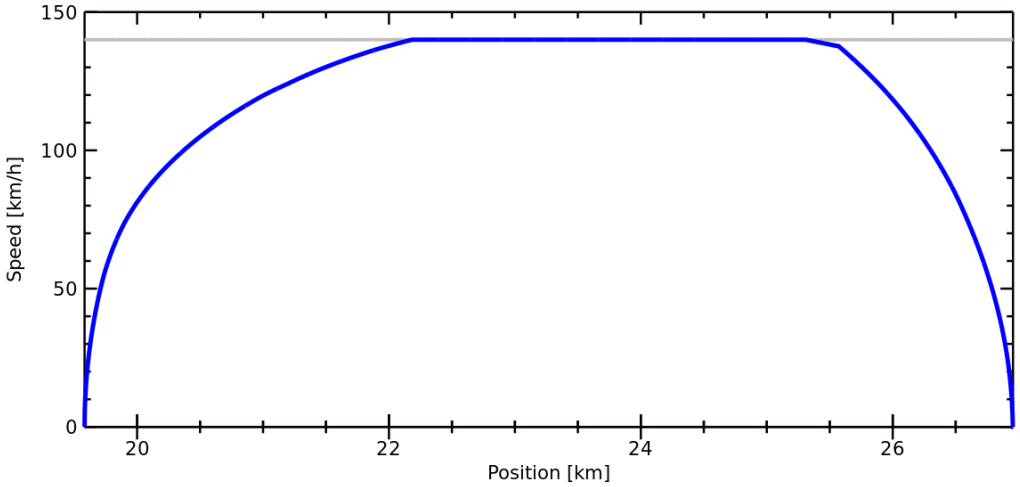



**Figure 3.7:** Interim result after the third step (Example 3.1)

**Example 3.2.** *In this example, a train using the hold-strategy consumes* $42.386\ \mathrm{kWh}$ *and when using the optimal driving strategy, it consumes* $37.908\ \mathrm{kWh}$*. In comparison to the hold strategy, the optimal driving strategy will save* $4.478\ \mathrm{kWh}$*, which is about* $11.8\ \%$*.*

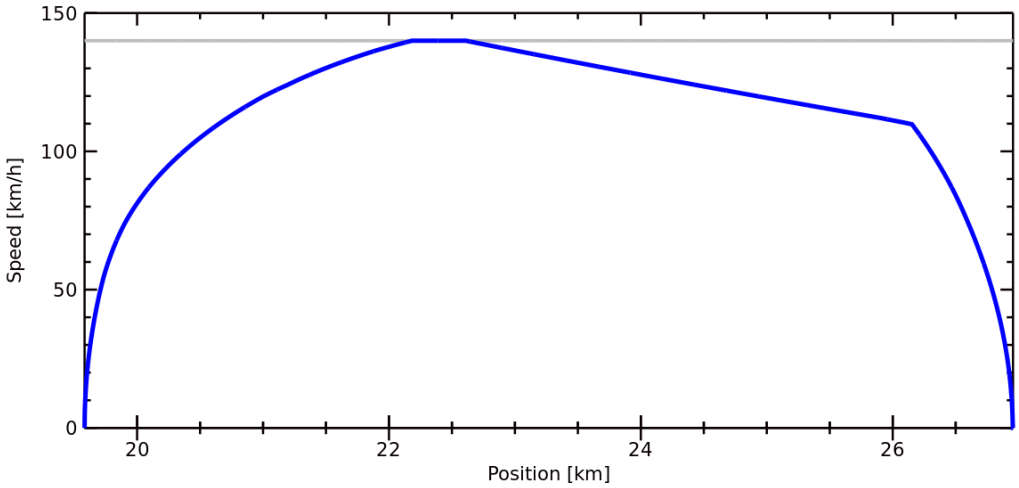



**Figure 3.8:** Interim result after the 30th step (Example 3.1)

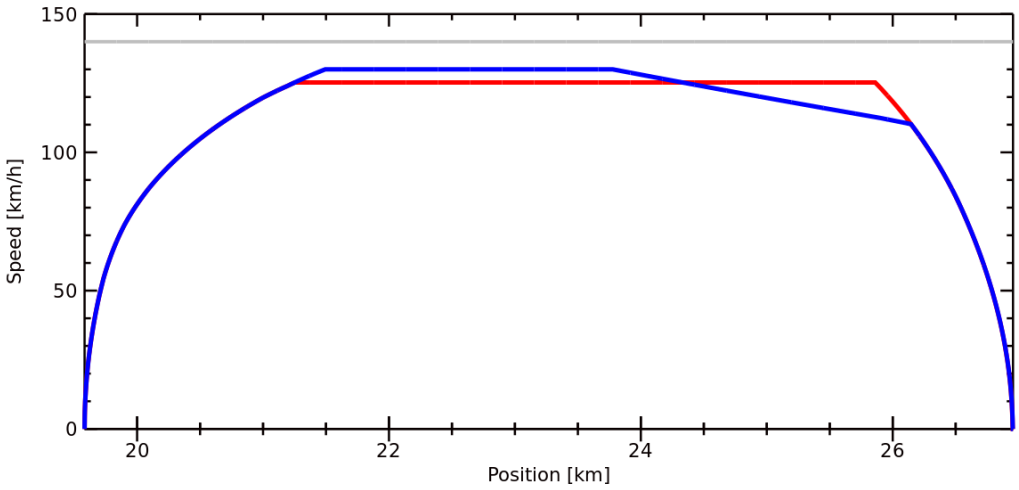



**Figure 3.9:** Optimal driving strategy (Example 3.1)

## 3.3 Multi-regime algorithm

In the previous section, the algorithm to determine the optimal driving strategy for a track with only one regime was introduced. This section explains how the algorithm is extended to handle tracks with several regimes.

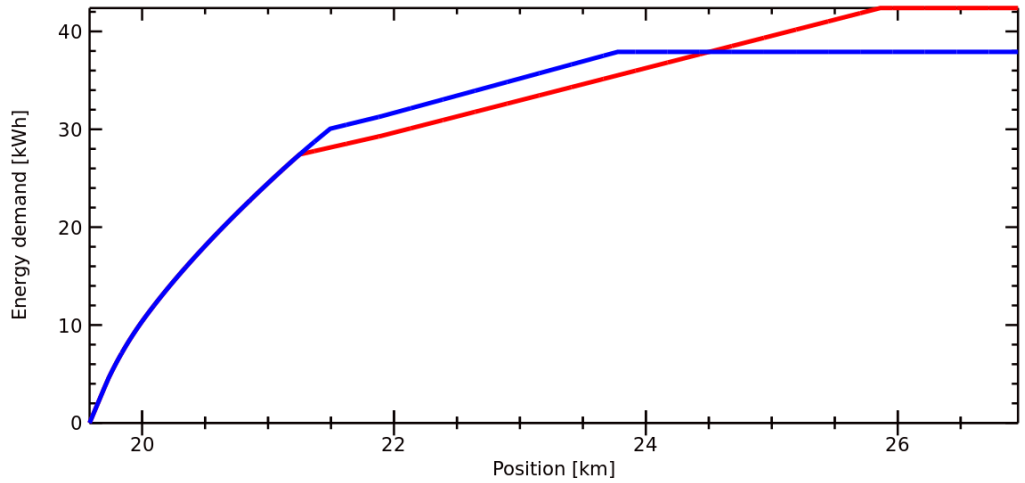



**Figure 3.10:** Energy demand for the optimal driving strategy (Example 3.1)

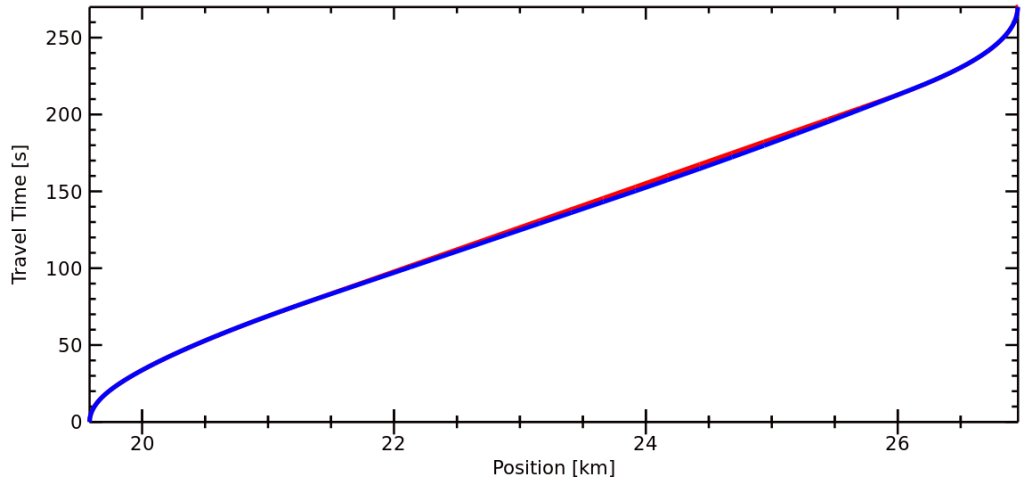



**Figure 3.11:** Distance-time diagram for the optimal driving strategy (Example 3.1)

### Algorithm

The first step of the algorithm is the same as in the previous section for the simple algorithm: A hold-strategy must be found. Due to speed restrictions on the track, the train may not drive with constant speed along the whole track. Assuming a hold-strategy with $v_h = 80\ \mathrm{km/h}$ and a planned travel time. Now assume a temporary speed restriction of $\overline{v} = 40\ \mathrm{km/h}$ in the middle of the track section. As a consequence, the train has to adapt its driving strategy, to ensure that

**Algorithm 3.2:** $OptimalDrivingStrategy$ (Algorithm to find the optimal driving strategy)

**input** : $s_s$, $s_f$, $t^k_{\textcircled{\scriptsize clock}}$, $\overline{v}$
**output**: $s_h$, $s_c$, $s_b$, $v_h$, $v_c$, $v_b$, $e_c$

```
1  not_finished ← true;
2  v_h ← v̄;
3  e ← ∞;
4  while not_finished do
5      (s_h,t_a,e_a) ← Accelerate (s_s,v_h) ;
6      (s_b,t_b) ← GetBrakingPosition (s_f v_h) ;
7      s_c ←s_h;
8      not_finished_coast ← true;
9      while not_finished_coast do
10         (t_h,e_h) ← SpeedHold (s_h,s_c,v_h) ;
11         (t_c) ← Coast (s_c,s_b,v_c) ;
12         if t > t^k_clock and s_c = s_b then
13             not_finished ← false;
14             not_finished_coast ← false;
15         else
16             if s_c = s_h or v_c = 0 or t > t^k_clock then
17                 not_finished_coast ← false;
18             else
19                 t ← t_a + t_h + t_c + t_b;
20                 e ← e_a + e_h + e_b;
21                 if t ≈ t^k_clock and e < e then
22                     e ← e;
23                 end
24                 s_c ← s_c - s_+;
25                 if s_c < s_h then
26                     s_c ← s_h;
27                 end
28             end
29         end
30     end
31     v_h ← v_h - v_+;
32     if v_h = 0 then
33         not_finished ← false;
34     end
35 end
```

the maximum speed of $40\ \mathrm{km/h}$ would not be exceeded.

In general, for each change of the maximum allowed track speed, a new regime is introduced. The algorithm to determine the hold-strategy is based on the algorithm of Section 3.2, with some slight modifications:

- The maximum allowed speed must be determined for each regime.

- The overall travel time is the sum of the travel time values of each regime.

As a consequence, each regime may consist of acceleration-phase, speed-hold-phase, coasting-phase, and braking-phase and consists of the following values, which will be used for determining of the optimal driving strategy and for determining of the overall result (Chapter 5).

**Definition 3.6** (Planned travel time)**.** *The planned travel time for regime $i$ is denoted by $t_{i\clock}$.*

**Definition 3.7** (Output speed)**.** *The speed value when leaving track section $i$ is called output speed, denoted by $v_{\boxed{i}\rightarrow}$. The highest and lowest possible output speed values (with respect to the planned travel time $t_{i\clock}$) when leaving a track section $i$ are $\overline{v}_{\boxed{i}\rightarrow}$ and $\underline{v}_{\boxed{i}\rightarrow}$, respectively.*

**Definition 3.8** (Input speed)**.** *The speed value when entering track section $i$ is called input speed, denoted by $v_{\rightarrow\boxed{i}}$. The highest and lowest possible input speed values (with respect to the planned travel time $t_{i\clock}$) when entering a track section $i$ are denoted by $\overline{v}_{\rightarrow\boxed{i}}$ and $\underline{v}_{\rightarrow\boxed{i}}$, respectively. Obviously $v_{\rightarrow\boxed{i}} = v_{\boxed{i\text{-}1}\rightarrow}$ for all $1 < i \leq n$, where $n$ is the number of track sections*[3]*. The first track section of a track or sub-track, starting at a railway station has an input speed value of* $0\ \mathrm{km/h}$ *and the last section of a journey, ending at a station has an output speed of* $0\ \mathrm{km/h}$.

After the hold-strategy is calculated, the optimal driving strategy is determined for each regime based on the values defined above. In particular, the optimal driving strategy for the first regime will be calculated by using $v_{\rightarrow\boxed{1}}$ as input speed, $v_{\boxed{1}\rightarrow}$ as output speed, and $t_{1\clock}$ as planned travel time. The driving strategy of each regime may be optimized separately, when the hold speed $v_h$ of a regime is greater than the input speed of the subsequent regime.

**Example 3.3.** *In this example, the train data from the previous section are used (Table 3.1). The planned travel time is modified to guarantee that the train can arrive at its destination in time. The track data are nearly the same, except a temporary speed restriction of* $90\ \mathrm{km/h}$*, starting at position* $23\,000\ \mathrm{m}$ *and ending at* $23\,500\ \mathrm{m}$*. As a result, the train cannot drive along the whole track with constant speed. Due to the speed restriction the track is divided into three regimes. The resulting driving strategy for this example is illustrated in Figure 3.12.*

Speed restrictions, which don't have an effect on the hold-strategy are ignored. In particular, the regimes will be merged. The maximum allowed speed for the new regime is the minimum of the maximum allowed speed values of the merged regimes:

$$\overline{v}_{i\ldots j} = \min\left(\overline{v}_i, \overline{v}_{i+1}, \ldots, \overline{v}_j\right) \tag{3.13}$$

[3]Track section $i$ is the successor of section $i-1$ on the route.

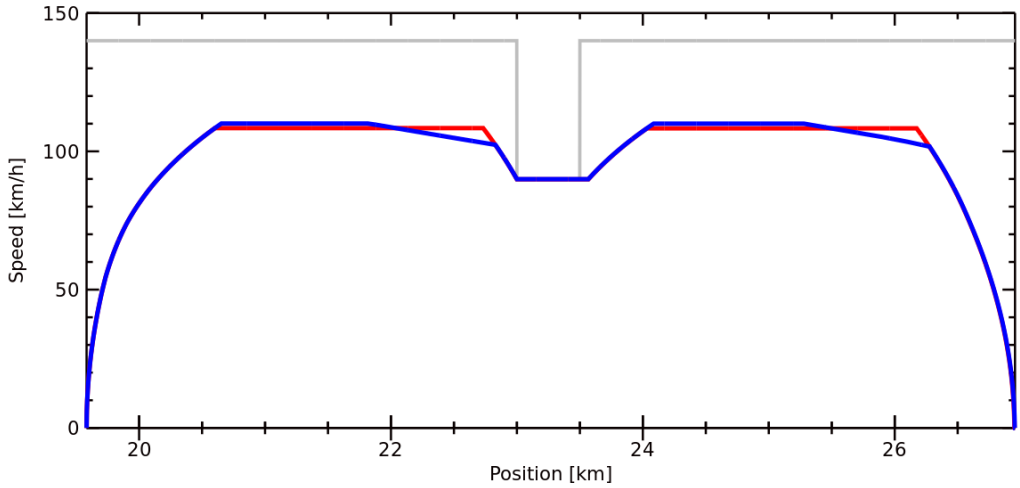



**Figure 3.12:** Optimal driving strategy with multiple regimes (Example 3.3)

The planned travel time of the new regime is calculated by summing up the travel times of the involved regimes. The input and output speed values of the new regime are defined as:

$$v_{\rightarrow\boxed{\mathrm{i}\ldots\mathrm{j}}} = v_{\rightarrow\boxed{\mathrm{i}}} \tag{3.14}$$

$$v_{\boxed{\mathrm{i}\ldots\mathrm{j}}\rightarrow} = v_{\boxed{\mathrm{j}}\rightarrow} \tag{3.15}$$

**Example 3.4.** *The resulting driving strategy for another example is illustrated in Figure 3.13, where the temporary speed restriction at the beginning of the track has no effect on the hold-strategy. The first regime (*$19\,583$ m–$20\,000$ m*) with maximum speed* $130$ km/h *and the second regime (*$20\,000$ m–$23\,000$ m*) with maximum speed* $140$ km/h *are merged to a new regime with a maximum speed value of* $130$ km/h*, which starts at position* $19\,583$ m *and ends at position* $23\,000$ m.

**Example 3.5.** *This example uses a track with five regimes and a planned travel time of* $t^k_{\textcircled{\scriptsize 1}} = 330$ s*. The resulting driving strategy is illustrated in Figure 3.14. About* $10\,\%$ *of the energy consumption can be saved due to the optimization of the third, the fourth, and the fifth regime. If a recovery factor of* $r = 0.85$ *is used, the driving strategy stays the same and the potential savings are about* $2.8\,\%$*. In general, the recovery factor may have an effect on the calculated driving strategy.*

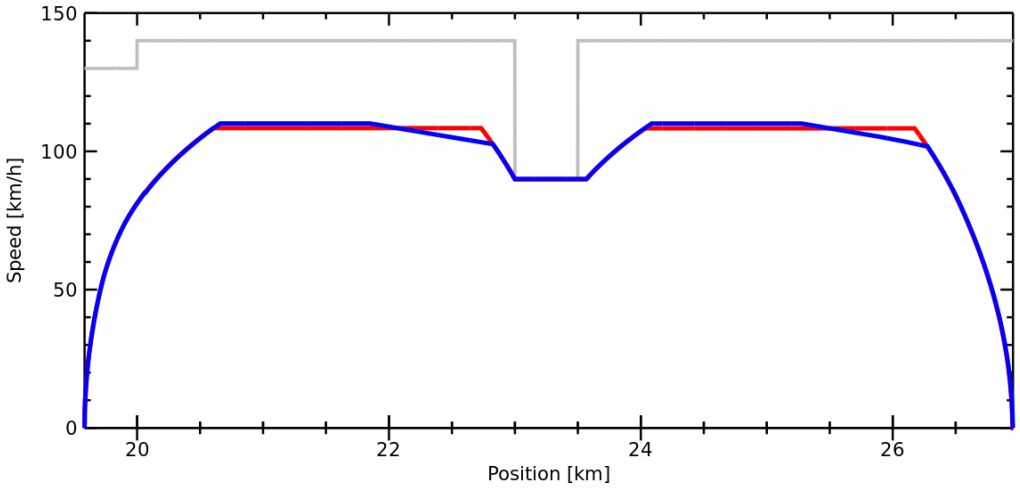



**Figure 3.13:** Optimal driving strategy with merged regimes (Example 3.4)

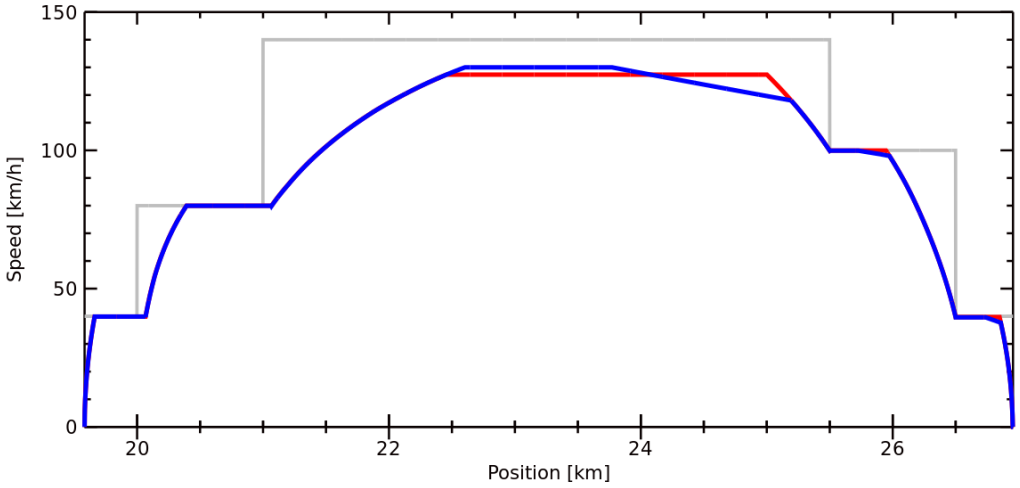



**Figure 3.14:** Optimal driving strategy with multiple regimes (Example 3.5)

## 3.4 Algorithm-Parameters

As already mentioned, there are several parameters to configure the algorithm:

- Accuracy vs. computation time: There exist two parameters $v_+$ and $s_+$ which can be used to configure the accuracy of the results. As a consequence, the computation time depends on these parameters.

- Driving comfort: The algorithm might calculate a driving strategy which consists of a very short phase, e.g. only a few seconds. To avoid such short phases, which are not feasible for the train driver, four parameters exist to configure the minimum time of each phase:
  - $\underline{t_a}$: Minimum time of the acceleration phase.
  - $\underline{t_h}$: Minimum time of the speed-hold phase.
  - $\underline{t_c}$: Minimum time of the coasting phase.
  - $\underline{t_b}$: Minimum time of the braking phase:

As a consequence, only results are taken into account if

$$t_a >= \underline{t_a} \tag{3.16}$$
$$t_h >= \underline{t_h} \tag{3.17}$$
$$t_c >= \underline{t_c} \tag{3.18}$$
$$t_b >= \underline{t_b} \tag{3.19}$$

but only if the corresponding minimum time is set to a value greater than $0$. Some examples which use different algorithm parameters can be found in Section 3.6.

## 3.5 Optimal Driving Strategy

The algorithm in the previous sections calculate a driving strategy for a number of regimes, depending on the maximum allowed speed on the track. In general, there might exist a driving strategy which is better in terms of energy consumption but might not be feasible for practical purposes. Now, the driving strategy is calculated in another way as previously described. The algorithm tries to find a solution by varying the hold-position $s_h$, the coasting position $s_c$, and the braking position $s_b$. To illustrate the potential savings and an optimal driving strategy with several regimes, the following example is used.

**Example 3.6.** *This example uses a very short track with a length of* $32\ \mathrm{m}$ *to illustrate several solutions with different numbers of regimes. It starts at position* $s_s = 1\ 000\ \mathrm{m}$ *and ends at position* $s_f = 1\ 032\ \mathrm{m}$*. The inclination on the complete track is* $0\ ‰$*. The planned travel time is set to* $t_{\clock} = 14\ \mathrm{s}$*. The train and algorithm parameters can be found in Table 3.3.*

*First, the driving strategy is calculated as described in the previous chapters. In particular, only one regime is used. The resulting driving strategy is shown in Figure 3.15. The train starts its journey at position* $s_s = 1\,000\ \mathrm{m}$ *and accelerates until* $s_h = 1\,008\ \mathrm{m}$ *is reached. Then there is a speed-hold-phase until* $s_c = 1\ 019\ \mathrm{m}$*, followed by the braking phase until the final position is reached. The complete energy consumption for this driving strategy is* $e = 0.2536469400\ kWh$*, calculated within* $10\ \mathrm{ms}$*.*

*Next, the track is divided into two regimes, where each regime has a length of* $16\ \mathrm{m}$*. As a result, it might be possible to have each driving type twice within the track. The now created driving strategy consists of the following parts:*

- *Regime 1 (*$1\ 000\ \mathrm{m} - -1\ 016\ \mathrm{m}$*:*

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $m_T$ | 46 000 t | $m_W$ | 74 000 t |
| $n$ | 2 | $r$ | 0.00 |
| $F(v)$ | see Ex. 2.1 | $B(v)$ | see Ex. 2.2 |
| $l$ | 67 m | $f_L$ | 3.3 |
| $k_{St_1}$ | $0.03\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $k_{Sa_1}$ | $0.0025\ \mathrm{s/m}$ |
| $k_{Sa_2}$ | $0.00696\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $\Delta v$ | $4.17\ \mathrm{m/s}$ |
| $\rho$ | 6 | $s_+$ | 1 m |

**Table 3.3:** Train and algorithm parameters (Example 3.6)

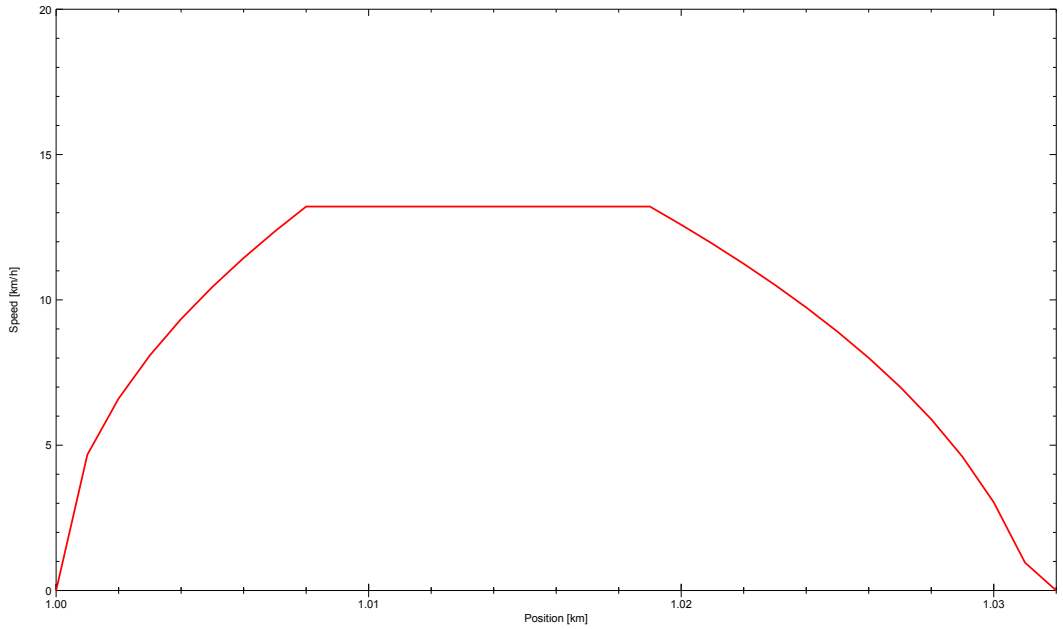



**Figure 3.15:** Resulting driving strategy with one regime (Example 3.6)

  - *Accelerating to* 1 007 m.
  - *Coasting to* 1 016 m.

- *Regime 2 (*1 016 m $-$ $-$1 032 m*:*
  - *Hold the current speed to* 1 018 m.
  - *Coasting to* 1 021 m.
  - *Braking to* 1 032 m.

*As a consequence, the speed-hold-phase and the coasting phase from the driving strategy in Figure 3.15 is divided into a coasting phase, followed by a speed-hold phase and again a*

*coasting-phase. The energy consumption for this driving strategy is* $e = 0.2155773640\ kWh$*, calculated within* 13 ms.

*Again the number of regimes can be increased. As the algorithm is designed to calculate the positions in* m *and thus, the maximum number of regimes is equal to the track-length in* m. *Table 3.4 gives an overview of the results, using different numbers of regimes.*

| Number of Regimes | Energy consumption [kWh] | Execution time [s] |
|---|---|---|
| 1 | 0.2536469400 | 0.010 |
| 2 | 0.2155773640 | 0.013 |
| 4 | 0.2155773640 | 0.030 |
| 8 | 0.2155770212 | 0.180 |
| 16 | 0.2155767977 | 0.800 |

**Table 3.4:** Resulting values (Example 3.8)

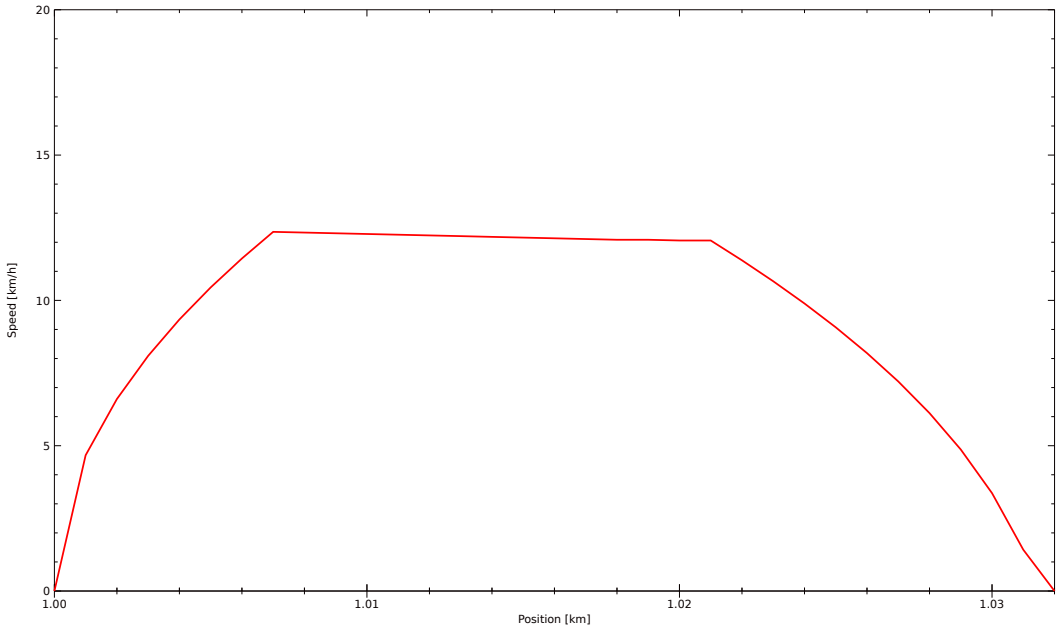



**Figure 3.16:** Resulting driving strategy with 16 regimes (Example 3.6)

*It can be seen that the energy saving slowly decreases when the number of regimes increases but the execution time increases exponentially. The resulting driving strategy for* 16 *regimes is illustrated in Figure 3.16 and consists of the following parts:*

- *Acceleration-phase from* 1000 m *to* 1008 m

- *Coasting-phase from* $1008\ \mathrm{m}$ *to* $1019\ \mathrm{m}$
- *Speed-hold-phase from* $1019\ \mathrm{m}$ *to* $1020\ \mathrm{m}$
- *Coasting-phase from* $1020\ \mathrm{m}$ *to* $1021\ \mathrm{m}$
- *Speed-hold-phase from* $1021\ \mathrm{m}$ *to* $1022\ \mathrm{m}$
- *Braking-phase from* $1022\ \mathrm{m}$ *to* $1032\ \mathrm{m}$

**Example 3.7.** *To illustrate the behavior of the algorithm and the potential of saving traction energy this example will use several changes of the track inclination, given in Table 3.5. The resulting driving strategy for* $1$ *regime and for* $16$ *regimes are illustrated in Figure 3.17 and 3.18, respectively. The traction energy can be decreased from* $0.1686721295\ \mathrm{kWh}$ *to* $0.1487099230\ \mathrm{kWh}$*, which conforms a saving of about* $11.8\ \%$*.*

| **From** [m] | **To** [m] | **Grad.** [‰] | **From** [m] | **To** [m] | **Grad.** [‰] |
|---|---|---|---|---|---|
| 1 000 | 1 002 | 4 | 1 002 | 1 004 | 8 |
| 1 004 | 1 006 | −6 | 1 006 | 1 008 | 7 |
| 1 008 | 1 010 | 6 | 1 010 | 1 012 | −8 |
| 1 012 | 1 014 | 10 | 1 014 | 1 016 | 6 |
| 1 016 | 1 018 | 0 | 1 018 | 1 020 | −4 |
| 1 020 | 1 022 | −3 | 1 022 | 1 024 | 3 |
| 1 024 | 1 026 | 7 | 1 026 | 1 028 | −4 |
| 1 028 | 1 030 | 2 | 1 030 | 032 | −5 |

**Table 3.5:** Track inclination (Example 3.7)

A driving strategy like that in the previous examples is definitely not feasible in the field due to the high number of changes of the driving type. In general, for tracks with much inclination changes, a driving strategy with several regimes can save more traction energy than on a track with constant gradient.

Due to the computation time and the driving and passenger comfort, the algorithm from the previous sections will be used further on. The algorithm of this section is given only for theoretical reasons.

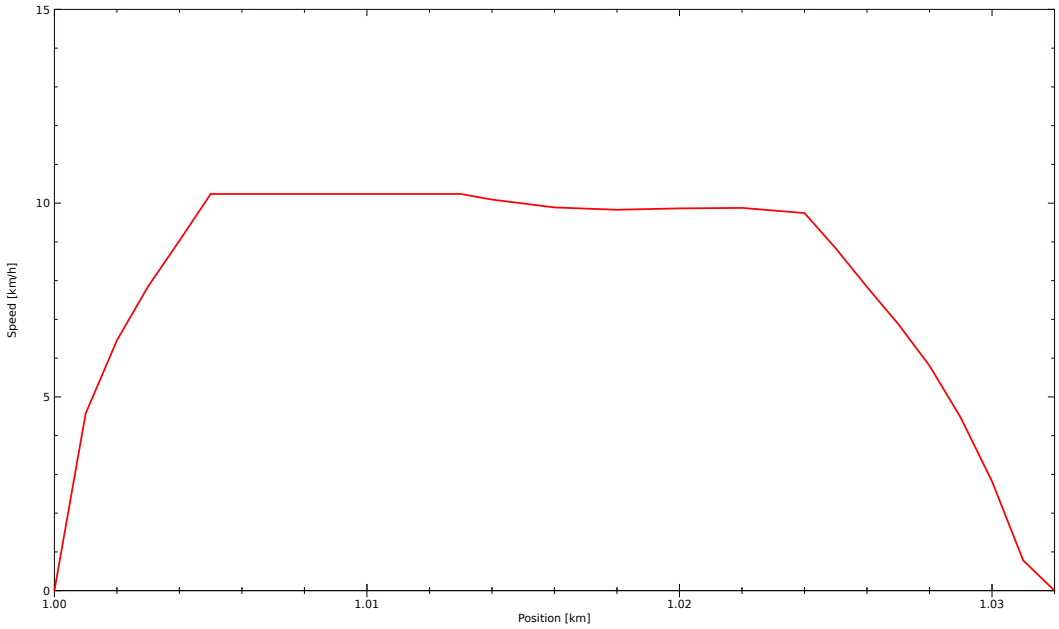



**Figure 3.17:** Resulting driving strategy with one regime (Example 3.7)

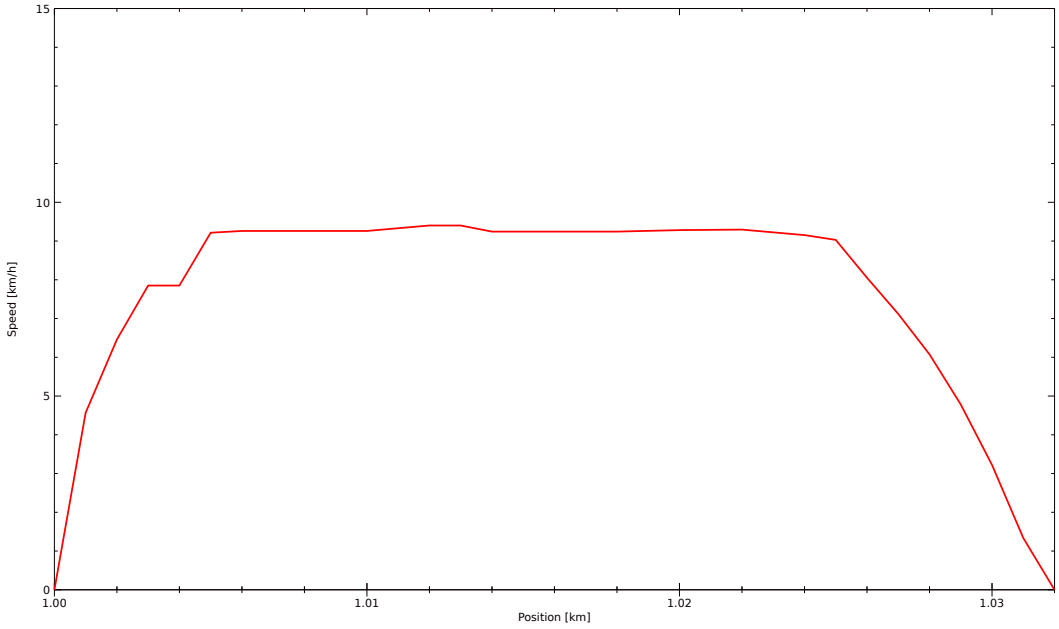



**Figure 3.18:** Resulting driving strategy with 16 regimes (Example 3.7)

## 3.6 Further Examples

This section contains some examples to illustrate the operating principle of the algorithm and the effects when changing train parameters, track parameters, and algorithm parameters.

| **From** [m] | **To** [m] | **Grad.** [‰] | **From** [m] | **To** [m] | **Grad.** [‰] |
|---|---|---|---|---|---|
| 400 | 500 | 2 | 500 | 700 | 0 |
| 700 | 900 | −16 | 900 | 1 100 | −5 |
| 1 100 | 1 500 | −16 | 1 500 | 1 600 | −13 |
| 1 600 | 1 800 | −15 | 1 800 | 2 100 | 0 |
| 2 100 | 2 500 | −2 | 2 500 | 2 600 | −6 |
| 2 600 | 2 900 | −2 | 2 900 | 3 000 | 17 |
| 3 000 | 3 200 | 16 | 3 200 | 3 300 | 1 |
| 3 300 | 3 700 | 0 | 3 700 | 3 800 | 1 |
| 3 800 | 4 300 | 0 | 4 300 | 4 500 | 9 |
| 4 500 | 4 800 | 10 | 4 800 | 4 900 | 3 |
| 4 900 | 5 100 | 0 | 5 100 | 5 300 | 1 |
| 5 300 | 5 600 | 0 | 5 600 | 5 800 | 16 |
| 5 800 | 6 000 | 17 | 6 000 | 6 500 | 16 |
| 6 500 | 6 700 | 11 | 6 700 | 6 800 | 13 |
| 6 800 | 7 100 | 15 | 7 100 | 7 500 | 11 |
| 7 500 | 7 900 | 15 | 7 900 | 8 200 | 18 |
| 8 200 | 8 400 | 16 | 8 400 | 8 500 | 5 |
| 8 500 | 8 700 | 10 | 8 700 | 8 900 | 1 |
| 8 900 | 9 100 | 15 | 9 100 | 9 300 | 12 |
| 9 300 | 9 700 | 0 | | | |

**Table 3.6:** Track inclination (Example 3.8)

**Example 3.8.** *This example uses the train, track, and algorithm parameters shown Table 3.7. The track gradient is given in Table 3.6 and the planned travel time is defined as* $t_{⏲} = 150\,\mathrm{s}$. *The resulting driving strategy is illustrated in Figure 3.19 and the resulting values for travel time, energy consumption, and the positions for changing the driving type are given in Table 3.8.*

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $m_T$ | 51 000 t | $m_W$ | 79 000 t |
| $n$ | 2 | $r$ | 0.85 |
| $F(v)$ | see Ex. 2.1 | $B(v)$ | see Ex. 2.2 |
| $l$ | 67 m | $f_L$ | 3.3 |
| $k_{St_1}$ | $0.03\ \mathrm{kg \cdot s^2/m^2}$ | $k_{Sa_1}$ | 0.0025 s/m |
| $k_{Sa_2}$ | $0.00696\ \mathrm{kg \cdot s^2/m^2}$ | $\Delta v$ | 4.17 m/s |
| $\rho$ | 9.0 | | |
| $s_s$ | 2 500 m | $s_f$ | 3 850 m |
| $v_s$ | 0 km/h | $v_f$ | 0 km/h |
| $v_+$ | 5 km/h | $s_+$ | 10 m |

**Table 3.7:** Train, track, and algorithm parameters (Example 3.8)

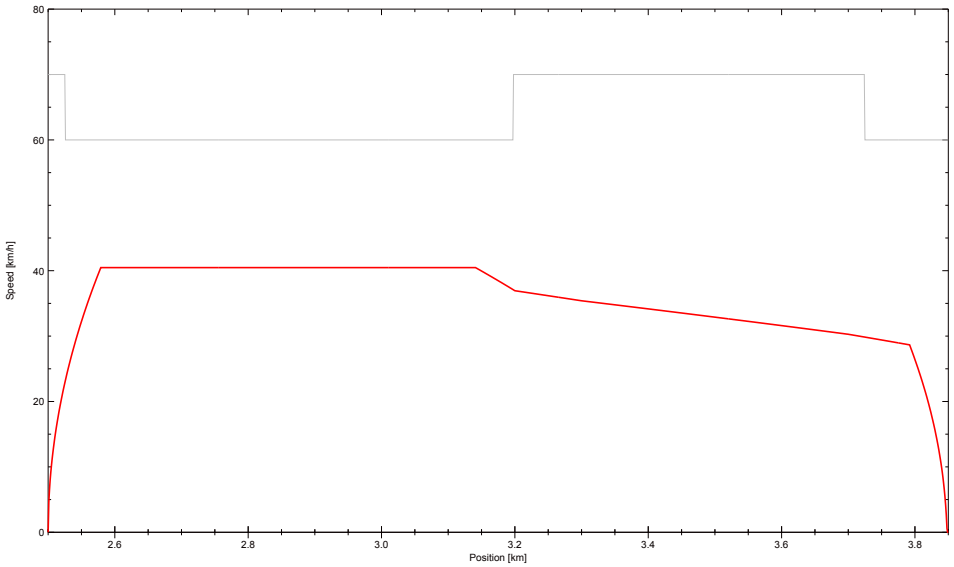



**Figure 3.19:** Resulting driving strategy (Example 3.8)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 2 500 m | $s_h$ | 2 579 m |
| $s_c$ | 3 141 m | $s_c$ | 3 792 m |
| $s_f$ | 3 850 m | | |
| $t$ | 149.3 s | $e$ | 3.279 kWh |

**Table 3.8:** Resulting values (Example 3.8)

**Example 3.9.** *Now, the same values as in Example 3.8 are used, except the rotating mass factor* $\rho = 6.0$ *and the wind speed* $\Delta v = 0.0\,\mathrm{m/s}$*. The resulting driving strategy is illustrated in Figure 3.20 and the resulting values are given in Table 3.9. It can be seen that changing the two parameters results in a new driving strategy and thus, in a lower energy consumption. Due to the reduced rotating mass factor and the reduced wind speed the complete resistance is lower than in the previous example which results in a longer coasting phase.*

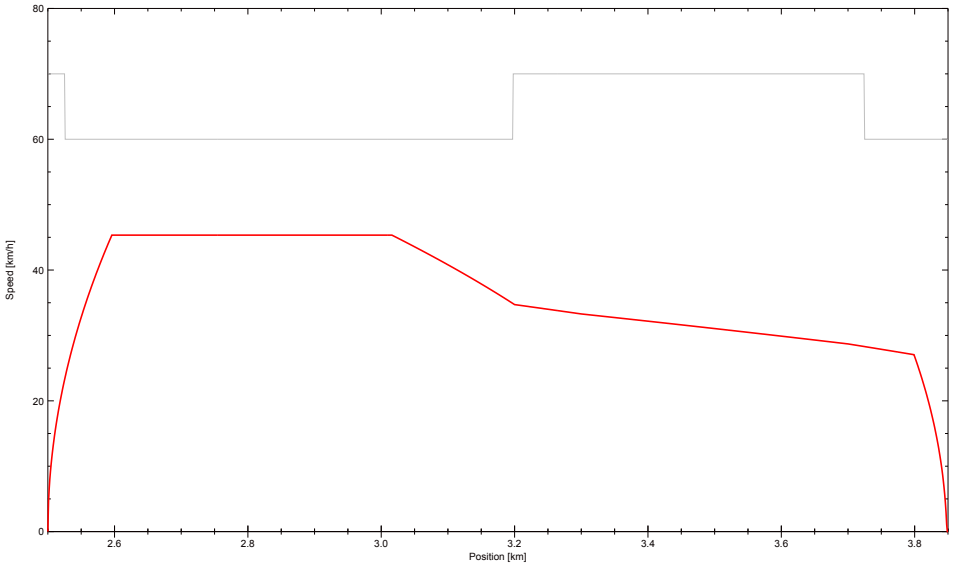



**Figure 3.20:** Resulting driving strategy (Example 3.9)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 2 500 m | $s_h$ | 2 596 m |
| $s_c$ | 3 016 m | $s_c$ | 3 799 m |
| $s_f$ | 3 850 m | | |
| $t$ | 148.3 s | $e$ | 2.984 kWh |

**Table 3.9:** Resulting values (Example 3.9)

**Example 3.10.** *This example uses the same parameters as the previous one except the weight of the train (*$m_T = 46\,000\,\mathrm{kg}$*) and the wagons (*$m_W = 74\,000\,\mathrm{kg}$*). The optimal driving strategy is illustrated in Figure 3.21 and the resulting values are given in Table 3.10. It can be seen that the reduced weight has an effect on the driving strategy. On one hand, the train will arrive at its hold position earlier due to a steeper acceleration curve and the other hand, the hold phase is longer than in the previous example.*

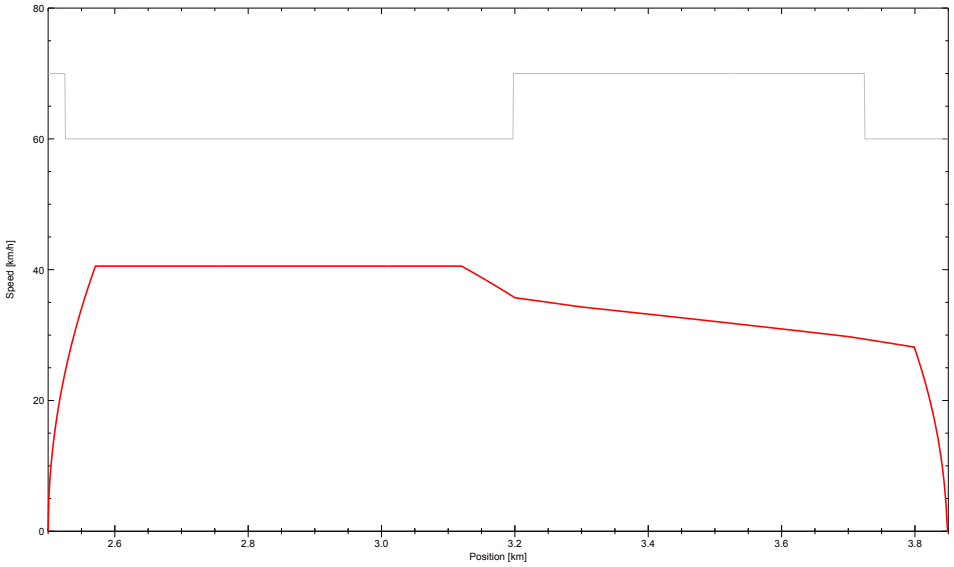



**Figure 3.21:** Resulting driving strategy (Example 3.10)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 2 500 m | $s_h$ | 2 571 m |
| $s_c$ | 3 120 m | $s_c$ | 3 799 m |
| $s_f$ | 3 850 m | | |
| $t$ | 149.0 s | $e$ | 2.762 kWh |

**Table 3.10:** Resulting values (Example 3.10)

**Example 3.11.** *For this example, the same train and track parameters as in Example 3.10 are used. Only the algorithm parameter $s_+$ is changed from* 10 m *to* 1 m*. The resulting driving strategy is illustrated in Figure 3.22 and the resulting positions are given in Table 3.11. It can be seen that there is only a small saving in energy consumption but the driving strategy is quite different, compared to Example 3.10.*

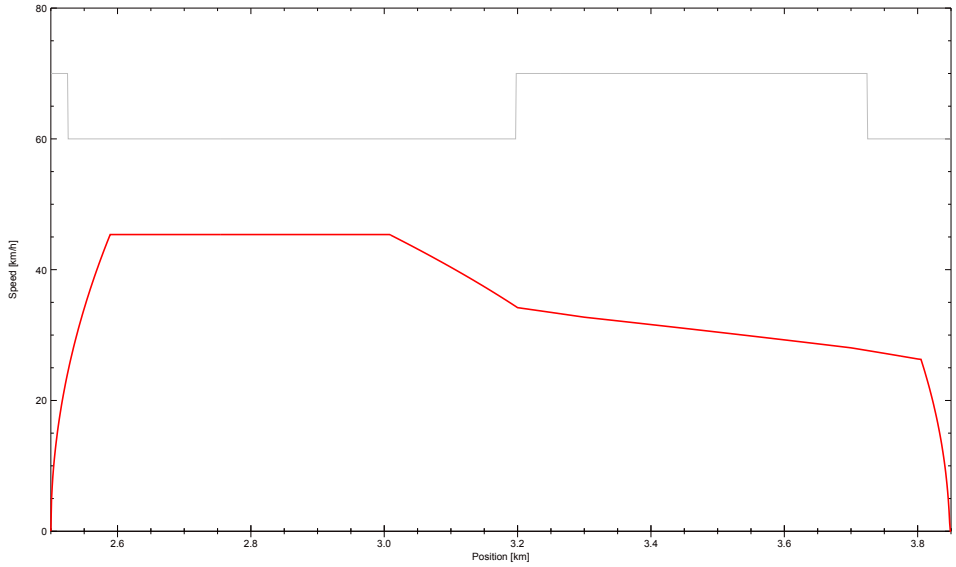



**Figure 3.22:** Resulting driving strategy (Example 3.11)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 2 500 m | $s_h$ | 2 589 m |
| $s_c$ | 3 008 m | $s_c$ | 3 805 m |
| $s_f$ | 3 850 m | | |
| $t$ | 148.6 s | $e$ | 2.712 kWh |

**Table 3.11:** Resulting values (Example 3.11)

**Example 3.12.** *The next example uses the same parameters as Example 3.10 but now the recovery factor is set to* $r = 0.0$*. As a result the energy consumption is much higher and the driving strategy is quite different (Figure 3.23, Table 3.12).*

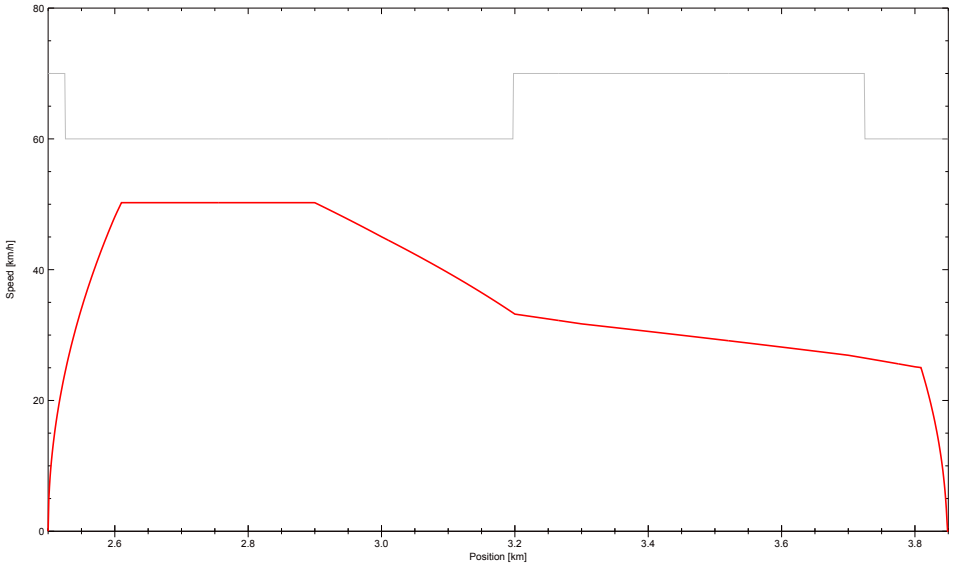



**Figure 3.23:** Resulting driving strategy (Example 3.12)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 2 500 m | $s_h$ | 2 610 m |
| $s_c$ | 2 898 m | $s_c$ | 3 809 m |
| $s_f$ | 3 850 m | | |
| $t$ | 148.9 s | $e$ | 3.525 kWh |

**Table 3.12:** Resulting values (Example 3.12)

**Example 3.13.** *This example shows the impact of changing the minimum time of the speed-hold phase to* $\underline{t_h} = 30$ s*. As a consequence, the speed-hold phase will be longer* ($t_h \approx 33$ s*) than in the previous example* ($t_h \approx 20$ s*) and as a consequence, the energy consumption is increased (Figure 3.24, Table 3.13).*

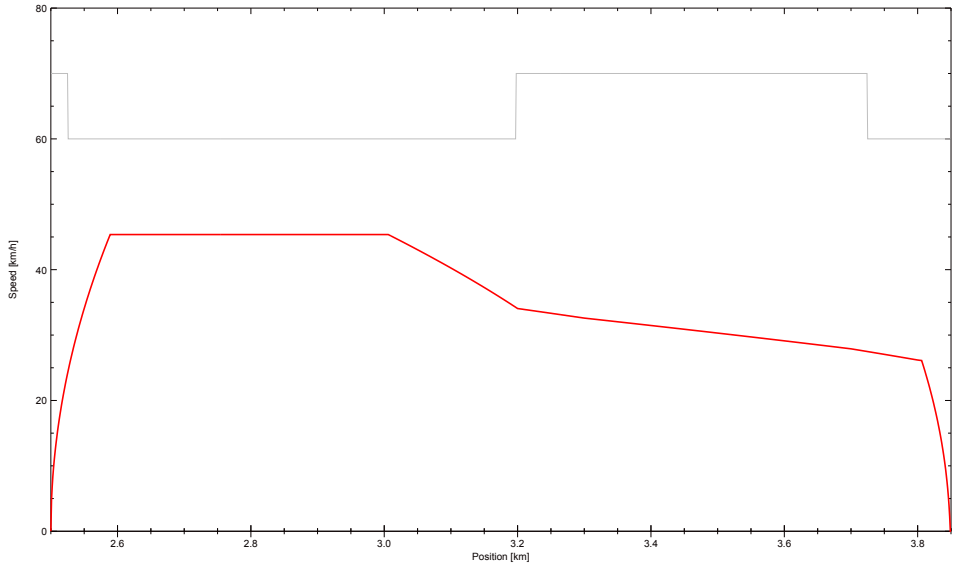



**Figure 3.24:** Resulting driving strategy (Example 3.13)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $s_s$ | 2 500 m | $s_h$ | 2 589 m |
| $s_c$ | 2 3006 m | $s_c$ | 3 806 m |
| $s_f$ | 3 850 m | | |
| $t$ | 148.5 s | $e$ | 3.587 kWh |

**Table 3.13:** Resulting values (Example 3.13)

Table 3.14 gives an overview of the results of the previous examples where all of them use the same track, but different train or algorithm parameters. The impacts can be seen in different values of the overall energy consumption of each example.

| Example | Energy consumption [kWh] | Example | Energy consumption [kWh] |
|---|---|---|---|
| Example 3.8 | 3.279 | Example 3.9 | 2.984 |
| Example 3.10 | 2.762 | Example 3.11 | 2.712 |
| Example 3.12 | 3.525 | Example 3.13 | 3.587 |

**Table 3.14:** Resulting energy consumption (Example 3.8 – 3.13)

**Example 3.14.** *This example shows the results of calculating the optimal driving strategy for the* ÖBB S45 *from* Wien Breitensee *to* Wien Heiligenstadt*. For the maximum allowed track speed and for the inclination, data from [28] are used. The train characteristics and the algorithm parameters can be found in Table 3.15. The schedule is given in Table 3.16.*

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $m_T$ | $51\,000$ t | $m_W$ | $79\,000$ t |
| $n$ | $2$ | $r$ | $0.85$ |
| $F(v)$ | see Ex. 2.1 | $B(v)$ | see Ex. 2.2 |
| $l$ | $67$ m | $f_L$ | $3.3$ |
| $k_{St_1}$ | $0.03\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $k_{Sa_1}$ | $0.0025\ \mathrm{s/m}$ |
| $k_{Sa_2}$ | $0.00696\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $\Delta v$ | $0.00\ \mathrm{m/s}$ |
| $\rho$ | $6.0$ | | |
| $v_+$ | $5$ km/h | $s_+$ | $1$ m |

**Table 3.15:** Train and algorithm parameters (Example 3.14)

| Railway station | Arrival | Departure | Travel time |
|---|---|---|---|
| Wien Breitensee | | 12:05:30 | |
| Wien Ottakring | 12:07:30 | 12:08:00 | 00:02:00 |
| Wien Hernals | 12:10:00 | 12:10:30 | 00:02:00 |
| Wien Gersthof | 12:12:30 | 12:13:00 | 00:02:00 |
| Wien Krottenbachstrasse | 12:15:30 | 12:16:00 | 00:02:30 |
| Wien Oberdöbling | 12:17:00 | 12:17:30 | 00:01:00 |
| Wien Heiligenstadt | 12:20:30 | | 00:03:00 |

**Table 3.16:** Schedule of Example 3.14

*The complete journey has an overall energy consumption of* $e = 1.175\,\mathrm{kWh}$*, due to low driving speeds and the track inclination. Figure 3.25 shows the resulting driving strategy. Table 3.17 gives an overview about the results, consisting of the following columns:*

- *Driving type (*Acceleration phase, *speed-*Hold phase, Coasting phase, Braking phase*)*
- *Start position* [m] *of the current driving type.*
- *Final position* [m] *of the current driving type.*
- *Speed* [km/h] *at the start position of the current driving type.*
- *Speed* [km/h] *at the final position of the current driving type.*
- *Accumulated energy consumption* [kWh] *at the end of the current driving type.*

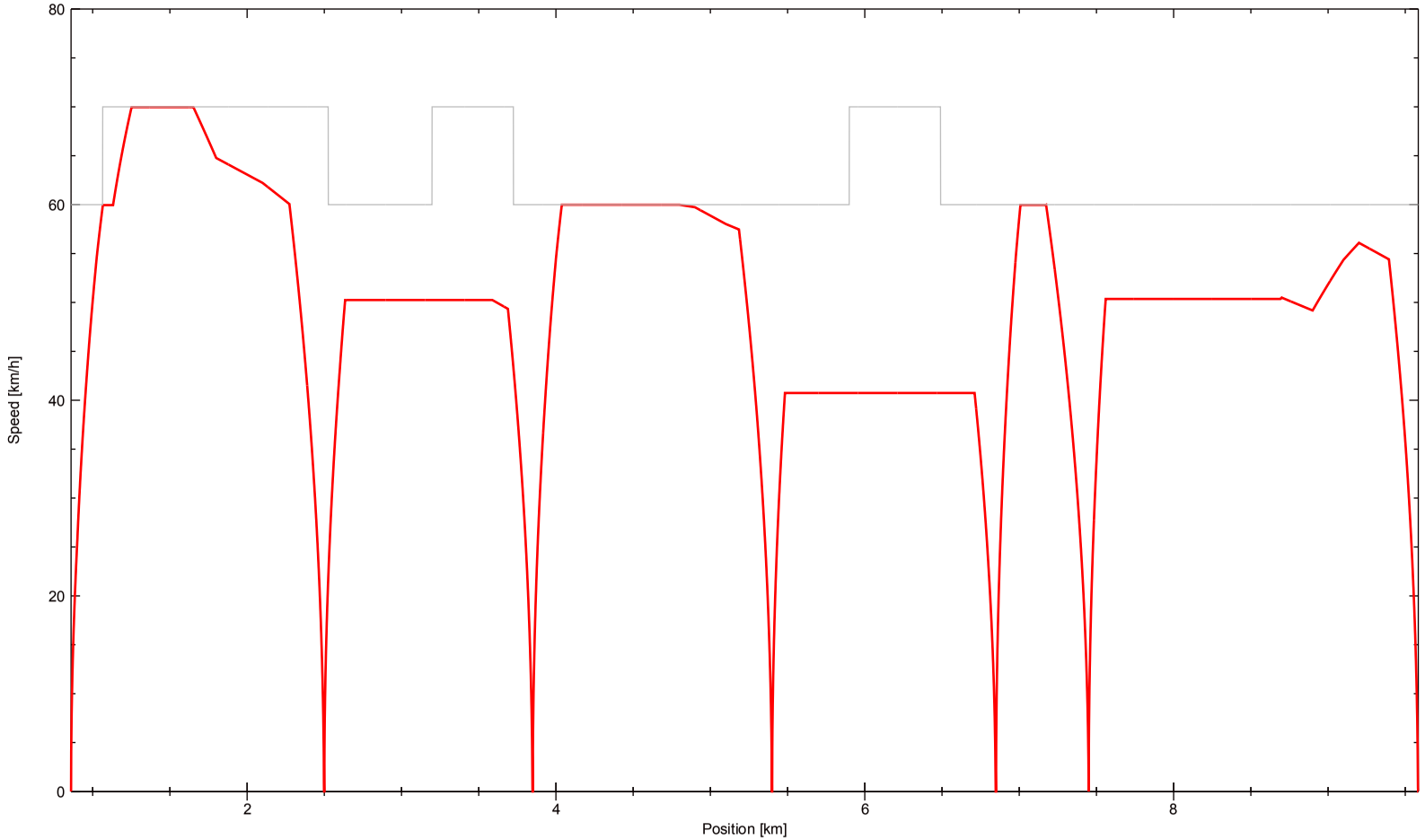



**Figure 3.25:** Driving strategy for Example 3.14

| | | | | | |
|---|---|---|---|---|---|
| **Wien Breitensee – Wien Ottakring** ($t = 119$ s) | | | | | |
| A | 860 | 1 066 | 0 | 60 | 6.076 |
| H | 1 066 | 1 132 | 60 | 60 | 6.418 |
| A | 1 132 | 1 252 | 60 | 70 | 9.216 |
| H | 1 252 | 1 652 | 70 | 70 | 12.055 |
| C | 1 652 | 2 274 | 70 | 60 | 12.055 |
| B | 2 276 | 2 500 | 60 | 0 | 7.805 |
| **Wien Ottakring – Wien Hernals** ($t = 119$ s) | | | | | |
| A | 2 500 | 2 635 | 0 | 50 | 4.115 |
| H | 2 635 | 3 588 | 50 | 50 | 3.996 |
| C | 3 588 | 3 689 | 50 | 49 | 3.996 |
| B | 3 689 | 3 850 | 49 | 0 | 0.973 |
| **Wien Hernals – Wien Gersthof** ($t = 118$ s) | | | | | |
| A | 3 820 | 4 038 | 0 | 60 | 5.542 |
| H | 4 038 | 4 800 | 60 | 60 | 5.137 |
| C | 4 800 | 5 185 | 60 | 57 | 5.137 |
| B | 5 185 | 5 400 | 57 | 0 | 1.094 |
| **Wien Gersthof – Wien Krottenbachstraße** ($t = 148$ s) | | | | | |
| A | 5 400 | 5 483 | 0 | 40 | 2.536 |
| H | 5 483 | 6 711 | 40 | 40 | −1.320 |
| B | 6 711 | 6 850 | 40 | 0 | −3.908 |
| **Wien Krottenbachstraße – Wien Oberdöbling** ($t = 62$ s) | | | | | |
| A | 6 850 | 7 008 | 0 | 60 | 4.662 |
| H | 7 008 | 7 173 | 60 | 60 | 4.217 |
| B | 7 173 | 7 450 | 60 | 0 | −0.997 |
| **Wien Oberdöbling – Wien Heiligenstadt** ($t = 170$ s) | | | | | |
| A | 7 450 | 7 560 | 0 | 50 | 3.352 |
| H | 7 560 | 8 691 | 50 | 50 | −0.222 |
| C | 8 691 | 9 394 | 50 | 54 | −0.222 |
| B | 9 437 | 9 584 | 54 | 0 | −3.792 |

**Table 3.17:** Results of Example 3.14

CHAPTER 4

# Kronecker Algebra

*Kronecker Algebra* consists of *Kronecker Sum* and *Kronecker Product* [56]. This chapter describes the mathematical background, the application of Kronecker Algebra in railway operation, and the representation of routes, track sections, and the behavior of the complete system.

## 4.1 Introduction to Kronecker Algebra

Kronecker Algebra was introduced to model concurrent systems (cf. [56]), in particular for computer systems consisting of several threads which access shared memory. The model was modified for the usage in railway systems (cf. [58, 76–81]). A short overview of several applications like travel time analysis or energy awareness are given at the end of this chapter.

Kronecker Sum and Kronecker Product are simple matrix operations, which can be used to model synchronization of shared resources and generate interleavings. In the following, both operations will be defined. The used matrices will be out of $\mathcal{M} = \{M = (m_{i,j}) | m_{i,j} \in \mathcal{L}\}$ and only matrices $M \in \mathcal{M}$ will be used furthermore. Let $o(M)$ refer to the order of matrix $M \in \mathcal{M}$[1]. Further on $n$-by-$n$ zero matrices $Z_n = (z_{i,j})$, where $\forall i, j : z_{i,j} = 0$ will be used.

In computer science, the access to a shared memory by several threads must be synchronized and for this reason Kronecker Algebra is applied. In contrast to computer science, trains are used instead of threads and track sections instead of shared memory for railway systems. The access to the resource is modeled by semaphores in the sense of computer science (cf. [19]). As a result, a train can enter a track section only if it is not occupied by another train.

The application of Kronecker Algebra allows finding conflicts (e.g. headway-conflicts, deadlocks). The resulting graph shows all possible movements of the trains and based on this graph, a conflict-free situation can be found (if it is possible, due to the given timetable). Further details

[1] A $k$-by-$k$ matrix is known as square matrix of order $k$.

can be found in the description of the optimization algorithm in Chapter 5. Other approaches try to find a conflict-free situation by adding the trains successively into the time-distance diagram (cf. [39], Introduction in Chapter 1). That will be possible only for small railway systems with a small amount of trains.

### Representation

As already mentioned, Kronecker Algebra is a mathematical model which can be used to calculate a matrix describing a complete railway system. Directed graphs are used to represent the movements of the trains and the operations on track sections. Each train is assigned at least one route consisting of at least one track section. A track section can be part of several routes. As a result the routes describe the movement of the trains and can be represented as graphs, too. Each graph can be represented by its adjacency matrix which will be used as input for the calculations.

Assume that the edges in the graphs are labeled by elements of a semiring. Definitions and properties can be found in [44, 56]. The semiring consists of a set of labels $\mathcal{L}$ containing the following semaphore calls (cf. [19]):

- $p_i$ denotes entering or reserving track section $i$, in particular $T_j.p_i$ means that train $j$ wants to enter track section $i$.
- $v_i$ denotes leaving or releasing track section $i$, in particular $T_j.v_i$ means that train $j$ will leave track section $i$.

A railroad system consists of several trains and track sections which are represented by graphs and their adjacency matrices. In general, binary semaphores are used to represent the operations on track sections, whereas each track section is modeled by its own semaphore. If an operational standard allows entry into an occupied block (*permissive driving*), this approach can deal with this issue as well by fine-scaling the track section or by using counting semaphores instead of binary semaphores. Each edge in the graphs is labeled by $l \in \mathcal{L}$. If there exists an edge labeled $a$ from node $i$ to node $j$, then the corresponding adjacency matrix $M$ has $m_{i,j} = a$. If there exists no edge between two nodes, then $m_{i,j} = 0$.

**Definition 4.1** (System model)**.** *The system model consists of the tuple $\langle \mathcal{T}, \mathcal{S}, \mathcal{L} \rangle$, where $\mathcal{T}$ and $\mathcal{S}$ refer to the set of graph adjacency matrices describing the train routes and the track sections, respectively. The labels in $T \in \mathcal{T}$ and $S \in \mathcal{S}$ are elements of $\mathcal{L}$.*

A directed labeled graph $G = \langle V, E, n_e \rangle$ consists of a set of labeled nodes $V$, a set of labeled directed edges $E \subseteq V \times V$, and an entry node $n_e$. The previously mentioned set $V$ and $E$ are out of $\langle \mathcal{T}, \mathcal{S}, \mathcal{L} \rangle$. A detailed description of the representation can be found in [56].

## 4.2 Modeling synchronization

**Definition 4.2** (Kronecker Product [56])**.** *Given a $m$-by-$n$ matrix $A$ and a $p$-by-$q$ matrix $B$, their Kronecker Product denoted by $A \otimes B$ is a $mp$-by-$nq$ block matrix defined by*

$$A \otimes B = \begin{pmatrix} a_{1,1} \cdot B & \cdots & a_{1,n} \cdot B \\ \vdots & \ddots & \dots \\ a_{m,1} \cdot B & \cdots & a_{m,n} \cdot B \end{pmatrix}$$

As already mentioned in [7, 56, 67] Kronecker Product allows to model synchronization. Kronecker Product is also known as *Zehfuss product*, *direct product of matrices*, or *matrix direct product* [54]. Knuth notes in [43] that Kronecker never published anything about it. Zehfuss was actually the first publishing it in the 19th century [85].

**Example 4.1.** *Let* $A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$ *and* $B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$.

*The Kronecker Product is given by*

$$C = A \otimes B = \begin{pmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,1}b_{1,3} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} & a_{1,2}b_{1,3} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,1}b_{2,3} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} & a_{1,2}b_{2,3} \\ a_{1,1}b_{3,1} & a_{1,1}b_{3,2} & a_{1,1}b_{3,3} & a_{1,2}b_{3,1} & a_{1,2}b_{3,2} & a_{1,2}b_{3,3} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,1}b_{1,3} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} & a_{2,2}b_{1,3} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,1}b_{1,3} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} & a_{2,2}b_{1,3} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,1}b_{1,3} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} & a_{2,2}b_{1,3} \end{pmatrix}$$

Due to readability 0 will be replaced by $\cdot$ in die following matrices. In the following some basic properties are given. The proofs can be found in [3, 17, 30, 56, 74]. Let $A$, $B$, $C$, and $D$ be matrices.

- Associativity of the Kronecker Product
$$A \otimes (B \otimes C) = (A \otimes B) \otimes C$$
- Distributivity of the Kronecker Product with respect to the addition of matrices
$$A \otimes (B + C) = (A \otimes B) + (A \otimes C)$$
$$(A \otimes B) + C = (A \otimes C) + (B \otimes C)$$
- The Kronecker Product is non-commutative because in general
$$A \otimes B \neq B \otimes A$$
- Relationship between the ordinary and Kronecker Product of matrices
$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$
- If $T$ denotes transposition
$$(A \times B)^T = A^T \otimes B^T$$
- If $A$ and $B$ are invertible square matrices having the inverses $A^{-1}$ and $B^{-1}$
$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$

## 4.3 Generating interleavings

**Definition 4.3** (Kronecker Sum [30, 45, 56, 68])**.** *Given a matrix $A$ of order $m$ and a matrix $B$ of order $n$, their Kronecker sum denoted by $A \oplus B$ is a matrix of order $mn$ defined by*

$$A \oplus B = A \otimes I_n + I_m \otimes B \tag{4.1}$$

*where $I_m$ and $I_n$ denote the identity matrices*[2] *of order $m$ and $n$, respectively.*

In general, Kronecker Sum calculates the Cartesian product graph of two graphs adjacency matrices (cf. [38, 43, 56]).

**Example 4.2.** *Let* $A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$ *and* $B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$.

*The Kronecker Sum is given by*

$$\begin{aligned}
C &= A \oplus B \\
&= A \otimes I_3 + I_2 \otimes B \\
&= \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \otimes \begin{pmatrix} 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \end{pmatrix} + \begin{pmatrix} 1 & \cdot \\ \cdot & 1 \end{pmatrix} \otimes \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \\
&= \begin{pmatrix} a_{1,1} & \cdot & \cdot & a_{1,2} & \cdot & \cdot \\ \cdot & a_{1,1} & \cdot & \cdot & a_{1,2} & \cdot \\ \cdot & \cdot & a_{1,1} & \cdot & \cdot & a_{1,2} \\ a_{2,1} & \cdot & \cdot & a_{2,2} & \cdot & \cdot \\ \cdot & a_{2,1} & \cdot & \cdot & a_{2,2} & \cdot \\ \cdot & \cdot & a_{2,1} & \cdot & \cdot & a_{2,2} \end{pmatrix} + \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & \cdot & \cdot & \cdot \\ b_{2,1} & b_{2,2} & b_{2,3} & \cdot & \cdot & \cdot \\ b_{3,1} & b_{3,2} & b_{3,3} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & b_{1,1} & b_{1,2} & b_{1,3} \\ \cdot & \cdot & \cdot & b_{2,1} & b_{2,2} & b_{2,3} \\ \cdot & \cdot & \cdot & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \\
&= \begin{pmatrix} a_{1,1} + b_{1,1} & b_{1,2} & b_{1,3} & a_{1,2} & \cdot & \cdot \\ b_{2,1} & a_{1,1} + b_{2,2} & b_{2,3} & \cdot & a_{1,2} & \cdot \\ b_{3,1} & b_{3,2} & a_{1,1} + b_{3,3} & \cdot & \cdot & a_{1,2} \\ a_{2,1} & \cdot & \cdot & a_{2,2} + b_{1,1} & b_{1,2} & b_{1,3} \\ \cdot & a_{2,1} & \cdot & b_{2,1} & a_{2,2} + b_{2,2} & b_{2,3} \\ \cdot & \cdot & a_{2,1} & b_{3,1} & b_{3,2} & a_{2,2} + b_{3,3} \end{pmatrix}
\end{aligned}$$

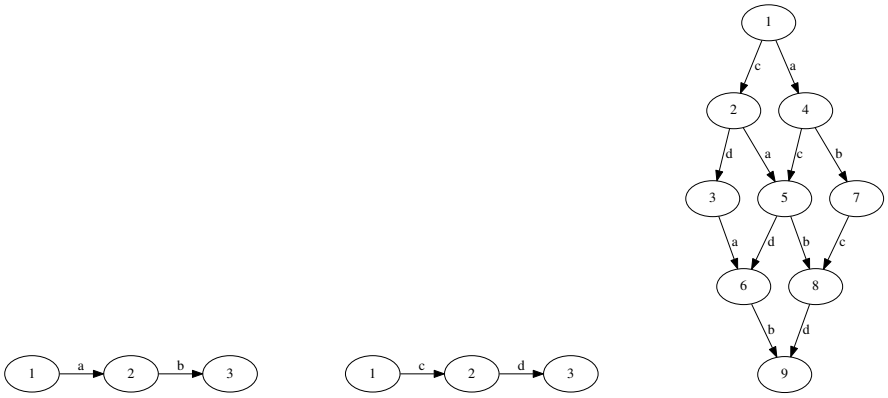
In the following some basic properties of Kronecker Sum are given. Their proofs can be found in [45, 56, 68].

- Let the matrices $A$ and $C$ have order $m$ and $B$ and $D$ have order $n$. Then

$$(A \oplus B) + (C \oplus D) = (A + C) \oplus (B + D)$$

  is called *Mixed Sum Rule*.

[2] The identity matrix $I_n$ is a $n$-by-$n$ square-matrix with ones on the main diagonal and zeros elsewhere




(a) $C$ (b) $D$ (c) $C \oplus D$

**Figure 4.1:** $C$, $D$, $C \oplus D$ (Example 4.3)

- Associativity of the Kronecker Sum

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

- Due to the fact, that the associativity property is valid for both operations, $\otimes$ and $\oplus$, the $k$-fold operations

$$\bigotimes_{i=1}^{k} A_i \text{ and } \bigoplus_{i=1}^{k} A_i$$

  are well defined, too.

- Kronecker Sum calculates all possible interleavings.

**Example 4.3.** *Let the matrices* $C = \begin{pmatrix} \cdot & a & \cdot \\ \cdot & \cdot & b \\ \cdot & \cdot & \cdot \end{pmatrix}$ *and* $D = \begin{pmatrix} \cdot & c & \cdot \\ \cdot & \cdot & d \\ \cdot & \cdot & \cdot \end{pmatrix}$. *The corresponding graphs are depicted in Figure 4.1(a) and 4.1(b), respectively. All possible interleavings by executing* $C$ *and* $D$ *are given in Table 4.1. The corresponding graph is depicted in Figure 4.1(c).*

Now assume that two trains want to use the same track section. Thus, the occupancy needs to be synchronized. An additional matrix will be used to model the track section:

$$S = \begin{pmatrix} \cdot & p \\ v & \cdot \end{pmatrix}$$

As already mentioned, $p$ denotes entering the track section and $v$ means leaving the section [19]. The correct system behavior can be described by a matrix, generated by

$$R = (C \oplus D) \otimes S. \tag{4.2}$$

| Interleavings |
|---|
| $a \cdot b \cdot c \cdot d$ |
| $a \cdot c \cdot b \cdot d$ |
| $a \cdot c \cdot d \cdot b$ |
| $c \cdot a \cdot b \cdot d$ |
| $c \cdot a \cdot d \cdot b$ |
| $c \cdot d \cdot a \cdot d$ |

**Table 4.1:** Interleavings of $C$ and $D$ (Example 4.3)

**Example 4.4.** *Now, let* $C = \begin{pmatrix} \cdot & p & \cdot \\ \cdot & \cdot & v \\ \cdot & \cdot & \cdot \end{pmatrix}$, $D = \begin{pmatrix} \cdot & p & \cdot \\ \cdot & \cdot & v \\ \cdot & \cdot & \cdot \end{pmatrix}$, *and* $S = \begin{pmatrix} \cdot & p \\ v & \cdot \end{pmatrix}$. *Based on Equation 4.2, the resulting matrix is calculated. The corresponding graph is depicted in Figure 4.2. It can be seen, that there are several parts not reachable from the start node and thus, these parts are not relevant for further analysis.*

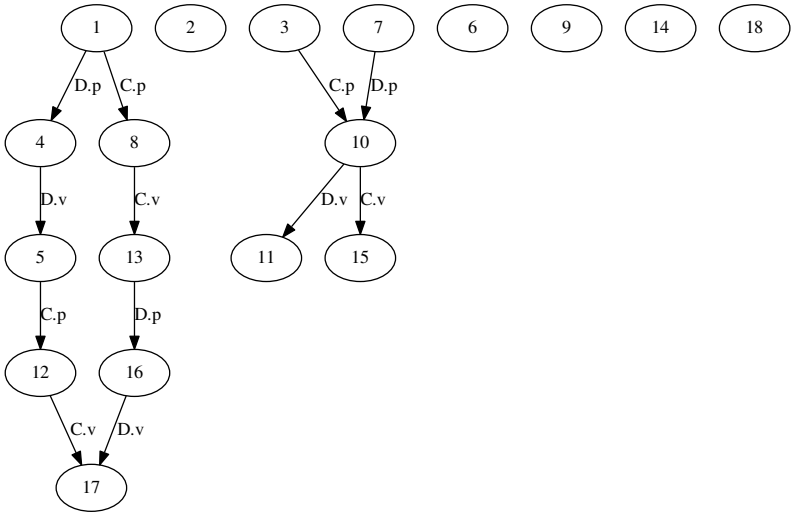



**Figure 4.2:** $(C \oplus D) \otimes S$ (Example 4.4)

## 4.4 System Model

In general, a railway system consists of a set of trains $L = \{L_j | 1 \leq j \leq t\}$. Each train $L_j$ has a route $R_j$, which is a sequence of track sections, whereas each track section is modeled by matrix

$$T_i = \begin{pmatrix} \cdot & p_i \\ v_i & \cdot \end{pmatrix}. \tag{4.3}$$

The set of routes is denoted by $R = \{R_j | 1 \leq j \leq t\}$. The complete system can be described by

$$S = \left( \bigoplus_{j=1}^{t} R_j \right) \otimes \left( \bigoplus_{i=1}^{r} T_i \right). \tag{4.4}$$

**Example 4.5.** *Assuming the same system as in Example 4.4, but now with the railway specific notation. It contains two trains, namely $L_1$ and $L_2$. Both trains want to enter track section 1. Thus, the routes will be as follows.*

$$\begin{aligned} R_1 &= \begin{pmatrix} \cdot & p_1 & \cdot \\ \cdot & \cdot & v_1 \\ \cdot & \cdot & \cdot \end{pmatrix} \\ R_2 &= \begin{pmatrix} \cdot & p_1 & \cdot \\ \cdot & \cdot & v_1 \\ \cdot & \cdot & \cdot \end{pmatrix} \end{aligned}$$

*Track section 1 is modeled by*

$$S_1 = \begin{pmatrix} \cdot & p_1 \\ v_1 & \cdot \end{pmatrix}$$

*Based on Equation 4.4, $\bigoplus_{j=1}^{t} R_j$ and $\bigoplus_{i=1}^{r} T_i$ are calculated:*

$$\begin{aligned} \bigoplus_{j=1}^{t} R_j &= R_1 \oplus R_2 \\ &= R_1 \otimes I_3 + I_3 \otimes R_2 \\ &= \begin{pmatrix} \cdot & \cdot & \cdot & R_1.p_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & R_1.p_1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & R_1.p_1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_1.v_1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_1.v_1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_1.v_1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \end{aligned}$$

$$
+ \begin{pmatrix}
\cdot & R_2.p_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & R_2.v_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & R_2.p_1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & R_2.v_1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_2.p_1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_2.v_1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{pmatrix}
$$

$$
= \begin{pmatrix}
\cdot & R_2.p_1 & \cdot & R_1.p_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & R_2.v_1 & \cdot & R_1.p_1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & R_1.p_1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & R_2.p_1 & \cdot & R_1.v_1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & R_2.v_1 & \cdot & R_1.v_1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_1.v_1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_2.p_1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_2.v_1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{pmatrix}
$$

$$
\bigoplus_{i=1}^{r} T_i \quad = \quad T_1 = \begin{pmatrix} \cdot & p_1 \\ v_1 & \cdot \end{pmatrix}
$$

*The complete system is now calculated by*

$$
\begin{pmatrix}
\cdot & R_2.p_1 & \cdot & R_1.p_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & R_2.v_1 & \cdot & R_1.p_1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & R_1.p_1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & R_2.p_1 & \cdot & R_1.v_1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & R_2.v_1 & \cdot & R_1.v_1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_1.v_1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_2.p_1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_2.v_1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{pmatrix}
\otimes \begin{pmatrix} \cdot & p_1 \\ v_1 & \cdot \end{pmatrix}
$$

*which results in a matrix of size* 18*. For further illustration, only nodes reachable from the entry node are depicted in the corresponding graphs. The resulting graph of the example is illustrated in Figure 4.3, already colored as described in the next section (Section 4.5). It can be seen that there are two possible paths within the graph:*

- $R_1.p_1$, $R_1.v_1$, $R_2.p_1$, $R_2.v_2$*: Train* 1 *(using route* $R_1$*) enters track section* 1 *and releases section* 1*. Then train* 2 *(using route* $R_2$*) can enter and release section* 1*.*
- $R_2.p_1$, $R_2.v_1$, $R_1.p_1$, $R_1.v_1$*: Train* 2 *(using route* $R_2$*) enters track section* 1 *and releases section* 1*. Then train* 1 *(using route* $R_1$*) can enter and release section* 1*.*

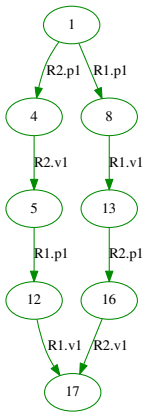



**Figure 4.3:** $\left(\bigoplus_{j=1}^{t} R_j\right) \otimes (\bigoplus_{i=1}^{r} T_i)$ (Example 4.5)

## 4.5 Node types

After applying Kronecker Algebra, the resulting graph may contain various types of nodes which are defined as follows.

**Definition 4.4** (Deadlock)**.** *If the final node cannot be reached from a certain node $d$, then node $d$ denotes a* deadlock *situation or a situation which will definitely result in a deadlock.*

**Definition 4.5** (Safe state)**.** *A state is* safe *if all trains can perform their actions without having to take into account the movements of others trains within the system. From safe states only safe states can be reached.*

**Definition 4.6** (Critical state)**.** *A state is* critical *if both, a deadlock and a safe state can be reached.*

**Definition 4.7** (Synchronizing nodes)**.** *A* synchronizing node *is a node $s$ such that*

- *there exists an edge $e_{in} = (i, s)$ with label $v_k$ and*
- *there exists an edge $e_{out} = (s, j)$ with label $p_k$,*

*where $k$ denotes the same track section and $e_{in}$ and $e_{out}$ are mapped to different trains.*

**Example 4.6.** *Regarding node $5$ of the resulting graph of Example 4.5 (Figure 4.3). It has an incoming edge labeled by $R_2.v_1$ and an outgoing edge $R_1.p_1$. Thus the node is a synchronizing node.*

The following two definitions will be used to indicate trains entering or leaving a track section.

**Definition 4.8** (Leaving trains)**.** *A train $j$, which is about to leave track section $i$ is denoted by $T^j_{\boxed{i}\rightarrow}$.*

**Definition 4.9** (Entering trains)**.** *A train $k$, which is about to enter a track section $i$ is denoted by $T^k_{\rightarrow\boxed{i}}$.*

**Definition 4.10** (Synchronizing condition)**.** *Each synchronizing node has assigned at least one synchronizing condition. Each synchronizing condition contains a train leaving a track section ($T^j_{\boxed{i}\rightarrow}$), a train entering the track section ($T^k_{\rightarrow\boxed{i}}$), and the track section ($i$) itself. The notation of a synchronizing condition is as follows.*

$$T^j_{\boxed{i}\rightarrow} \rightarrow T^k_{\rightarrow\boxed{i}} : i$$

**Example 4.7.** *Again, regarding node* $5$ *of Example 4.5 (Figure 4.3), the synchronizing condition follows as*

$$2 \rightarrow 1 : 1$$

*as train* $2$ *leaves track section* $1$ *and train* $1$ *enters it.*

**Definition 4.11** (Multi-synchronizing nodes)**.** *A* multi-synchronizing node *is similar to the synchronizing node, but with more than one synchronizing conditions.*

**Example 4.8.** *A multi-synchronizing node can be found in the resulting graph of Example 4.10 (Figure 4.5). Node* $4627$ *has two synchronizing conditions:*

$$1 \rightarrow 2 : 3$$
$$1 \rightarrow 3 : 3$$

**Definition 4.12** (Stop-node)**.** *A* stop node *is a node where a train has a stop at a railway station for departure or arrival.*

**Example 4.9.** *Stop nodes will be available in the resulting graph of Example 5.1 (Figure 5.5).*

Table 4.2 gives an overview of the different node types and their graph-illustration.

| Node type | Illustration |
|---|---|
| Deadlocks | Red |
| Safe states | Green |
| Critical states | Orange |
| Synchronizing nodes | Filled |
| Multi-synchronizing nodes | Filled, double border |
| Stop nodes | Rectangular |

**Table 4.2:** Node types and their illustration

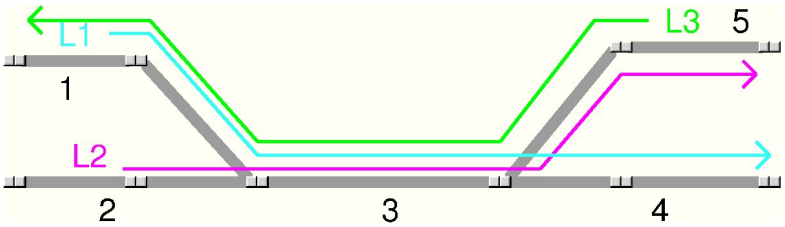



**Figure 4.4:** Railway system (Example 4.10)

**Example 4.10.** *This example was introduced in previous publications (eg. [58, 81]) and shows a railway system (Figure 4.4) with five track sections and three trains. The routes of the trains are as follows:*

$$R_1 = \begin{pmatrix} \cdot & p_3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & v_1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & p_4 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & v_3 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & v_4 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix},$$

$$R_2 = \begin{pmatrix} \cdot & p_3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & v_2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & p_5 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & v_3 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & v_5 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}, \text{ and}$$

$$R_1 = \begin{pmatrix} \cdot & p_3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & v_5 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & p_1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & v_3 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & v_1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

*The five track sections are modeled by* $T_i = \begin{pmatrix} \cdot & p_i \\ v_i & \cdot \end{pmatrix}$ *for* $1 \leq i \leq 5$. *The resulting matrix will have order* $6 \cdot 6 \cdot 6 \cdot 2^5 = 6912$ *but only a small part (42 nodes) is reachable from the entry node. The resulting graph is shown in Figure 4.5. It contains safe states, critical states, deadlocks and various types of synchronizing nodes.*

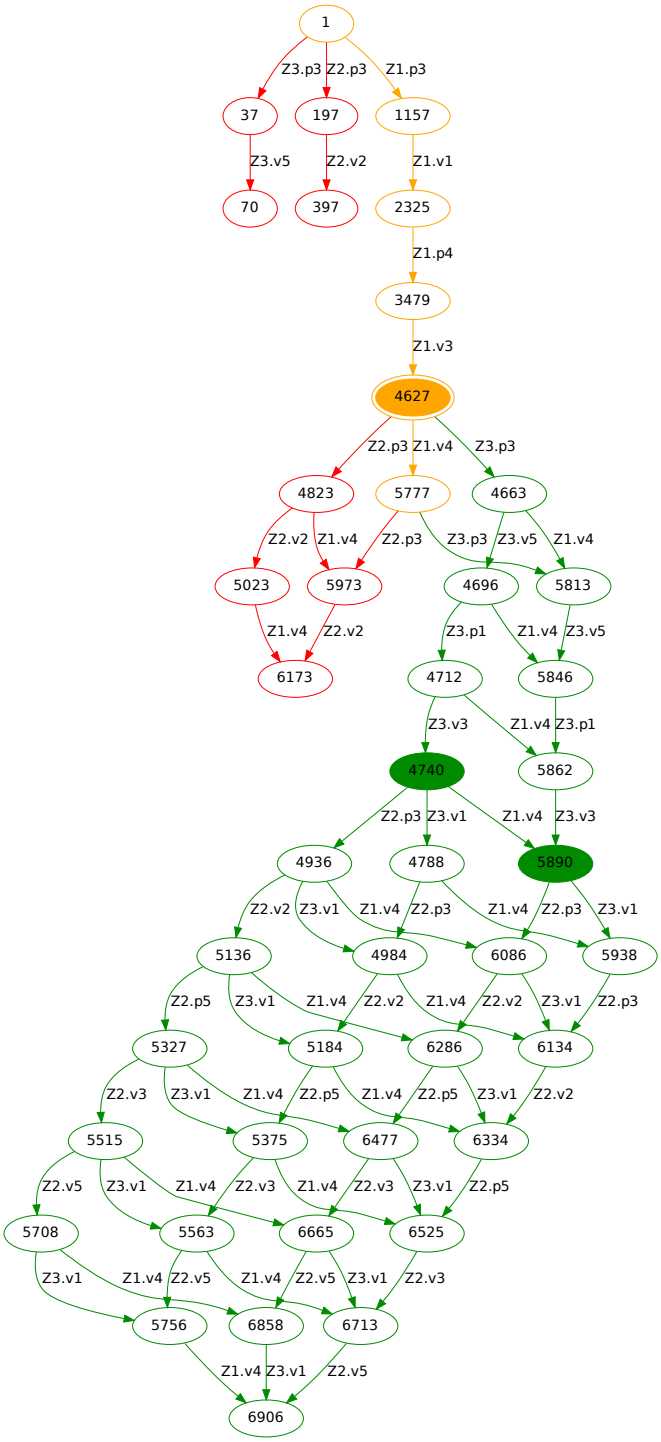



**Figure 4.5:** Resulting graph (Example 4.10)

## 4.6 Alternative routes

In general, each train has a well defined route within the railway system. In some special situations it is necessary that a train leaves the main track, for instance while overtaking or due to a blocked track section. Therefore *alternative routes* can be defined (cf. [39]).

**Example 4.11.** *Again, the same railway system as in Example 4.10 is used, but now, each train has an alternative route:*

- $T_1$*:* $1 \to 3 \to 4$ *or* $1 \to 3 \to 5$.
- $T_2$*:* $2 \to 3 \to 5$ *or* $2 \to 3 \to 4$.
- $T_3$*:* $5 \to 3 \to 1$ *or* $5 \to 3 \to 2$.

*As a consequence, the resulting graph (Figure 4.6) is much wider than in the previous example, but now it is possible that the trains can use a different route, if the preferred route cannot be used.*

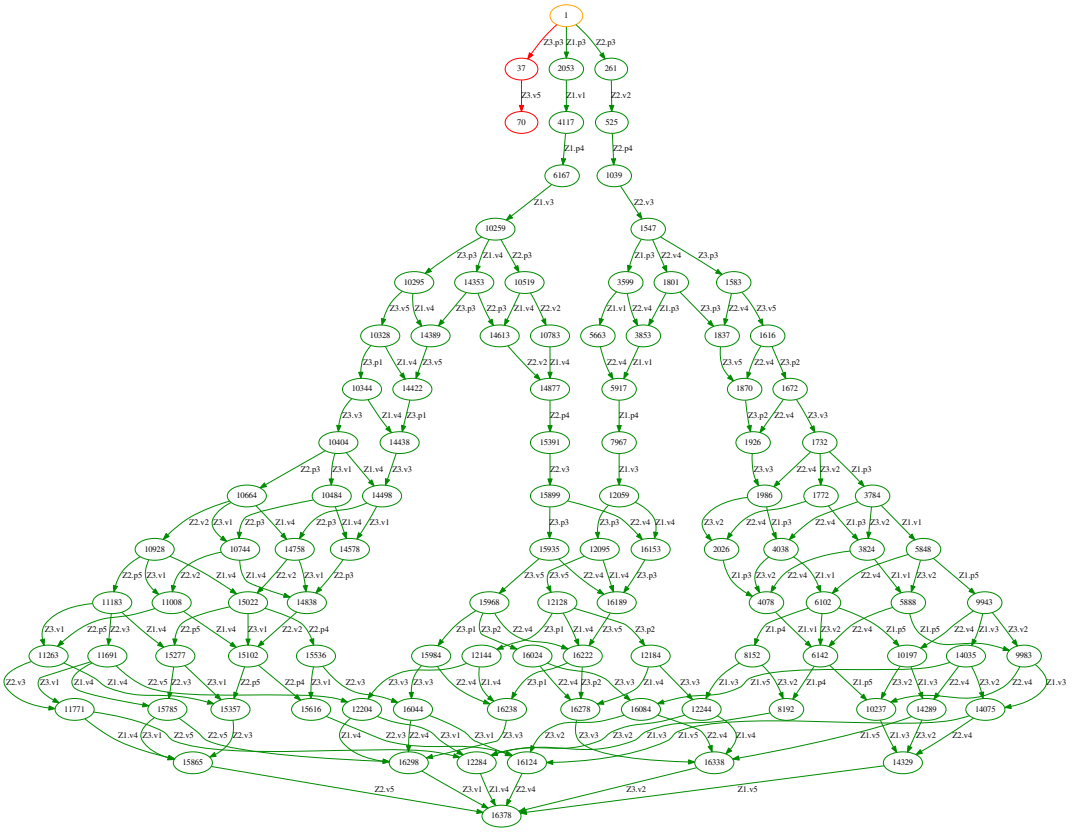


**Figure 4.6:** Resulting graph (Example 4.11)

## 4.7 Properties of the resulting matrix

As already proved in [56], the number of edges after the application of Kronecker Sum is $O(mn)$[3]. Thus, the number of edges is linear in the order of the resulting adjacency matrix. The number of nodes is bounded from above by $n^k$, where $k$ is the number of trains within the railway system and the route of each train consists of $n$ nodes. Based on Lemma 7 of [56], the resulting matrix, describing the complete railway system, is sparse. This enables the application of memory saving data structures and efficient algorithms (e.g. by using adjacency lists, which are linear in the number of nodes). In the worst case, however, the number of nodes increases exponentially in the number of trains [56].

Further information about the efficient implementation of the matrix operations and the lazy implementation of Kronecker Algebra are given in [56] and are not further explained here.

## 4.8 Lazy Implementation of Kronecker Algebra

The resulting graph after applying Kronecker Algebra contains unreachable parts due to synchronization (cf. [56]). For railway systems with lots of synchronizations, the reachable parts may be very small. This observations motivates the lazy implementation (cf. [31]) described in [56].

Choosing a lazy implementation for the matrix operations ensures that, when extracting the reachable parts of the underlying graph, the overall effort is reduced to exactly these parts. By starting from the graph's start node and calculating all reachable successors the lazy implementation exactly does this. Thus, for example, if the resulting graph's size is linear in terms of the involved trains, only linear effort will be necessary to generate the graph. Further information of the implementation can be found in [56].

## 4.9 Extensions of the model

### Travel time analysis

As introduced in [81] the model can be extended to calculate the travel time for each train within the railway system. This is based on [57], which is concerned with the Timing Analysis of Concurrent Programs.

For the travel time analysis each node is assigned a variable and an equation is setup based on the predecessor. The variable contains a vector where each component of the vector corresponds to a train. The equations are used to calculate the travel time. Additionally, each edge is assigned a travel time value which is used in the equations to calculate the complete travel time for each train. Definitions and examples can be found in [57, 76–78, 80, 81].

[3] Assuming the corresponding matrices have order $m$ and $n$, respectively

### Energy awareness

The model can be extended to use shared resources which can be used by more than one train simultaneously. Counting semaphores [19] can be used to model discrete power resources. In particular, the available energy is quantized into standardized packages, e.g. $1\,\mathrm{MWh}$. A power station or substation produces an amount of energy packages. Trains can reserve a well defined amount of these packages when entering a track section. After leaving a track section, the packages are released and thus, available for other trains. This model was introduced in [76] and can be used to find deadlocks and to minimize energy demand. Examples can be found in [76, 77, 79, 80].

### Extra-long trains

In the previous examples it was assumed that the length of a train is less than the length of a track section and thus, the train reserves only two track sections simultaneously. For extra-long trains which are much longer than a track section, several track sections must be reserved simultaneously. An example for handling such trains can be found in [79] and [76].

CHAPTER 5

# System optimization

Chapter 3 describes the optimization of the driving strategy of a single train. If there are more trains within a railway network, each train might influence other trains. For instance, a train wants to enter a track section, which is occupied by another train. In this case, the train has to wait until the track section is released.

This chapter describes the optimization of the complete railway system, based on Kronecker Algebra (Chapter 4) and the single trip optimization (Chapter 3). The algorithm to get the optimized behavior in term of energy consumption of all trains within a railway system is explained in the following sections and consists of several parts:

- Kronecker Algebra based system analysis
- Graph reduction
- Determine all possible routes
- Finding the optimal driving strategy

The explanation of the algorithm will be supported by the following example.

**Example 5.1.** *The railway system is illustrated in Figure 5.1. Each track of the railway system is divided into track sections and each section has a unique number* [1]*. The example in Figure 5.1 consists of seven track sections. The unique number, the start position, and the end position of each track section are given in Table 5.1.*

[1] The red numbers in Figure 5.1 denote the unique number of the track section. The borders of each track section are marked by blue lines.

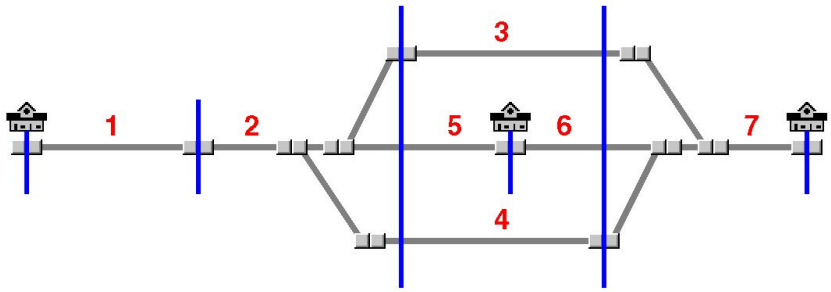



**Figure 5.1:** Railway System

| Track Section | Start Position [m] | End Position [m] |
|---|---|---|
| 1 | 1 000 | 5 000 |
| 2 | 5 000 | 9 000 |
| 3 | 9 000 | 14 000 |
| 4 | 9 000 | 14 000 |
| 5 | 9 000 | 12 000 |
| 6 | 12 000 | 14 000 |
| 7 | 14 000 | 18 000 |

**Table 5.1:** Track sections of railway system (Figure 5.1)

*There are three trains within the railway system with the following routes*[2]*:*

- *Train* 1 *(*$T_1$*):* $1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7$*. Train* $T_1$ *starts at position* 1 000 m *at track section* 1*, has a stop at position* 12 000 m *at the end of section* 5 *and will finish its journey at position* 18 000 m *at track section* 7*.*
- *Train* 2 *(*$T_2$*):* $1 \rightarrow 2 \rightarrow 3 \rightarrow 7$*. Train* $T_2$ *starts at position* 1 000 m *at track section* 1 *and finishes its journey at position* 18 000 m *at section* 7 *without any further stop.*
- *Train* 3 *(*$T_3$*):* $7 \rightarrow 4 \rightarrow 2 \rightarrow 1$*. The third train (*$T_3$*) starts its journey at position* 18 000 m *(section* 7*) and stops at position* 1 000 m *at track section* 1*.*

*Obviously, the three trains have to use some common track sections (*1*,* 2*, and* 7*). For the railway application it has to be ensured that a track section is occupied by one train only and thus the allocation of the track sections must be analyzed and their access synchronized, to guarantee that there is no overlapping of their blocking-time. This approach will be explained in the following subsections.*

[2] A route of a train consists of several track sections. A track section can be part of several routes.

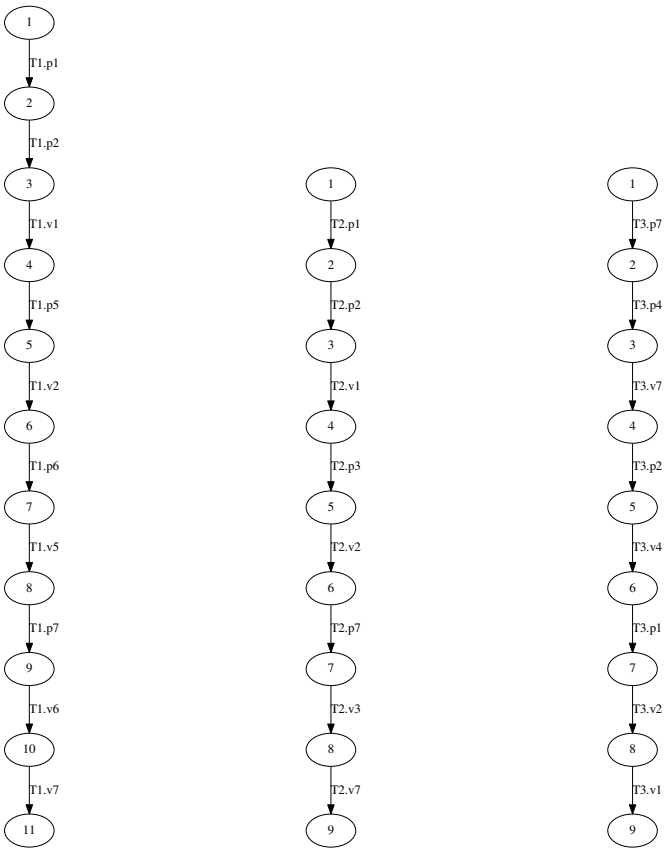



**Figure 5.2:** Routes of train $T_1$, $T_2$, and $T_3$

## 5.1 Kronecker Algebra based system analysis

In a railway system with several trains there might occur a deadlock situation. For instance it might happen that train $T_3$ reaches track section 2 before $T_1$ has left track section 1. Train $T_1$ wants to enter section 2 which is occupied by $T_3$.

The first part of the optimization algorithm is to apply Kronecker Algebra, which was already introduced in Chapter 4, to get all movements of the trains within the railway network. For this reason each route must be given by using the $p$ and $v$ operations (cf. [19]) of the track sections. Assuming that the length of the train is less than the length of each track section, a train does not need to reserve more than two track sections at once. The routes of the trains now read as follows:

$$\begin{aligned} R_1 &= p_1, p_2, v_1, p_5, v_2, p_6, v_5, p_7, v_6, v_7 \\ R_2 &= p_1, p_2, v_1, p_3, v_2, p_7, v_3, v_7 \\ R_3 &= p_7, p_4, v_4, p_2, v_4, p_1, v_2, v_1 \end{aligned}$$

Each route can be illustrated as a graph, where the edges are labeled by the $p$ and $v$ operations. The corresponding graphs of the three routes are depicted in Figure 5.2. As already explained in Chapter 4, each graph can be represented by its adjacency matrix, which are the following for the three routes:

$$R_1 = \begin{pmatrix} \cdot & p_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & p_2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & v_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & p_5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & v_2 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & p_6 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_5 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & p_7 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_6 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_7 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix},$$

$$R_2 = \begin{pmatrix} \cdot & p_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & p_2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & v_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & p_3 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & v_2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & p_7 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_3 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_7 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}, \text{ and}$$

$$R_3 = \begin{pmatrix} \cdot & p_7 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & p_4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & v_7 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & p_2 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & v_4 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & p_1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_2 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & v_1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

In addition, the track sections must be modeled by semaphores in the sense of computer science (cf. [19]). The matrix of track section $i$ will be the following ($1 \leq i \leq n$, where $n$ is the number of track sections, in particular $n = 7$ for the example):

$$S_i = \begin{pmatrix} \cdot & p_i \\ v_i & \cdot \end{pmatrix}$$

By applying Kronecker Algebra, a matrix of size 19 200 is created and thus there exist 19 200 states. The resulting matrix consists of all possible train movements, including deadlocks. It will be reduced to the relevant nodes in the following section.

## 5.2 Graph reduction

The resulting graph after the application of Kronecker Algebra can be reduced to the relevant synchronizing nodes. The algorithm is designed to run on multi-core CPUs. There are several running tasks, where each task has a unique id, which is a positive integer. The maximum number of tasks can be adjusted in the algorithm settings. The graph reduction consists of the following four steps, which are described afterwards:

1. Determine the graph nodes.
2. Find the synchronizing nodes.
3. Check the synchronizing nodes.
4. Find a path between the synchronizing nodes.

Each part is designed to run on multi-core CPUs. Most of the operations are done solely within the task. If access to shared memory is unavoidable, it is guarded by *protected objects* (cf. [5, 75]), which are part of the Ada programming language. After the execution of the tasks, relevant data is copied from the local memory of the tasks to the global program memory.

### Part I: Determine the graph nodes

As already mentioned the resulting graph is represented by a matrix. The first step of the graph reduction is to find all nodes, which are reachable from the entry node[3]. This is done by a breadth-first search, where each reachable node is added to a task-local set of nodes. After all nodes are examined, each task copies its data to a global set of nodes. As a result, a graph consisting of nodes reachable from the entry node remains. In this resulting graph, there might be all types of nodes available, including deadlocks which will be eliminated in the next step. There might exist nodes within the resulting graph which are not reachable from the entry node. These nodes do not show up because of this reduction procedure. As a consequence, all remaining nodes represent possible states in the railway system and all edges represent train movements, respectively.

The resulting matrix after the application of Kronecker Algebra for the example has size 19 200 and thus 19 200 possible states. Due to the reduction the resulting graph is scaled down to 499 nodes which is illustrated in Figure 5.3. It contains red nodes (deadlocks), green nodes (save states), and orange nodes. The entry node has node id 1 and the final node 113 921.

[3] A node $u_j$ is reachable from $u_i$, if there exists a path from $u_i$ to $u_j$

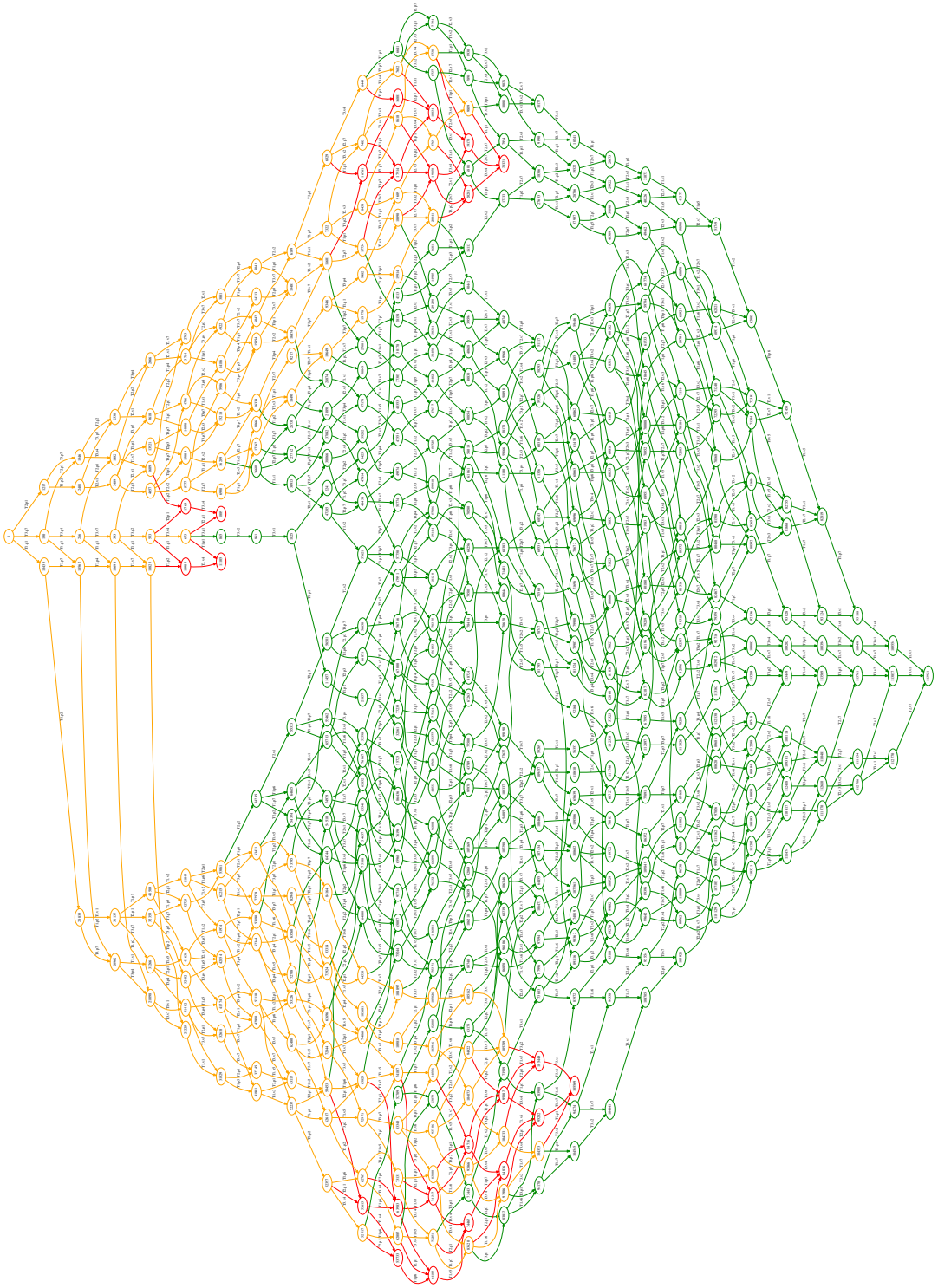


**Figure 5.3:** Resulting graph (Part I)

### Algorithm

The algorithm starts by adding the start node to the node-queue of a particular task. Based on the node-id (*nid*), a hash-function calculates the id of the task, where the computations are done. The hash-function is used to guarantee the optimal distribution of the nodes to the running tasks. Immediately afterwards all tasks will start their execution and run until all task queues are empty. The tasks will use so called *Barriers* (cf. [5, 75]) from the Ada programming language to guarantee that all tasks simultaneously check their queues. If the tasks are not synchronized it might happen that a task checks its queue before another task will enqueue a new node. As a result, the task will stop its execution although its queue is not empty. To avoid that a node is examined several times, it is added to a set of nodes after its first examination. When a task takes a new node from its queue, it only examines the node, if it was not examined yet. Lock-free queues are used in each task to avoid additional synchronization and thus, to increase performance. Each task will execute the following until the termination condition is valid:

- Fetch the next node from the queue, if it is not empty.
- If the node was not examined yet, it is added to the set of nodes and its successors will be added to the corresponding node-queues, calculated by the hash-function.

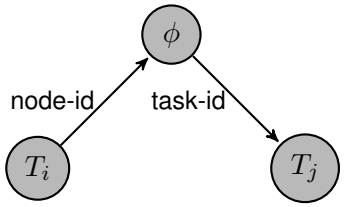



**Figure 5.4:** Hash-function

The node-ids are evenly distributed over all available tasks by the hash-function (cf. [42]). Another advantage using the hash-function is that a particular node-id is assigned to the same task every time. Thus all information of this node can be stored at the tasks internal memory and does not need to be shared with other tasks, which eliminates the risk of race conditions and additional blocking time due to further synchronization. Figure 5.4 illustrates the functionality of the hash-function $\phi$. Equation 5.1 shows the calculation of the task-id based on a given node-id, where *tid* is the calculated id of the task, *nid* is the given node-id, and *ntask* contains the number of available tasks. The description of the algorithm can be found in Algorithm 5.1.

$$tid = nid \cdot \left[ nid \cdot \frac{\sqrt{5}-1}{2} - \lfloor nid \cdot \frac{\sqrt{5}-1}{2} \rfloor \right] \mod (ntask + 1) \tag{5.1}$$

## Part II: Find the synchronizing nodes

The goal of the second part of the graph reduction is to eliminate all deadlock-nodes. There exists no path from the deadlock-nodes to the final node (cf. [56, 57]). Therefore, the graph

**Algorithm 5.1:** Determine the graph nodes

```
1  not_finished:= true;
2  while not_finished do
3      while task_queue.Not_Empty do
4          node:= task_queue.Dequeue;
5          if node ∉ task_nodes then
6              task_nodes:= task_nodes ∪ node;
7              foreach successor of node.Successors do
8                  task_queue.Enqueue(successor);
9              end
10         end
11     end
12     if AllQueuesEmpty then
13         not_finished:=false;
14     end
15 end
```

is traversed bottom-up from the final node to the entry node. Similar to the first part of the algorithm, each reachable node is added to a set of nodes and its predecessors are added to the queues of the running tasks, whereby the task-id is again calculated by the hash-function. As a consequence, the deadlock-nodes are not reachable from the final node and thus not added to the set of nodes. In addition, all synchronizing nodes and stop nodes are determined and saved in separate maps.

### Algorithm

Initially, the final node is added to the corresponding task-queue (calculated by the hash-function). Afterwards all available tasks will start their execution, which consists of the following steps.

- Fetch the next node from the queue, if it is not empty.
- If the node was not examined yet, it is added to the set of nodes and its predecessors will be added to the corresponding node-queues, calculated by the hash-function.

Additionally each node is examined if it is a stop-node or a synchronizing-node:

- If the node was found in the previous part of the algorithm and it was not examined yet, it is added to the set of visited nodes.
- If the node is a synchronizing node, it is added to the set of synchronizing nodes and to the queue of synchronizing nodes, which is used in the next part of the algorithm
- If the node is a stop node, it is added to the set of stop nodes and to the queue of synchronizing nodes.

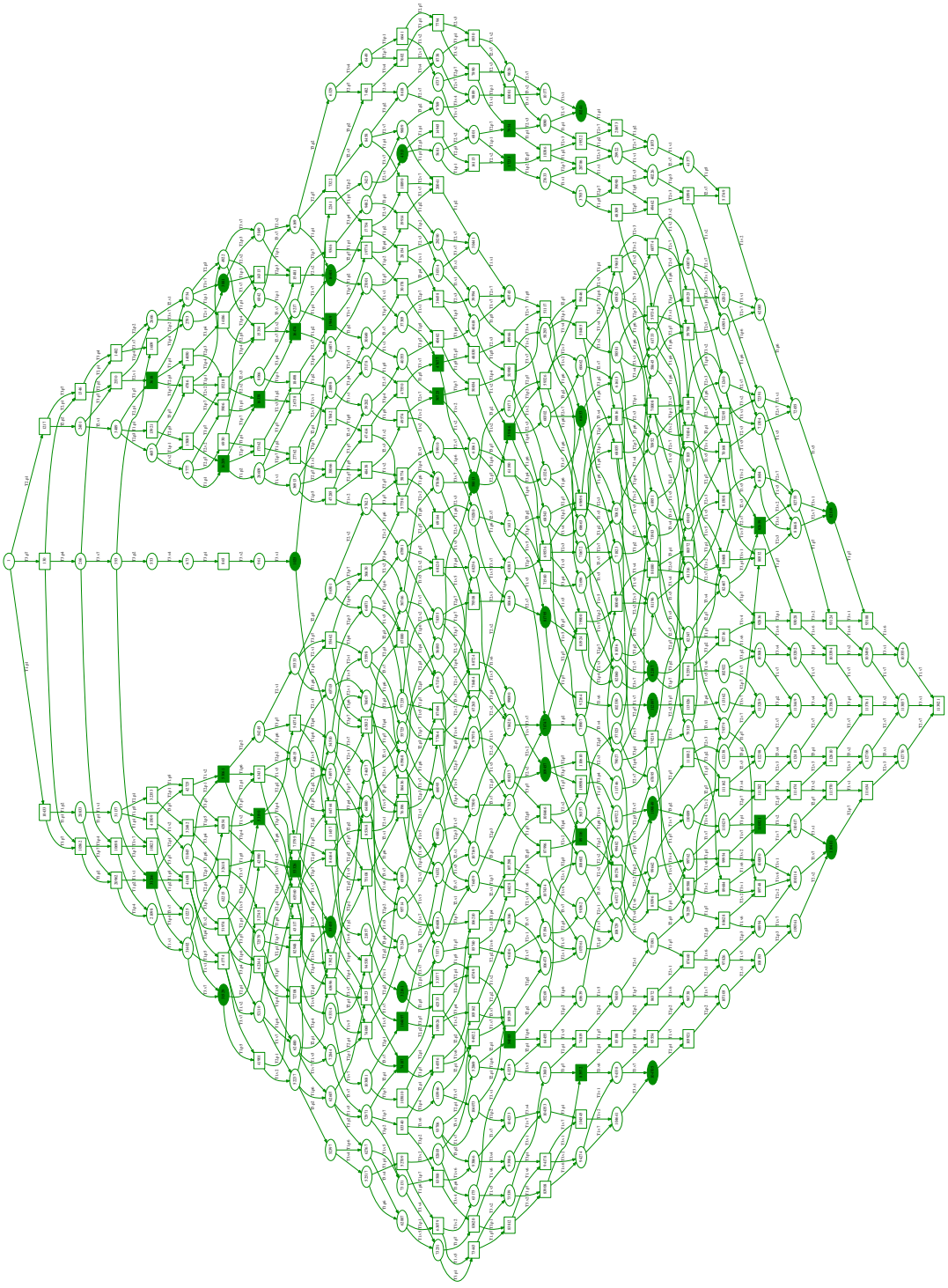


**Figure 5.5:** Resulting graph (Part II)

Similar to the previous part of the algorithm, the task terminates, if all task queues are empty. Synchronization is again done by Barriers. Figure 5.5 shows the resulting graph of part II of the example. In comparison to Figure 5.3 the red nodes have been eliminated and the size has shrunken from 499 nodes to 475 nodes. All remaining nodes are colored green because each state in the resulting graph is safe now. The filled nodes represent synchronizing nodes, the rectangular nodes denote stop nodes, and some of them represent both, synchronizing and stop nodes. For instance node 31 266 has an input label $T_1.v_1$ and an output label $T_2.p_1$. These two trains must be synchronized in their access on track section 1, which is released by train $T_1$ and reserved by train $T_2$, which additionally has track section 1 as start of its journey.

Table 5.2 gives an overview of the number of nodes, the number of synchronizing nodes and stop nodes, and the number of normal nodes.

| Node type | Amount |
|---|---|
| All nodes | 475 |
| Normal nodes | 234 |
| Stop nodes | 222 |
| Synchronizing nodes | 40 |
| Synchronizing/Stop nodes | 21 |

**Table 5.2:** Number of nodes after part II of the reduction algorithm

Algorithms 5.2 and 5.3 describe the functionality of this part of the graph reduction algorithm.

### Part III: Check the synchronizing nodes

The third part of the graph reduction algorithm checks, if the synchronizing nodes found in part two are so-called *leader synchronizing nodes* or *leaders* for short. Only leader synchronizing nodes are required for synchronizations between trains and the calculation of their optimal driving strategy.

**Definition 5.1** (Leader Synchronizing Node, Leaders)**.** *If there exists a chain of synchronizing nodes with exactly the same synchronization conditions, the last node of the chain is called* leader synchronizing node*. Isolated synchronizing nodes (with no direct connection to other synchronizing nodes) are leader synchronizing nodes, too.*

Assume a graph consisting of several nodes, but without synchronizing nodes. There may exist several paths between the entry and the final node. As there are no synchronizing nodes between the entry node and the final node, the movements of the involved trains are independent (no common track section used). As a result, all paths between the entry node and the final node are equal in a way that each train has the same movements (in the same order) on each path, without any influence on other trains.

Now assume that there exists exactly one synchronization node within the graph. In such a situation, all paths between the entry node and the synchronizing node and on the other hand

**Algorithm 5.2:** Find the synchronizing nodes

```
1  not_finished ← true;
2  while not_finished do
3      node ← task_queue.Dequeue;
4      if node ∉ task_nodes_up then
5          task_nodes_up ← task_nodes_up ∪ node;
6      end
7      if node ∈ task_nodes and node ∉ visited_nodes then
8          visited_nodes ← visited_nodes ∪ node;
9          node_type ← isSynchronizingNode(node);
10         if node_type ≠ NODE_NORMAL and node ∉ task_synchronizing_nodes then
11             task_synchronizing_nodes ← task_synchronizing_nodes ∪ node;
12             task_synchronizing_queue.Enqueue(node);
13             if node_type = NODE_STOP or node_type = NODE_SYNCH_STOP then
14                 if node ∉ task_stop_nodes then
15                     task_stop_nodes ← task_stop_nodes ∪ node;
16                 end
17             end
18         end
19         foreach predecessor in node.Predecessors do
20             if predecessor ∈ global_nodes then
21                 task_queue.Enqueue(predecessor);
22             end
23         end
24     end
25     if AllQueuesEmpty then
26         not_finished:=false;
27     end
28 end
```

all paths between the synchronizing node and the final node are the same. These idea can be extended to a graph containing several synchronizing nodes.

If there exists a chain of synchronizing nodes, where each of them has the same synchronization condition(s), only the last synchronizing node is relevant, because each path which ends at the last synchronizing node of the chain, will have the same movements of the trains and the last synchronization will be done definitely. If a path contains several synchronizing nodes with the same synchronization conditions and each synchronization is done, the last one will have an effect on the resulting driving strategy, independently from other synchronizing nodes in the chain. Thus, all synchronizing nodes, except the last one of the chain, can be handled as normal nodes. As a consequence, all isolated synchronizing nodes and the last synchronizing nodes

**Algorithm 5.3:** Check the synchronizing nodes: $isSynchronizingNode$

```
1  result ← NODE_NORMAL;
2  foreach predecessor in node.Predecessors do
3      if node ∈ global_nodes then
4          if node.t_departure > 0 then
5              result ← NODE_STOP;
6          end
7      end
8  end
9  foreach successor in node.Successors do
10     if successor ∈ global_nodes then
11         label_successor ← EdgeLabelSection(node,successor);
12         if label_successor = p then
13             foreach predecessor in node.Predecessors do
14                 if predecessor ∈ global_nodes then
15                     label_predecessor ← EdgeLabelSection(predecessor,node);
16                     if label_predecessor = v then
17                         if label_successor = label_predecessor then
18                             if result = NODE_STOP then
19                                 result ← NODE_SYNCH_STOP;
20                             else
21                                 result ← NODE_SYNCH;
22                             end
23                             CreateSynchronizationCondition;
24                         end
25                     end
26                 end
27             end
28         end
29     end
30 end
```

of a chain will be handled as leaders, the others will be handled as non-synchronizing nodes. The entry node and the final node are generally handled as synchronizing nodes, in particular as leader synchronizing nodes. As stop nodes are a special kind of synchronizing nodes, this part of the algorithm is applied in the same way for both synchronizing nodes and stop nodes. Due to simplification only the term synchronizing nodes will be used in the explanation.

**Example 5.2.** *Figure 5.6 shows an example with three synchronizing nodes, namely node* $5$*,* $8$*, and* $11$*. Train* $n$ *and train* $m$ *are involved in the synchronization on track section* $k$*. The three mentioned synchronizing nodes form a chain, where each node of them has the same synchro-*

*nization condition ($m \to n : k$). For future calculations, only the last node of the chain is relevant and remains as leader, whereas node 6 and 8 will further be handled as normal nodes.*

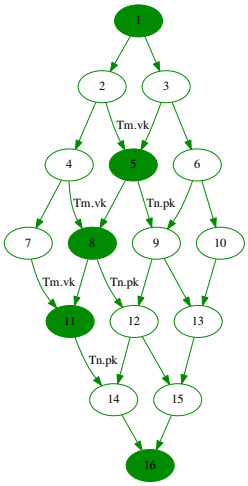



**Figure 5.6:** Synchronizing nodes (Example 5.2)

*Figure 5.7 shows the modified graph of Figure 5.6, where the entry node, the final node, and node 11, which is the last node of the chain of synchronizing nodes remain as synchronizing nodes.*

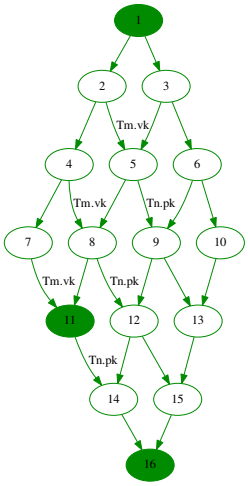



**Figure 5.7:** Leader synchronizing nodes (Example 5.2)

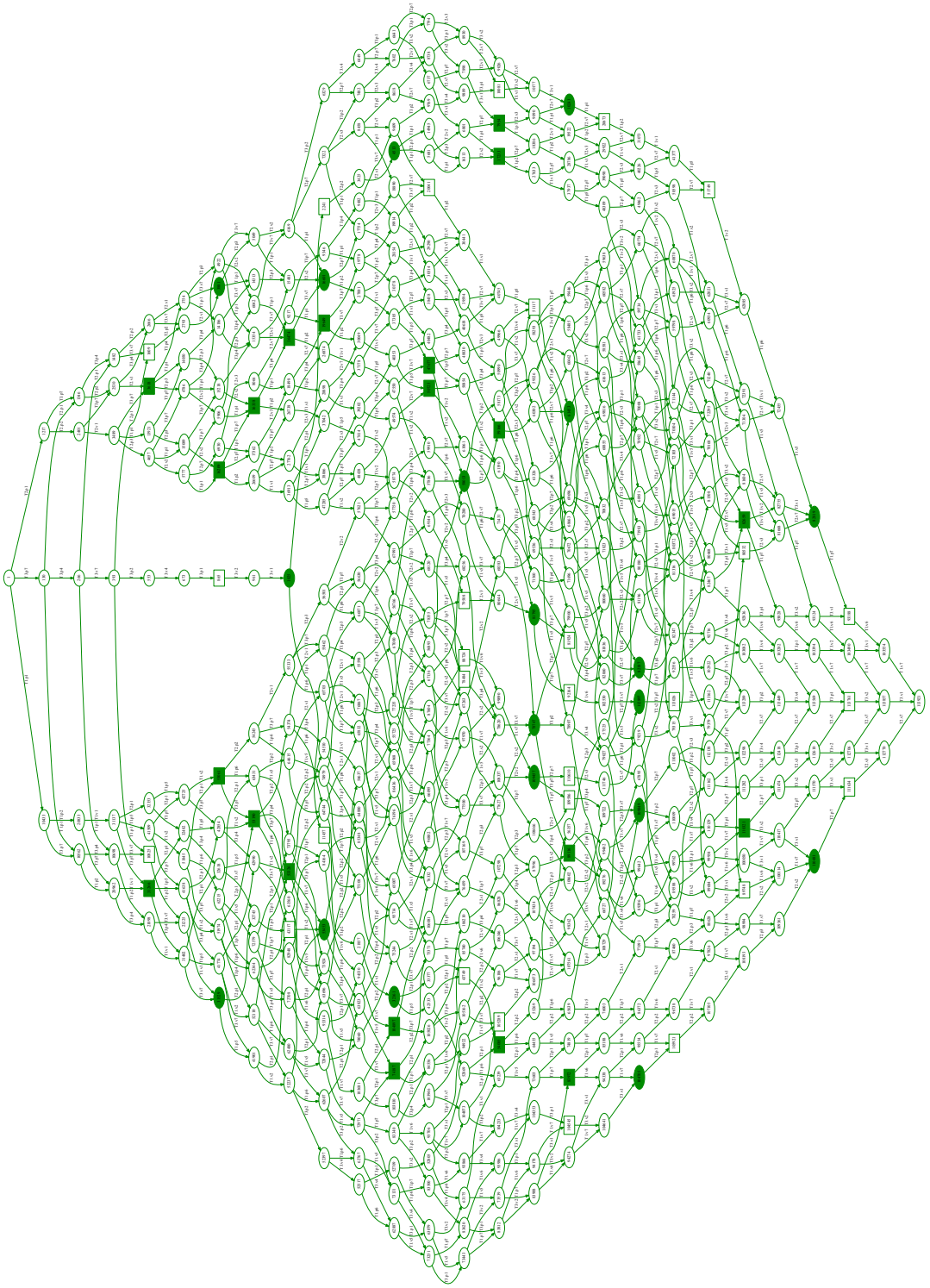


**Figure 5.8:** Resulting graph (Part III)

After executing this part of the algorithm, the relevant nodes for further calculations are determined, which consist of the start node, the leader synchronizing nodes, and the final node. The description of the algorithm can be found in Algorithms 5.4 and 5.5.

The resulting graph of the example (Figure 5.1) after determining the leader synchronizing nodes is illustrated in Figure 5.8.

| Node type | Amount (II) | Amount (III) |
|---|---|---|
| All nodes | 475 | 475 |
| Normal nodes | 234 | 407 |
| Stop nodes | 222 | 49 |
| Synchronizing nodes | 40 | 40 |
| Synchronizing/Stop nodes | 21 | 21 |

**Table 5.3:** Number of nodes after part III of the reduction algorithm

The number of nodes has been changed and a comparison to the previous part of the algorithm is shown in Table 5.3. It can be seen that the number of stop nodes has decreased from 222 to 49 nodes. In this example the number of synchronizing nodes remains the same but in general, it might shrink enormously. Due to the fact that only the synchronizing nodes and the stop nodes are of interest for further computations, this step reduces the complexity for the last part of the reduction algorithm.

**Algorithm 5.4:** Check the synchronizing nodes

```
1 while task_synchronizing_queue.Not_Empty do
2 |   node ← task_synchronizing_queue.Enqueue;
3 |   if isRealSynchronizingNode(node) then
4 |   |   real_synchronizing_nodes ← real_synchronizing_nodes ∪ node;
5 |   end
6 end
```

## Part IV: Find a path between the synchronizing nodes

In the previous parts of the algorithm, the graph was reduced and the leader synchronizing nodes were computed. The last step is to find paths between these nodes with the restriction that each incoming and outgoing edge, being part of the synchronization, must be used. The algorithm works as follows:

- The graph is traversed bottom-up from the final node to the entry node by taking nodes from the task-local queue. At each node, the node itself, the last visited leader synchronizing node, and the visited track sections in between the nodes are added to a task-local queue. The id of the task is again calculated by the hash function.

**Algorithm 5.5:** Check the synchronizing nodes: $isRealSynchronizingNode$

```
1  if node = entry_node or node = final_node then
2  |   return true;
3  end
4  contain_condition_all ← false;
5  foreach successor of node.Successors do
6  |   if successor ∈ global_synchronizing_nodes then
7  |   |   if successor ∈ global_nodes or successor in global_stop_nodes then
8  |   |   |   contain_condition ← true;
9  |   |   |   foreach synch of node.Synchronizing_Condition do
10 |   |   |   |   if synch ∉ successor.Synchronizing_Condition then
11 |   |   |   |   |   contain_condition ← false;
12 |   |   |   |   end
13 |   |   |   end
14 |   |   |   if contain_condition = true then
15 |   |   |   |   contain_condition_all ← true;
16 |   |   |   end
17 |   |   end
18 |   end
19 end
20 return not contain_condition_all;
```

- Each node is checked if it is a synchronizing node or a normal node.
- If the node is a synchronizing node, the additional information is updated (last visited synchronizing node and visited track sections). Thus the information of the leader synchronizing nodes contains the node itself and at least one synchronizing node, which can be reached from this node, including the track sections between them.
- If the node is a normal node, its predecessors must be examined. If a predecessor is not in the set of nodes from the first part, then it is not reachable from the entry node and thus not relevant for the deadlock-free situation and the reduced graph. On the other hand, if the node is in the set of nodes form the first part, the following situations must be considered:
  - If the predecessor is a real synchronizing node, then this path is not relevant.
  - If the predecessor is a normal node, then the path is relevant.
  - If the previously visited synchronizing node is the final node, then the path is relevant.
  - All the other paths are not relevant for calculating the optimal driving strategy.
- If a relevant path is found, the information from the last node (previously visited leader synchronizing node, the used track sections, and the current edge-label) is extended by the current track section $i$, if the edge label indicates entering or reserving a track section, indicated by $p_i$. If the entry node is reached, then the starting track section of the corresponding train is added to the node information, because this section is reserved initially

and thus has no $p$-operation in the graph. Finally, the node itself and its node information are added to the queue.

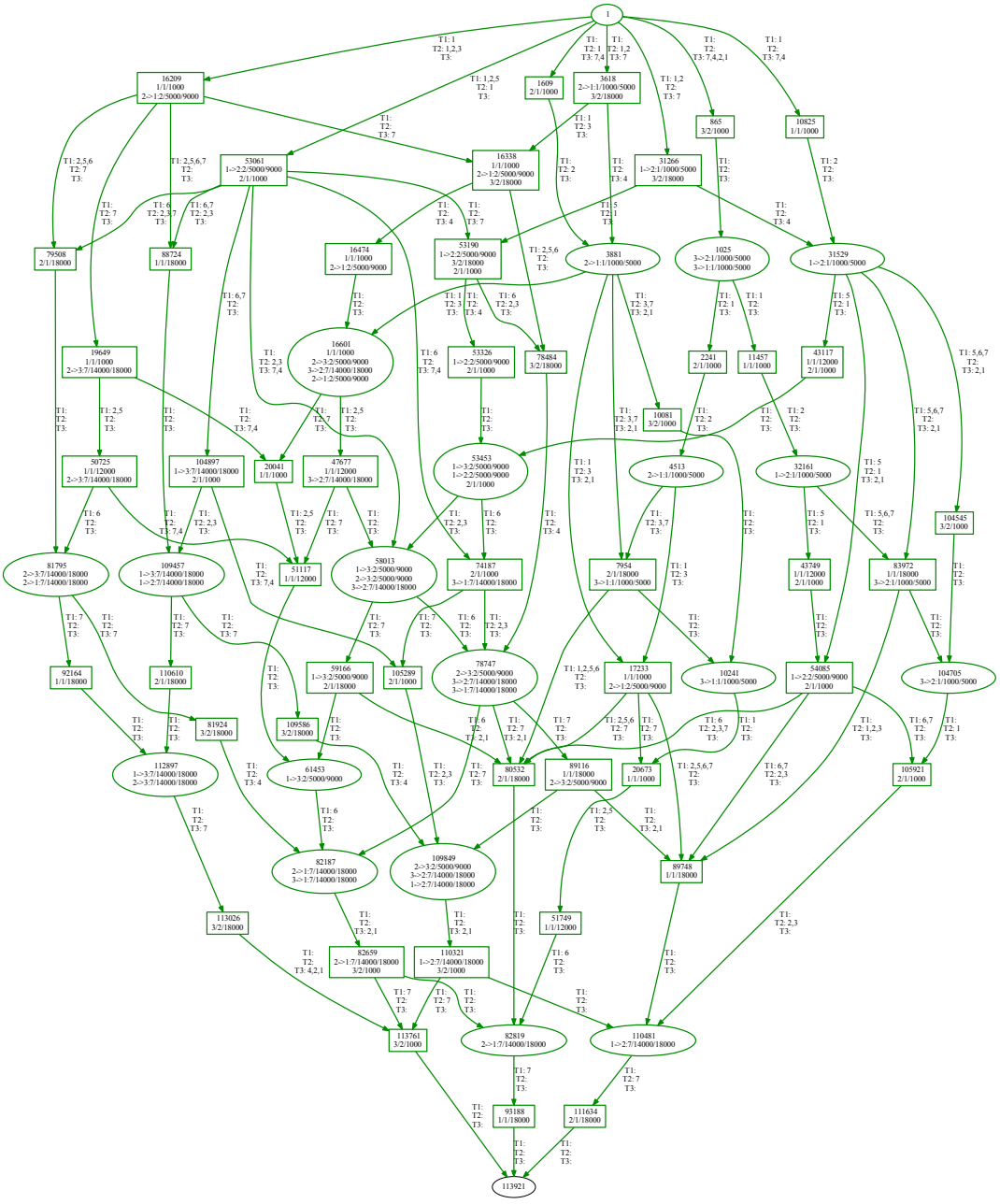


**Figure 5.9:** Resulting graph (Part IV)

When the resulting graph is created, it contains only the entry node, the leader synchronizing nodes, and the final node. Algorithm 5.6 shows the functionality of the fourth part of the graph reduction algorithm.

The reduced graph of the example is illustrated in Figure 5.9 and contains 70 nodes (49 stop nodes, 40 synchronizing nodes, 21 synchronizing/stop nodes, start node and final node). Nodes are labeled with their node-id and the synchronization conditions. Edges are labeled by the used track sections between two nodes.

**Example 5.3.** *Node* $109849$ *in Figure 5.9 is a* multi-synchronizing *node with the following synchronization conditions:*

- $2 \rightarrow 3 : 2$. *Additionally, the start position and the final position of track section* $2$ *are given:*
  - *Track section* $2$ *starts at position* $5\,000\ \mathrm{m}$ *for train* $T_2$.
  - *Track section* $2$ *ends at position* $9\,000\ \mathrm{m}$ *for train* $T_2$.
  - *Train* $T_3$ *is driving in the opposite direction and thus, the start and end position of the track section are inverted.*
- $3 \rightarrow 2 : 7$ *with boundaries of track section* $7$*:* $14\,000\ \mathrm{m}$ *and* $18\,000\ \mathrm{m}$.
- $1 \rightarrow 2 : 7$ *with the same boundaries of the track section as before.*

**Example 5.4.** *Node* $89116$ *in Figure 5.9 is a* stop-node *for train* $T_1$ *and a* synchronizing node*, too:*

- $1/1/18000$ *indicates that train* $T_1$ *has a stop at position* $18\,000\ \mathrm{m}$ *which is the end of its journey.*
- *Additionally, there is a synchronization at track section* $2$ *between train* $T_2$ *and* $T_3$.

**Example 5.5.** *The label between node* $74187$ *and* $78747$ *contains the following:*

- $T1 :$. *No track section is passed by train* $T_1$ *between these two nodes of the graph.*
- $T2 : 2, 3$. *Train* $T_2$ *passes track section* $2$ *and* $3$.
- $T3 :$. *No track section is passed by train* $T_3$.

Due to the reduction algorithm, only synchronizing nodes remain in the graph. For a better readability, the remaining nodes are not filled in Figure 5.9 as defined in Section 4.5.

**Algorithm 5.6:** Find a path between synchronizing nodes

```
1  not_finished ←true;
2  task_queue.Enqueue(final_node,final_node,∅,null);
3  while not_finished do
4      (node,prev,track_sections,edge_label) ← task_queue.Dequeue;
5      if node ∈ task_nodes and (node,prev,edge_label) ∉ visited_nodes then
6          visited_nodes ← visited_nodes ∪ (node,prev,edge_label);
7          if node ∉ real_synchronizing_nodes then
8              real_synchronizing_nodes.Update(node,prev,track_sections);
9              task_nodes.Enqueue(node,node,∅,null);
10         else
11             foreach predecessor in node.Predecessors do
12                 path_allowed ← false;
13                 if predecessor ∈ global_nodes then
14                     if predecessor ∈ real_synchronizing_nodes then
15                         path_allowed ← true;
16                     else
17                         if predecessor ∉ global_synchronizing_nodes then
18                             path_allowed ← true;
19                         else
20                             if prev = final_node then
21                                 path_allowed ← true;
22                             end
23                         end
24                     end
25                     if path_allowed then
26                         edge_label ← GetEdgeLabel(node,predecessor);
27                         if 'p' in edge_label then
28                             track_sections ← track_sections ∪ GetTrackSection(edge_label);
29                         end
30                         if predecessor = entry_node then
31                             track_sections ← track_sections ∪ GetInitialSections;
32                         end
33                         task_nodes.Enqueue(predecessor,prev,track_sections,edge_label);
34                     end
35                 end
36             end
37         end
38     end
39 end
```

## 5.3 Determine all possible routes

After the graph is reduced as described in Section 5.2, it remains to calculate all possible routes for each train. If trains have more than one possible route on their journey there exists several combinations of the routes:

$$R = \prod_{i=0}^{n} r_i \tag{5.2}$$

where $R$ is the number of combinations, $n$ is the number of trains, and $r_i$ is the number of routes for train $i$. There may exist some combinations which lead to deadlock situations. These combinations of routes must not be considered for the overall result. In the previous section the graph was created, deadlocks were eliminated, and the graph was reduced to the relevant nodes. So the resulting graph is the basis for further calculations without deadlocks. To find all possible combinations of the routes of all trains within the railway system, the reduced graph is traversed top-down, using a depth-first search algorithm. When the final node is reached, a route is found and the passed track sections of each train are added to a set of routes. Each route contains at least the entry node and the final node. Between these two nodes, there might be stop nodes, if the train has stops at railway stations or synchronizing nodes, if there is the demand of a synchronization on track sections between trains.

Due to the result of the application of Kronecker Algebra, there might be several paths between two synchronizing nodes with the same track sections. To ensure that each synchronization is done, all synchronizing and stop nodes, which are found while traversing the graph must be saved in a set for a particular route. Each combination of routes has its own set with a unique id, which is created by the track sections of the involved routes.

Assume that there is a railway system, including two trains $T_n$ and $T_m$ and the track sections $a$, $b$, $c$, and $d$. Each train can use two routes to reach their destination:

- $T_n$: $a \rightarrow b \rightarrow d$ and $a \rightarrow c \rightarrow d$
- $T_m$: $d \rightarrow b \rightarrow a$ and $d \rightarrow c \rightarrow a$

Based on equation 5.2, there exist four combinations of the routes.

- $a \rightarrow b \rightarrow d$ and $d \rightarrow b \rightarrow a$
- $a \rightarrow b \rightarrow d$ and $d \rightarrow c \rightarrow a$
- $a \rightarrow c \rightarrow d$ and $d \rightarrow b \rightarrow a$
- $a \rightarrow c \rightarrow d$ and $d \rightarrow c \rightarrow a$

Obviously, the first and the fourth item above will result in a deadlock and thus only the second and third train will be available for the calculation of the optimal driving strategies.

## 5.4 Finding the optimal driving strategy

In the previous section all possible routes were determined. The last step of the algorithm consists of finding the optimal solution in terms of energy. It is necessary to calculate the energy consumption for each route. A route might be partitioned into sub-tracks by stop nodes. Two types of sub-tracks can be distinguished:

- Sub-tracks without synchronization
- Sub-tracks with synchronization

### Sub-tracks without synchronization

A sub-track of a particular route of a train does not contain the demand of a synchronization if there exists no synchronization node between two stop nodes. As a result the train can drive undisturbed along its route and its driving strategy can be calculated as described in Chapter 3 (single trip optimization).

### Sub-tracks with synchronization

If there exists at least one synchronization node within a sub-track, the calculation of the optimal driving strategy is more complex than the single trip optimization. The following restrictions will be used in this chapter:

- Trains are denoted by $T$ and a particular train $j$ by $T^j$.
- In general, travel time values are denoted by $t$ and for a particular train $j$ by $t^j$. If the time value is related to a track section i, a subscript is used: $t_i$ and $t_i^j$ for train $j$ on track section $i$, respectively. To indicate the minimum travel time an underline is used, e.g. the minimum travel time for train $j$ on track section $i$ is denoted by $\underline{t}_i^j$. For the maximum travel time an overline is used, e.g. the maximum travel time for train $j$ on track section $i$ is denoted by $\overline{t}_i^j$.
- In general, the entering of track sections (e.g. section $i$) is denoted by $\rightarrow\boxed{\mathrm{i}}$ (as subscript) and the leaving of section $i$ by $\boxed{\mathrm{i}}\rightarrow$, respectively.

By using the notions above, the following definitions for trains are given:

**Definition 5.2** (Leaving trains)**.** *A train $j$, which is about to leave track section $i$ is denoted by* $T^j_{\boxed{i}\rightarrow}$.

**Definition 5.3** (Entering trains)**.** *A train $k$, which is about to enter a track section $i$ is denoted by* $T^k_{\rightarrow\boxed{i}}$.

In addition, some instants of time, which are used in the synchronization between trains must be defined:

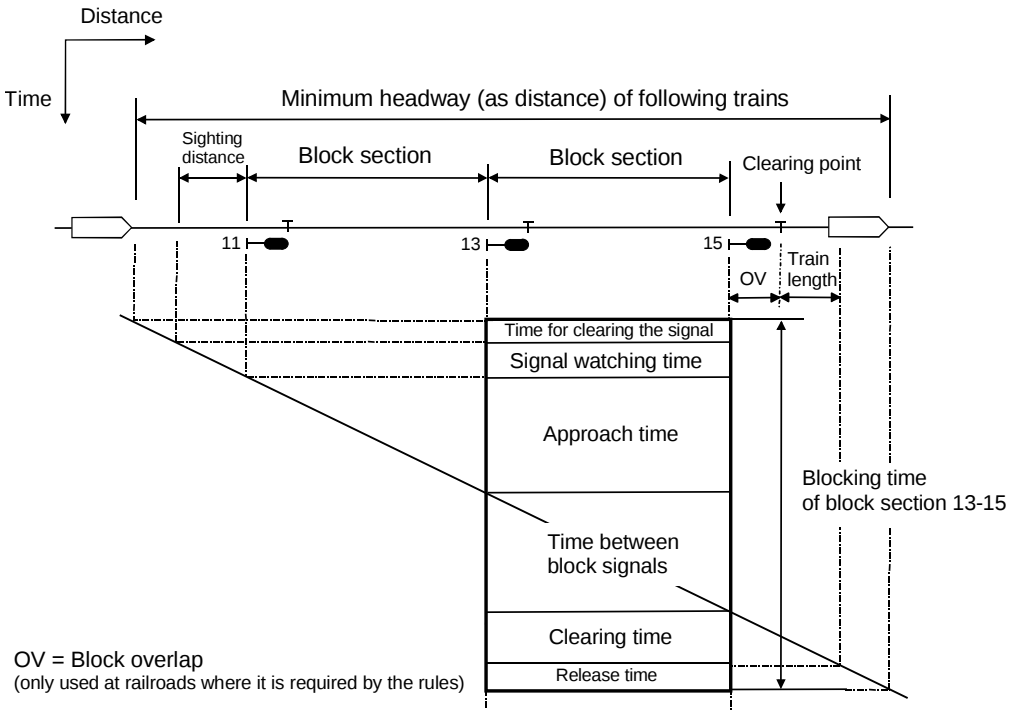



**Figure 5.10:** Blocking Time of a Block Section [62, 64]

**Definition 5.4** (Planned travel time (Extension of Definition 3.6))**.** *The planned travel time $t^k_{\circlearrowleft}$ for train $k$ on a particular journey is given by the schedule. A journey of a train consists of $n$ track sections. The planned travel time for train $k$ on a track section $i$ is given by $t^k_{i\circlearrowleft}$. That implies the overall travel time for train $k$ on its journey as*

$$t^k_{\circlearrowleft} = \sum_{i=1}^{n} t^k_{i\circlearrowleft} \tag{5.3}$$

**Example 5.6.** *Assume that the departure time is given as 03:00:00 o'clock and the arrival time is given as 03:05:30. The planned travel time is the difference between the departure and the arrival time and will be 00:05:30. All calculations will be done in seconds and thus the planned travel time is $t^k_{\circlearrowleft} = 330$ seconds.*

There exist some additional system behavior in railway systems, which must be considered in the calculations. Figures 5.10 and 5.11 show the blocking time of a block section and an example of a blocking stairway, respectively and will be used as base to determine the blocking time of each track section. It can be seen that the blocking time of a track section is longer than its occupation time and consists of the following parts [64]:

- *Time for clearing the signal.*
- The *Sighting distance* is illustrated at the left side on top of in Figure 5.10. It starts at the *point of view* (when the driver can see the distant signal) and ends at the position of the

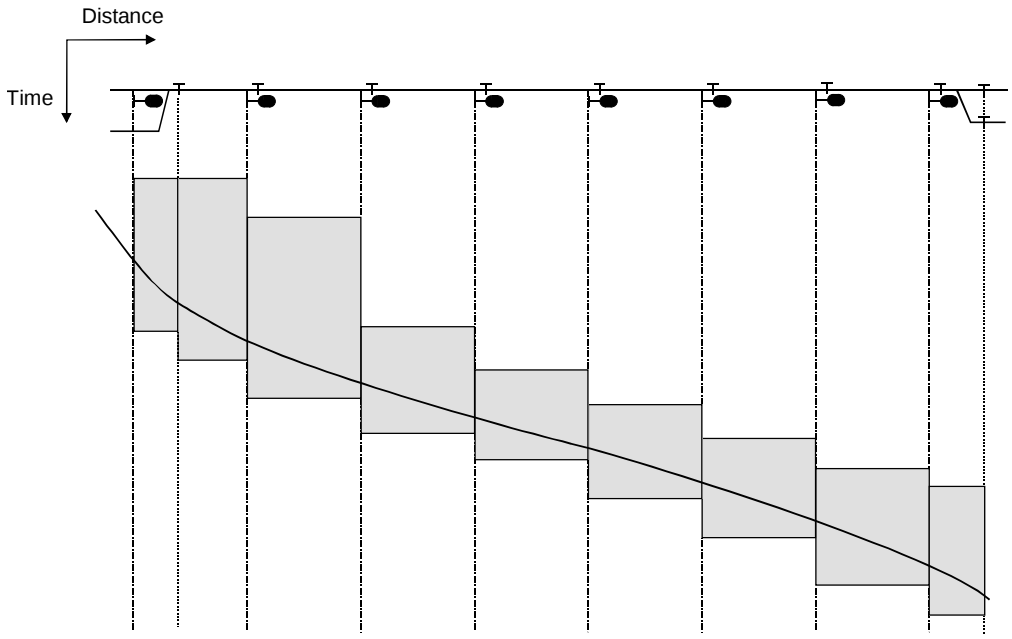



**Figure 5.11:** Blocking Time Stairway [62, 64]

distant signal. The time from seeing the signal until the train passes it, is called *Signal watching time*.

- The *Approach Time* gives the complete time while the train is between the distant signal and the beginning of the section.
- The time while the train is within the track section is called *Time between block signals*.
- The time between leaving a track section and completely passing a clearing point (e.g. all axis passed an axle counter) is called *Clearing time*.
- In the end an additional offset is needed for releasing the track section (*Release time*).

As a result a track section will be blocked some given time for clearing the signal before the train arrives the point of view and released some additional release time after the complete train has passed the clearing point (e.g. an axle counter). The following definitions are based on these assumptions and will be used for synchronization of trains using a common track section.

**Definition 5.5** (Blocking and releasing a track section)**.** *A track section $i$ is blocked by a train $k$ at $t^k_{\rightarrow\boxed{i}}$ and released at $t^k_{\boxed{i}\rightarrow}$. It follows that the complete blocking time $\boxed{t^k_i}$ of a track section $i$ by train $k$ is*

$$\boxed{t^k_i} = t^k_{\boxed{i}\rightarrow} - t^k_{\rightarrow\boxed{i}} \tag{5.4}$$

**Definition 5.6** (Earliest release time)**.** *The earliest instant of time for train $j$ to release a track section $i$ is denoted by $t^j_{\boxed{i}\rightarrow}$. This value is calculated in accordance to the schedule and indicates*

*the earliest instant of time when a train releases a track section, assuming that it arrives at its destination within the planned travel time.*

**Definition 5.7** (Latest release time)**.** *The latest instant of time for train $j$ to release a track section $i$ is denoted by $\overline{t}^{j}_{\boxed{i}\rightarrow}$. This value is calculated in accordance to the schedule and indicates the latest instant of time when a train releases a track section, assuming that it arrives at its destination within the planned travel time.*

**Definition 5.8** (Earliest reservation time)**.** *The earliest instant of time for train $k$ to reserve a track section $i$ is denoted by $\underline{t}^{k}_{\rightarrow\boxed{i}}$. This value is calculated in accordance to the schedule and indicates the earliest instant of time when a train reserves a track section, assuming that it arrives at its destination within the planned travel time.*

**Definition 5.9** (Latest reservation time)**.** *The latest instant of time for train $k$ to reserve a track section $i$ is denoted by $\overline{t}^{k}_{\rightarrow\boxed{i}}$. This values is calculated in accordance to the schedule and indicates the latest instant of time when a train reserves a track section, assuming that it arrives at its destination within the planned travel time.*

In contrast to the release and reservation time values which are calculated at the point of view or at the axle counter, regarding the time for clearing the signal and the release time, the following definition for the input and output speed values, which are needed for the calculation of the driving strategy, are defined at the beginning and the end of a track section, respectively.

**Definition 5.10** (Output speed (Extension of Definition 3.7))**.** *The speed value of train $j$ when leaving track section $i$ is called output speed, denoted by $v^{i}_{\boxed{j}\rightarrow}$. The highest and lowest output speed values of train $j$ when leaving a track section $i$ while satisfying the schedule are $\overline{v}^{i}_{\boxed{j}\rightarrow}$ and $\underline{v}^{i}_{\boxed{j}\rightarrow}$, respectively.*

**Definition 5.11** (Input speed (Extension of Definition 3.8))**.** *The speed value of train $j$ when entering track section $i$ is called input speed, denoted by $v^{i}_{\rightarrow\boxed{j}}$. The highest and lowest input speed values of train $j$ when entering a track section $i$ while satisfying the schedule are denoted by $\overline{v}^{i}_{\rightarrow\boxed{j}}$ and $\underline{v}^{i}_{\rightarrow\boxed{j}}$, respectively.*

*Obviously, $v^{j}_{\rightarrow\boxed{i}} = v^{j}_{\boxed{i\text{-}1}\rightarrow}$ for all $1 < i \leq n$, where $n$ is the number of track sections*[4]*. The first track section of a track or sub-track, starting at a railway station has an input speed value of 0 km/h and the last section of a journey, ending at a station has an output speed of 0 km/h. If the calculation starts while the train is driving along its journey, its current speed is used as input parameter of the calculation.*

**Definition 5.12** (Track section start and end position)**.** *A track section is bounded by its start position $s_{|i}$ and its end position $s_{i|}$.*

[4]Track section $i$ is the successor of section $i-1$ on the route.

Based on the definitions 5.5–5.9, the following equations must be valid to ensure that the train will arrive at its destination in time:

$$\underline{t}^{j}_{\boxed{\mathrm{i}}\rightarrow} \leq t^{j}_{\boxed{\mathrm{i}}\rightarrow} \leq \bar{t}^{j}_{\boxed{\mathrm{i}}\rightarrow} \tag{5.5}$$

$$\underline{t}^{k}_{\rightarrow\boxed{\mathrm{i}}} \leq t^{k}_{\rightarrow\boxed{\mathrm{i}}} \leq \bar{t}^{k}_{\rightarrow\boxed{\mathrm{i}}} \tag{5.6}$$

**Synchronizing of trains**

As already mentioned, trains must be synchronized if they want to use the same track section. In particular, a train can enter a track section only if it was already released by the other train.

**Example 5.7.** *Assume that there are two trains, namely $T_j$ and $T_k$, which want to use track section $i$. Based on the schedule the time values (in seconds) in Table 5.4 are calculated.*

| $T_j$ | $T_k$ | $T_{k'}$ |
|---|---|---|
| $\underline{t}^{j}_{\rightarrow\boxed{\mathrm{i}}} = 100$ | $\underline{t}^{k}_{\rightarrow\boxed{\mathrm{i}}} = 400$ | $\underline{t}^{k'}_{\rightarrow\boxed{\mathrm{i}}} = 320$ |
| $\bar{t}^{j}_{\rightarrow\boxed{\mathrm{i}}} = 150$ | $\bar{t}^{k}_{\rightarrow\boxed{\mathrm{i}}} = 450$ | $\bar{t}^{k'}_{\rightarrow\boxed{\mathrm{i}}} = 330$ |
| $\underline{t}^{j}_{\boxed{\mathrm{i}}\rightarrow} = 300$ | $\underline{t}^{k}_{\boxed{\mathrm{i}}\rightarrow} = 600$ | $\underline{t}^{k'}_{\boxed{\mathrm{i}}\rightarrow} = 520$ |
| $\bar{t}^{j}_{\boxed{\mathrm{i}}\rightarrow} = 350$ | $\bar{t}^{k}_{\boxed{\mathrm{i}}\rightarrow} = 650$ | $\bar{t}^{k'}_{\boxed{\mathrm{i}}\rightarrow} = 530$ |

**Table 5.4:** Time values for synchronization

*Track section $i$ is blocked by train $T_j$ in the worst case from* 100 s *to* 350 s*. As train $T_k$ will enter track section $i$ at the earliest after* 400 s*, no synchronization is needed between these two trains. Now, assume train $T_{k'}$ is used instead of $T_k$. Now, there is an overlap between the two instants of time of train $T_j$ for leaving the section and $T_{k'}$ entering the track section. As a result $T_j$ must have left track section $i$ at the latest after* 330 s*. Otherwise train $T_{k'}$ will not be in time at its destination.*

The algorithm is designed to find all solutions where all conflicts are avoided and the trains arrive at their destination within the planned travel time. Now assume that train $T_k$ wants to enter a track section, which is blocked by $T_j$. Based on these assumptions the following equation must be valid for $T_k$

$$\underline{t}^{k}_{\rightarrow\boxed{\mathrm{i}}} > \underline{t}^{j}_{\boxed{\mathrm{i}}\rightarrow} \tag{5.7}$$

and on the other hand for $T_j$

$$\bar{t}^{j}_{\boxed{\mathrm{i}}\rightarrow} < \bar{t}^{k}_{\rightarrow\boxed{\mathrm{i}}} \tag{5.8}$$

The set $\mathcal{S}$ contains all trains which are part of the synchronization. Based on the above restrictions, the previously defined instants of time must be slightly modified:

$$\underline{t}^{k*}_{\rightarrow\boxed{\mathrm{i}}} = \max_{j\in\mathcal{S},\, j\neq k} \left(\underline{t}^{k}_{\rightarrow\boxed{\mathrm{i}}}, \underline{t}^{j}_{\boxed{\mathrm{i}}\rightarrow}\right) \tag{5.9}$$

$$\bar{t}^{j*}_{\boxed{\mathrm{i}}\rightarrow} = \max_{j\in\mathcal{S},\, j\neq k} \left(\bar{t}^{k}_{\rightarrow\boxed{\mathrm{i}}}, \bar{t}^{j}_{\boxed{\mathrm{i}}\rightarrow}\right) \tag{5.10}$$

$\bar{t}^{k}_{\rightarrow\boxed{\mathbf{i}}}$ and $\underline{t}^{j}_{\boxed{\mathbf{i}}\rightarrow}$ remain unchanged:

$$\bar{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}} = \bar{t}^{k}_{\rightarrow\boxed{\mathbf{i}}} \tag{5.11}$$

$$\underline{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow} = \underline{t}^{j}_{\boxed{\mathbf{i}}\rightarrow} \tag{5.12}$$

For each synchronization between train $j$ and $k$ on track section $i$, these four time values ($\underline{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}}$, $\bar{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}}$, $\underline{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow}$, $\bar{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow}$) are relevant. Only solutions where no overlap between the instants of time for entering and leaving a track section are allowed to be taken into account for finding the optimal solution in terms of minimal energy consumption. Before analyzing the synchronizing nodes, these time values are computed for every pair of trains involved in a synchronization condition.

### Analysis of synchronizing nodes

The next step is to analyze the synchronizing nodes of each route. Three types of routes can be distinguished:

- No synchronization between trains is needed.
- No synchronization between trains is possible.
- Synchronization between trains is needed.

**No synchronization needed.** There are several situations where no synchronization between trains is needed:

- A track section is used by a single train.
- A track section $i$ is used by more than one train, but there is no overlap of the blocking time of a leaving train $j$ and an entering train $k$, written as

$$\underline{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}} > \bar{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow} \tag{5.13}$$

If no synchronization is needed, the corresponding synchronization condition is removed from the synchronizing node. If all synchronization conditions are removed from a node, the node itself is removed from the route. Due to the schedule it might be possible that all synchronizing nodes are removed from a route and thus, only the entry node and the final node will remain and the optimal driving strategy is calculated by the single trip optimization from Chapter 3.

**No synchronization possible.** A synchronization between two trains on a particular track section $i$ is not possible if the following equation is valid:

$$\underline{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow} > \bar{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}} \tag{5.14}$$

If a synchronization is not possible, then at least one train would not be able to reach its destination within the planned travel time. As the algorithm considers only results with a punctual arrival at each destination for the calculation of the driving strategies, the route is not taken into account for the overall result.

**Synchronization is needed.** If the following conditions are valid, a synchronization on track section $i$ between the leaving train $j$ and the entering train $k$ is possible and the node and its synchronization condition will remain unchanged:

$$\underline{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}} \leq \bar{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}} \tag{5.15}$$

$$\underline{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow} \leq \bar{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow} \tag{5.16}$$

$$\underline{t}^{k*}_{\rightarrow\boxed{\mathbf{i}}} \leq \bar{t}^{j*}_{\boxed{\mathbf{i}}\rightarrow} \tag{5.17}$$

There are several situations, which may occur after the instants of time are calculated and their relations are analyzed:

- If all routes are removed, it is not possible that the trains will reach their destinations in time. Possible solutions are adaptation of the time table or reassignment of the routes (e.g. a train might use an alternative route).
- If all routes consist of the entry node and the final node only, there will be no influence between all trains within the railway network and thus the optimal driving strategy for each train can be calculated independently from each other.
- For all routes consisting of at least one synchronizing node, the optimal driving strategy is calculated as described in the following section.

### Determining the optimal driving strategy

Each node within the resulting routes must be analyzed. For this reason the nodes put in ascending order by their node ids. As a result, the positions which are part of the synchronization conditions are also ordered correctly. The stop nodes divide the complete complete track into sub-tracks. Due to the synchronization nodes, the sub-tracks are divided into several parts.

If a train is involved in a synchronization condition, the node, in particular the track section from the synchronization condition is relevant for the train. Based on the Definitions 5.6–5.11, the relevant time and speed values for the corresponding track section are calculated and saved in a vector together with the synchronization position. For a train entering a track section the start position of the track is used as synchronization position and for a leaving train, the end position is used, respectively. For each route and within the route for each train a separate vector is used to save the data. This procedure is repeated for each node of the route until the final node is reached. If available, the next route is analyzed in the same way. If all routes and their nodes are analyzed, the optimal driving strategies can be calculated.

The determination of the optimal driving strategy is done for each route and in each route for each train. So the calculation starts with the first train of the first route. Due to the fact that the complete track is split up into several parts and each part can have several output speed values, in combination with several travel time values, a recursive algorithm will calculate several driving strategies for the overall result. The algorithm calculates the optimal driving strategy between two positions, taken from the vector. The input speed of the current calculation is based on the previous calculation, in other words, the input speed is the output speed of the previous calculation. The output speed can vary between the minimum and maximum output speed value (see Definitions 5.10). The travel time is again based on the previous calculation and can vary between the minimum and maximum travel time values (see Definition 5.6–5.9). The algorithm from Chapter 3 calculates the optimal driving strategy for the given input parameters. If a solution is found, the calculated output values are taken as input parameters for the next part. This procedure is repeated until the last element of the vector is reached. Then the calculation for the next train within the system starts and when all computations of all trains are finished, the same procedure is done for the next route.

The instant of time for entering a track section is the sum of the travel time of each previously passed track section $\sum_{j=1}^{i-1} \Delta t_j^k$, the instant of time for leaving the section is bounded by $\underline{t}^{j*}_{\boxed{\text{i}}\rightarrow}$ and $\bar{t}^{j*}_{\boxed{\text{i}}\rightarrow}$. As a consequence, the following equation must be valid for the travel time on track section $i$:

$$\underline{t}^{k*}_{\boxed{\text{i}}\rightarrow} - \sum_{j=1}^{i-1} \Delta t_j^k \leq t_i^k \leq \bar{t}^{k*}_{\boxed{\text{i}}\rightarrow} - \sum_{j=1}^{i-1} \Delta t_j^k \tag{5.18}$$

and thus

$$\Delta \underline{t}_i^k = \underline{t}^{k*}_{\boxed{\text{i}}\rightarrow} - \sum_{j=1}^{i-1} \Delta t_j^k \tag{5.19}$$

$$\Delta \bar{t}_i^k = \bar{t}^{k*}_{\boxed{\text{i}}\rightarrow} - \sum_{j=1}^{i-1} \Delta t_j^k \tag{5.20}$$

The travel time of a train $k$ on track section $i$ is bounded by $\Delta \underline{t}_i^k$ and $\Delta \bar{t}_i^k$.

Assuming $\Delta \underline{t}_i^k = 180\,\text{s}$ and $\Delta \bar{t}_i^k = 210\,\text{s}$, the best result in terms of energy consumption may be between these two values. Due to restrictions on the calculation time, it might not be possible or efficient to calculate the optimal driving strategy for each travel time value between these values, e.g. for 180 s, 181 s, 182 s and so on. There exits an algorithm parameter to adjust the time offset between two travel time values $\Delta t_+$ in seconds. Thus the algorithm will calculate the optimal driving strategy for

$$\Delta \underline{t}_i^k, \Delta \underline{t}_i^k + \Delta t_+, \Delta \underline{t}_i^k + 2 \cdot \Delta t_+, \Delta \underline{t}_i^k + 3 \cdot \Delta t_+ \ldots \Delta \bar{t}_i^k.$$

As already mentioned, an input and output speed exists for each track section. The input speed is fixed, as it is equal to the output speed of the previous section ($v^j_{\rightarrow\boxed{\text{i}}} = v^j_{\boxed{\text{i-1}}\rightarrow}$). To

find the optimal driving strategy, the output speed of each part of the track may be between the minimum and maximum output speed value. Thus the algorithm will use several speed values between these borders. The step size of these speed values can be adjusted by the algorithm parameter $\Delta v_+$ in $\mathrm{km/h}$. As a result, the output speed for the calculation of the driving strategy for train $j$ on track section $i$ is

$$\underline{v}^i_{\boxed{\mathrm{j}}\rightarrow}, \underline{v}^i_{\boxed{\mathrm{j}}\rightarrow} + \Delta v_+, \underline{v}^i_{\boxed{\mathrm{j}}\rightarrow} + 2 \cdot \Delta v_+, \underline{v}^i_{\boxed{\mathrm{j}}\rightarrow} + 3 \cdot \Delta v_+ \ldots \overline{v}^i_{\boxed{\mathrm{j}}\rightarrow}.$$

To sum up the functionality of the algorithm, the calculation of the overall result works as follows:

- Start at the first route.
  - Start at the first train with the following recursive algorithm.
    - Start at the first part of the track with a given input speed (e.g. $0\ \mathrm{km/h}$, when starting at a railway station).
    - Calculate the driving strategies for all combinations of possible travel times and output speed values and use the calculated values as input for the next part.
    - Repeat until the last position of the route is reached.
  - Repeat until the driving strategies for all trains have been calculated.
- Repeat until all routes have been analyzed.

Thus, the algorithm calculates various driving strategies for each train. The last step of the optimization algorithm is to find the best solution in terms of energy consumption. For this reason all combinations of the results of the involved trains must be analyzed. As there are several instants of time for entering and leaving a track section, an overlap between a train entering and another one leaving a track section may occur. Such results are not valid and must be discarded from the set of results. Consequently, the results must be checked, if they fulfill the following restrictions:

- Each train is within its schedule.
- A track section can be used only if it is not occupied by another train. In other words, train $k$ can enter a track section $i$ only, after it was released by another train $j$: $\underline{t}^{k*}_{\rightarrow\boxed{\mathrm{i}}} \geq \underline{t}^{j*}_{\boxed{\mathrm{i}}\rightarrow}$ for each track section $i$, where train $j$ and $k$ are involved in the synchronization.
- $E \rightarrow \min$, where $E$ is the complete energy consumption of all trains.

The algorithm to determine the optimal driving strategy is designed to run on multi-core CPUs. The calculation of the energy demand for each driving strategy (with different travel time and output speed values) can be easily parallelized, as the calculations are independent from each other.

It can be easily seen, that the execution time of the algorithm grows exponentially in the number of track sections. However, the model behaves well for practical situations, e.g. a small number of trains (two in almost all cases) and a small number of track sections or for systems with less synchronizations between trains.

**Example 5.8.** *Assume the schedule from Table 5.5 for three trains in the given railway system (Figure 5.1, Example 5.1).*

| **Train** | **Position** [m] | **Arrival** | **Departure** | **Traveltime** [s] |
|---|---|---|---|---|
| $T_1$ | 1 000 | | 09:23:00 | |
| | 12 000 | 09:31:30 | 09:41:30 | 510 |
| | 18 000 | 09:48:30 | | 420 |
| $T_2$ | 1 000 | | 09:01:00 | |
| | 18 000 | 09:14:00 | | 780 |
| $T_3$ | 18 000 | | 09:27:30 | |
| | 1 000 | 09:40:30 | | 780 |

**Table 5.5:** Schedule for $T_1$, $T_2$, and $T_3$ (Example 5.8)

*The routes of the trains read as follows:*

- *Train* 1 *(*$T_1$*):* $1 \to 2 \to 5 \to 6 \to 7$. *Train* $T_1$ *starts at position* 1 000 m *at track section* 1*, has a stop at position* 12 000 m *at the end of section* 5 *and will finish its journey at position* 18 000 m *at track section* 7*.*

- *Train* 2 *(*$T_2$*):* $1 \to 2 \to 3 \to 7$. *Train* $T_2$ *starts at position* 1 000 m *at track section* 1 *and finishes its journey at position* 18 000 m *at section* 7 *without any further stop.*

- *Train* 3 *(*$T_3$*):* $7 \to 4 \to 2 \to 1$. *Train* $T_3$ *starts its journey at position* 18 000 m *(track section* 4*) and arrives at its destination at position* 1 000 m *at track section* 1*.*

*Train* $T_1$ *is driving on the main track and the allowed track speed is* $\overline{v} = 140$ km/h *for the complete track. The other trains are driving over switches and thus, there is a reduced maximum speed. The speed restrictions for train* $T_2$ *and* $T_3$ *can be found in Tables 5.6 and 5.7. The track inclination is given in Table 5.8 and is the same for each route. The three trains are identical in their train parameters which are given in Table 5.9.*

*As this example uses the same routes and track section for the involved trains, the resulting graph is the same as in Example 5.1 (Figure 5.3). The reduced graphs are illustrated in Figures 5.5, 5.8, and 5.9. Based on the departure and arrival times of the trains, obviously no synchronization on the track sections is needed (Figure 5.13) and thus, each train can perform its optimal driving strategy. The resulting energy consumption for each train is given in Table 5.13 and the driving strategies are illustrated in Figure 5.12.*

| **From** [m] | **To** [m] | $\overline{v}$ $[km/h]$ |
|---|---|---|
| 1 000 | 8 500 | 140 |
| 8 500 | 9 000 | 40 |
| 9 000 | 14 500 | 140 |
| 14 500 | 15 000 | 40 |
| 15 000 | 18 500 | 140 |

**Table 5.6:** $\overline{v}$ for $R_2$ (Example 5.8)

| **From** [m] | **To** [m] | $\overline{v}$ $[km/h]$ |
|---|---|---|
| 1 000 | 8 000 | 140 |
| 8 000 | 8 500 | 40 |
| 8 500 | 14 000 | 140 |
| 14 000 | 14 500 | 40 |
| 14 500 | 18 500 | 140 |

**Table 5.7:** $\overline{v}$ for $R_3$ (Example 5.8)

| **From** [m] | **To** [m] | **Grad.** [‰] | **From** [m] | **To** [m] | **Grad.** [‰] |
|---|---|---|---|---|---|
| 1 000 | 1 480 | 1 | 1 480 | 1 770 | −3 |
| 1 770 | 1 921 | −1 | 1 921 | 2 100 | 0 |
| 2 100 | 2 500 | 5 | 2 500 | 2 700 | 4 |
| 2 700 | 3 300 | 5 | 3 300 | 3 800 | 4 |
| 3 800 | 4 500 | 5 | 4 500 | 5 000 | 4 |
| 5 000 | 5 200 | 2 | 5 200 | 5 900 | 5 |
| 5 900 | 6 200 | 8 | 6 200 | 6 300 | 4 |
| 6 300 | 6 500 | 7 | 6 500 | 6 900 | 3 |
| 6 900 | 7 700 | 2 | 7 700 | 7 900 | 0 |
| 7 900 | 8 100 | 5 | 8 100 | 8 200 | 4 |
| 8 200 | 8 300 | 5 | 8 300 | 8 400 | 3 |
| 8 400 | 8 500 | 1 | 8 500 | 9 200 | 0 |
| 9 200 | 9 500 | 5 | 9 500 | 9 700 | 3 |
| 9 700 | 9 800 | 5 | 9 800 | 9 900 | 4 |
| 9 900 | 10 000 | 3 | 10 000 | 10 100 | 6 |
| 10 100 | 10 200 | 1 | 10 200 | 10 600 | 0 |
| 10 600 | 10 900 | −3 | 10 900 | 11 000 | 0 |
| 11 000 | 11 300 | 3 | 11 300 | 11 500 | 0 |
| 11 500 | 11 800 | −4 | 11 800 | 12 300 | −5 |
| 12 300 | 12 378 | −6 | 12 378 | 13 300 | −9 |
| 13 300 | 13 500 | −6 | 13 500 | 14 000 | −3 |
| 14 000 | 14 100 | 0 | 14 100 | 14 200 | −2 |
| 14 200 | 14 300 | −3 | 14 300 | 14 602 | −2 |
| 14 602 | 17 800 | −2 | 17 800 | 18 000 | −1 |

**Table 5.8:** Track inclination (Example 5.8)

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $m_T$ | $42\,000$ t | $m_W$ | $72\,000$ t |
| $n$ | 2 | $r$ | 0.85 |
| $F(v)$ | see Ex. 2.1 | $B(v)$ | see Ex. 2.2 |
| $l$ | 67 m | $f_L$ | 3.3 |
| $k_{St_1}$ | $0.03\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $k_{Sa_1}$ | $0.0025\ \mathrm{s/m}$ |
| $k_{Sa_2}$ | $0.00696\ \mathrm{kg} \cdot \mathrm{s}^2/\mathrm{m}^2$ | $\Delta v$ | $4.17\ \mathrm{m/s}$ |
| $\rho$ | 6.0 | | |
| $v_+$ | $5\ \mathrm{km/h}$ | $s_+$ | 100 m |

**Table 5.9:** Train and algorithm parameters (Example 5.8)

**Example 5.9.** *Now, assume that there is a new schedule with changed arrival and departure times for trains $T_2$ and $T_3$ (Table 5.10).*

| Train | Position [m] | Arrival | Departure | Traveltime [s] |
|---|---|---|---|---|
| $T_1$ | 1 000 | | 09:23:00 | |
| | 12 000 | 09:31:30 | 09:41:30 | 510 |
| | 18 000 | 09:48:30 | | 420 |
| $T_2$ | 1 000 | | 09:27:30 | |
| | 18 000 | 09:40:30 | | 780 |
| $T_3$ | 18 000 | | 09:29:30 | |
| | 1 000 | 09:42:30 | | 780 |

**Table 5.10:** Schedule for $T_1$, $T_2$, and $T_3$ (Example 5.9)

*Due to the new schedule, train $T_1$ and $T_2$ have to synchronize their access on track section* 2 *(*5 000 m–9 000 m*) which means that train $T_2$ has to wait until $T_1$ has left track section* 2*. In particular, train $T_2$ is allowed to pass position* $3\,600$ m[5] *after the complete train has passed position* $9\,167$ m*. The driving strategy of these two trains must be adopted which can be easily seen in Figure 5.14. As a result, the energy consumption of the two trains is increased (Table 5.13). The driving strategy for train $T_3$ is not influenced by the new schedule. Figure 5.15 illustrates the modified distance-time diagram for this example.*

[5]The explanation can be found in Section 5.4, Figure 5.10.

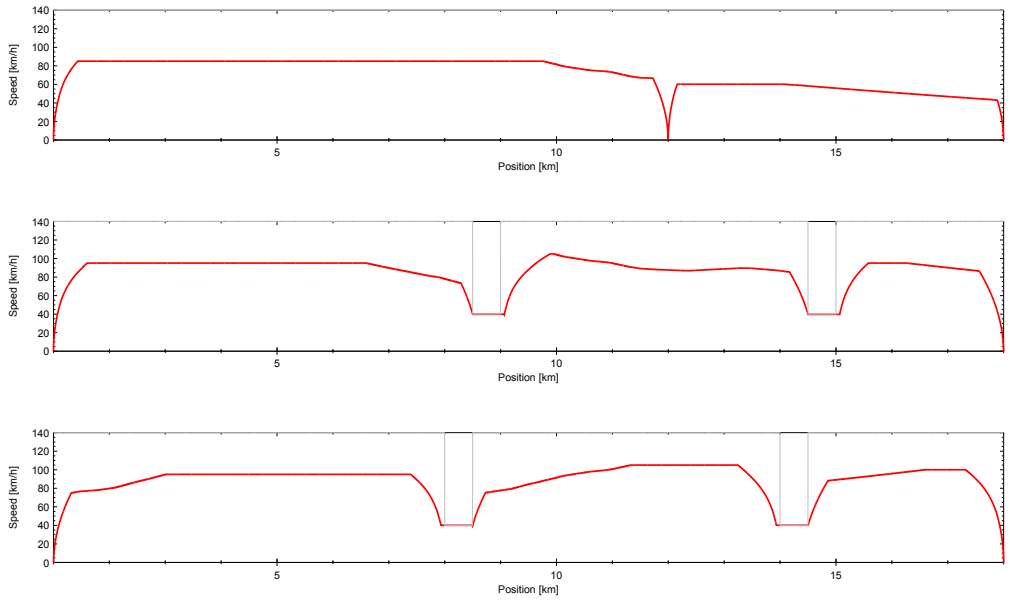



**Figure 5.12:** Optimal driving strategy (Example 5.8)

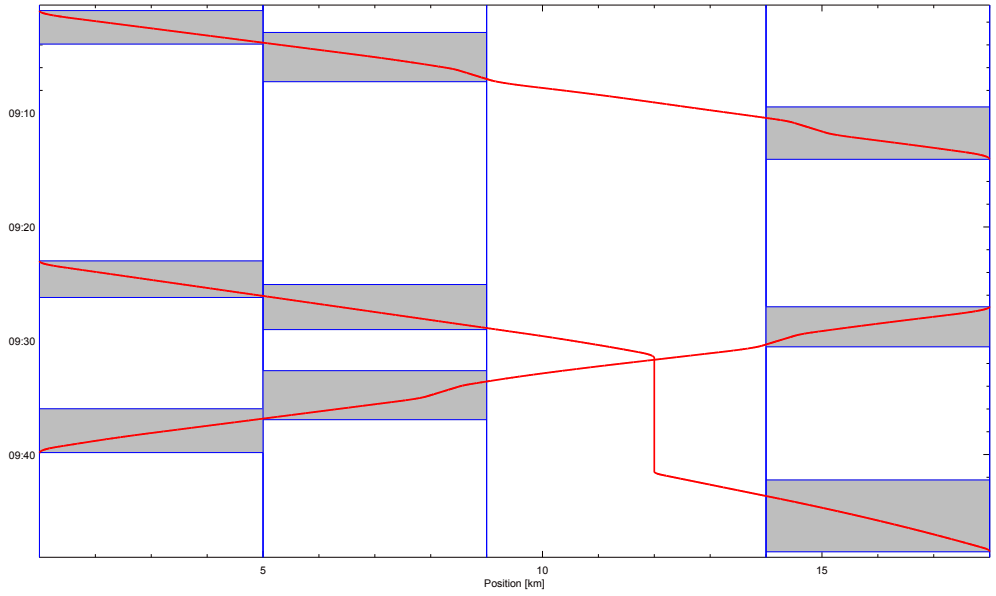



**Figure 5.13:** Distance-Time diagram (Example 5.8)

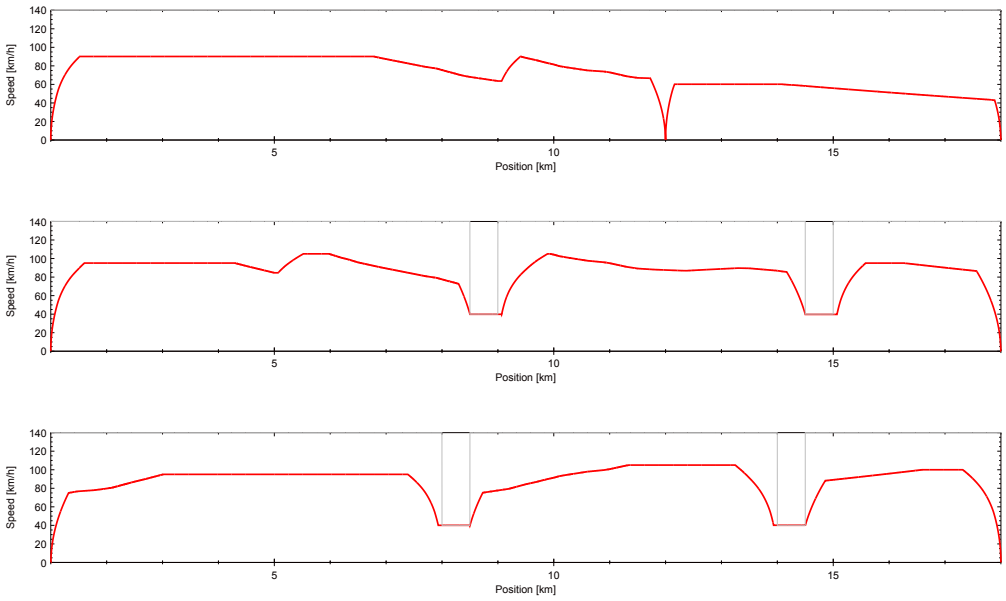



**Figure 5.14:** Optimal driving strategy (Example 5.9)

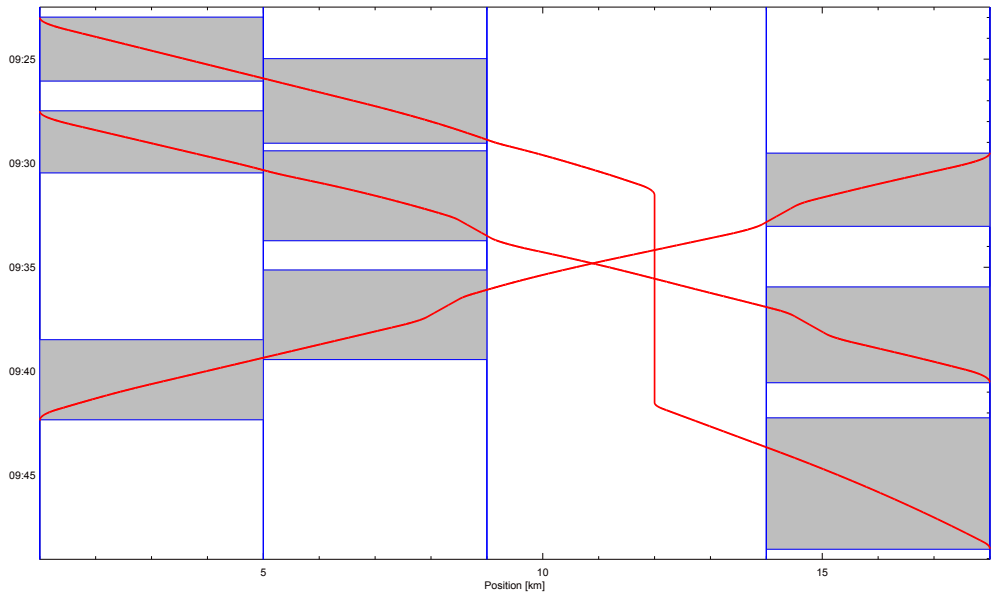



**Figure 5.15:** Distance-Time diagram (Example 5.9)

**Example 5.10.** *This example, again, will use the same infrastructure. The schedule is modified as shown in Table 5.11.*

| Train | Position [m] | Arrival | Departure | Traveltime [s] |
|---|---|---|---|---|
| $T_1$ | 1 000 | | 09:23:00 | |
| | 12 000 | 09:31:30 | 09:41:30 | 510 |
| | 18 000 | 09:48:30 | | 420 |
| $T_2$ | 1 000 | | 09:26:30 | |
| | 18 000 | 09:39:30 | | 780 |
| $T_3$ | 18 000 | | 09:27:30 | |
| | 1 000 | 09:40:30 | | 780 |

**Table 5.11:** Schedule for $T_1$, $T_2$, and $T_3$ (Example 5.9)

*The resulting driving strategies and the distance-time diagram are illustrated in Figure 5.16 and Figure 5.17, respectively. Obviously, the three trains have to do a synchronization on track section* 2 *as follows:*

- *Similar to the previous example, train $T_2$ has to wait until train $T_1$ has left track section* 2*.*
- *Train $T_3$ has to wait until train $T_2$ has left the track section.*

*As a consequence, train $T_1$ starts its journey with a higher hold speed within the first two track sections which results in a decreased travel time for these two section. Thus, the train has to be slower in the third track section to arrive at the railway station (position* $14\,000$ m*) at the planned arrival time (09:31:30). Train $T_2$ starts with a lower hold speed at the first track section and increases its speed in the second one to leave the section that train $T_3$ can use it in a way to be in time at the destination. As a result, both, train $T_2$ and $T_3$ have be drive faster within track section* 7 *and* 2 *and* 1*, respectively, to be punctually at their destination. Nevertheless, an optimal driving strategy is determined for each train, based on these restrictions. The resulting energy consumption for the trains is given in Table 5.13.*

**Example 5.11.** *Now, the schedule of train $T_3$ is modified as given in Table 5.12. As a consequence, it is not possible that the will arrive at its destination in time because the release of track section* 2 *is too late, even when train $T_2$ will drive with maximum speed.*

| Train | Position [m] | Arrival | Departure | Traveltime [s] |
|---|---|---|---|---|
| $T_1$ | 1 000 | | 09:23:00 | |
| | 12 000 | 09:31:30 | 09:41:30 | 510 |
| | 18 000 | 09:48:30 | | 420 |
| $T_2$ | 1 000 | | 09:26:30 | |
| | 18 000 | 09:39:30 | | 780 |
| $T_3$ | 18 000 | | 09:25:30 | |
| | 1 000 | 09:38:30 | | 780 |

**Table 5.12:** Schedule for $T_1$, $T_2$, and $T_3$ (Example 5.9)

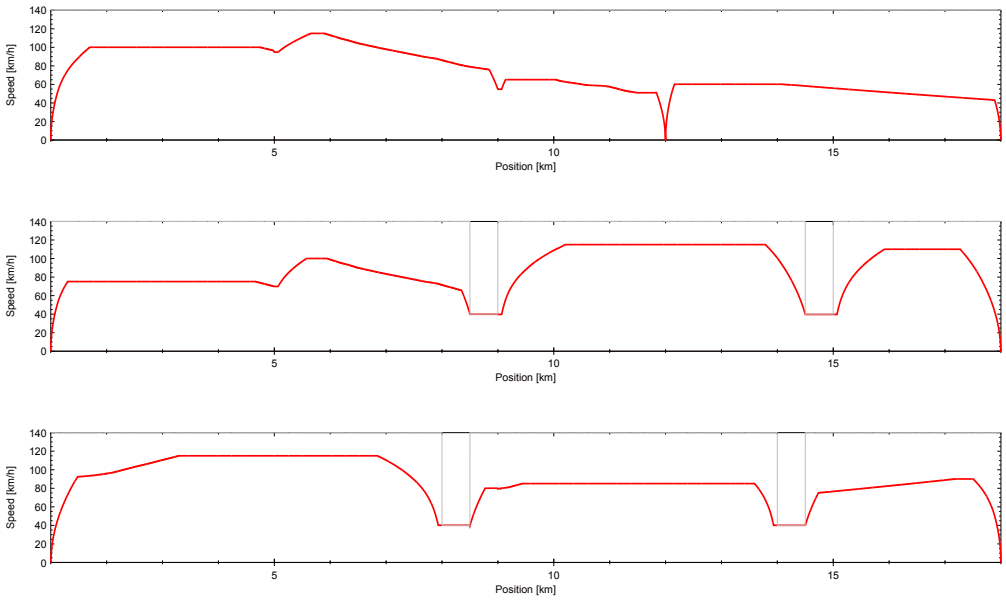



**Figure 5.16:** Optimal driving strategy (Example 5.10)

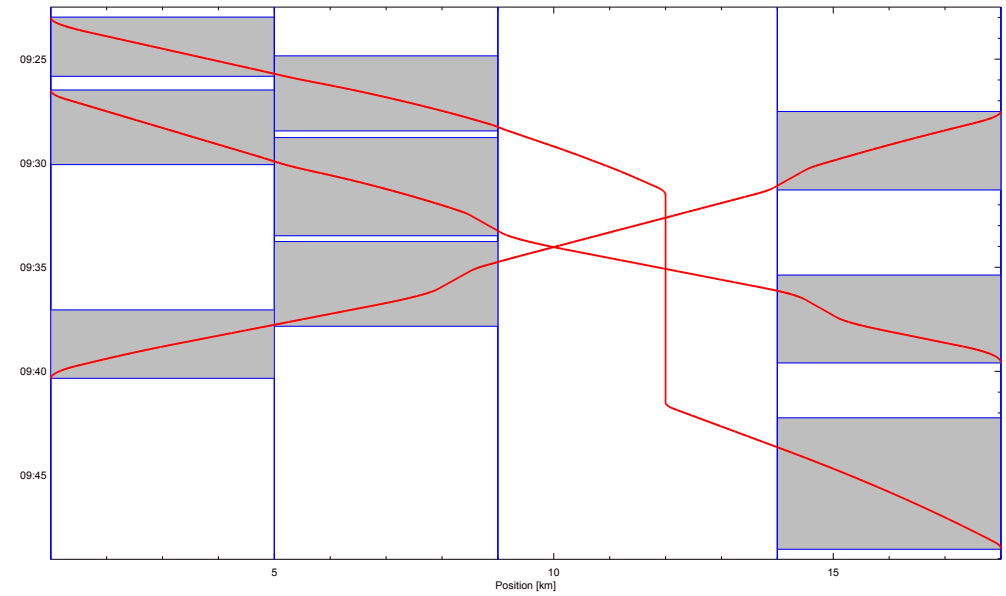



**Figure 5.17:** Distance-Time diagram (Example 5.10)

Table 5.13 gives an overview about the resulting energy consumption of the previous examples. Obviously, the energy consumption increases, depending on the degree of synchronization.

| **Train** | $e$ [kWh] **(Example 5.8)** | $e$ [kWh] **(Example 5.9)** | $e$ [kWh] **(Example 5.10)** |
|---|---|---|---|
| $T_1$ | 32.983 | 33.204 | 35.117 |
| $T_2$ | 41.157 | 41.241 | 44.666 |
| $T_3$ | 41.961 | 41.961 | 43.163 |
| $\sum$ | 116.101 | 116.406 | 122.946 |

**Table 5.13:** Energy consumption (Examples 5.8 – 5.10)

CHAPTER 6

# Extensions of the model

The presented model can act as base for further development in automation systems for railway systems. Some ideas for extending the model are introduced in this section.

## 6.1 Extension of the schedule

As already mentioned, the algorithm only calculates driving strategies for the trains, if they are all within the planned travel time. A possible extension could be to calculate driving strategies for unpunctual trains, too, assuming that the trains will be in time at another railway station. In particular, the schedule could be extended for intermediate stops to the end that the train can be in time at a defined final stop (cf. [39]).

Another approach to avoid conflicts on the track could be the modification of the departure time of the trains. Therefore, the arrival time of the trains will be fixed, but the departure time can be varied (e.g. the train must leave the railway station between 10:00:00 and 10:00:10). This may lead to a new driving strategy with less energy consumption because the train need not break and accelerate on the track, due to a blocked track section.

## 6.2 Train-priorities

The model can be extended by assigning priorities to trains. As a consequence, a train with a higher priority may be handled in a different way than trains with low priority, e.g. when entering track sections. The introduction of train-priorities may lead to unpunctuality for trains with low priority and thus the previous extension may be introduced together with the implementation of train-priorities.

## 6.3 Hard- and software-coupling

The algorithm may be coupled with some hardware parts and some dispatching or monitoring software. The advantage would be that some important characteristics of the network can be considered in the calculations. In particular:

- The current schedule can be considered. It might be possible that the schedule may be changed while operating due to delays, new train within the system or other factors that may have an influence on the dispatching.
- As the algorithm uses some track information (e.g. maximum allowed speed), it would be an advantage if this information can be updated online.
- If there is an energy bottleneck, some routes within the resulting graph may not be possible due to a high energy demand. Such routes can be eliminated if the current state of the power plants can be taken into account.
- Assume that the algorithm is implemented on a central computation unit and calculates the driving strategy for each train within the system. If the position of each train is know exactly at each instant of time in the system, it can be compared with the calculated driving strategy. If there is a deviation between these two values, the algorithm must be restarted with the new position (of each train). As a result, new driving strategies can be found for trains within the system, depending on the influence of the deviant train.

## 6.4 Automated train control

Due to the fact that each train driver has a reaction time longer than the reaction time of any machine, the best results can be achieved when the train control is done automatically, based on the results of the algorithm. As a result, deviations and thus, new calculations can be avoided.

In general, each coupling to hardware or software modules can improve the correctness and the performance of the algorithm and its results due to actuality of the used information.

CHAPTER 7

# Conclusion and prospect

This thesis is concerned with the development of an algorithm to determine the optimal driving strategy for each train within a railway network which is feasible in the field. Therefore the dependencies between the trains (e.g. for entering and leaving a track section) are analyzed and the results are used in the calculations for the driving strategies. Due to the fact that there are several restrictions regarding the feasibility in the field, the calculated driving strategy is optimal under these restrictions, but could theoretically be better. There exists a number of parameters to adjust the accuracy and the calculation time of the algorithm.

The algorithm is divided into two parts. The first one uses so-called Kronecker Algebra to determine all possible movements of the trains within the railway network. Conflicts (e.g. deadlocks or headway-conflicts) are found due to this mathematical model using matrices and operations on them. Based on the results of these operations an optimal driving strategy is calculated in the second part of the algorithm for each involved train under the assumption that each train should arrive at its destination within the schedule and safety aspects are guaranteed. The best solution in terms of energy consumption is presented as overall result.

The algorithm uses an implementation of the train model and the track with a comprehensive number of adjustable parameters (e.g. train mass, number of wagons, various resistance values) to be near-term to reality. The better the mapping of the real world to the parameters the better the calculated result is.

As the algorithm is designed to determine only driving strategies where the trains will arrive at their destination in time, there might exist several situations (e.g. due to delays) where no result is available. A useful extension of the algorithm might be to calculate driving strategies for unpunctual trains, too, as well as the usage of train priorities which could be considered in the sequence of entering and leaving common used track sections.

As already mentioned in previous parts of this thesis, the algorithm is designed to run on multi-core CPUs. As a result, the calculation time will decrease with a higher number of available CPUs. Due to simplicity of the mathematical operations, the algorithm could be ported to graphic cards with a very high number of processing units which would result in a dramatic reduction of calculation time.

The development of the algorithm was part of the *EcoRailNet* project in the frame of the program New Energy 2020 (Project-ID: 834586). Several parts of the algorithm were tested in the field with good results and valuable perceptions. Other parts were discussed within meetings and presented on conferences and appear to be a good base for further development.

# Bibliography


[1] Amie Albrecht, Phil Howlett, Peter Pudney, and Xuan Vu. Optimal train control: Analysis of a new local optimization principle. In *American Control Conference (ACC)*, pages 1928–1933, San Francisco, USA, Jun 2011.

[2] Thomas Albrecht. Energy-Efficient Train Operation. In Ingo A. Hansen, Jörn Pachl, and Thomas Albrecht, editors, *Railway, Timetable & Traffic: Analysis, Modelling, Simulation*, pages 83–105. Eurailpress, 1 edition, 2008.

[3] Richard Bellman. *Introduction to Matrix Analysis*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2nd edition, 1997.

[4] Jasna Blašković Zavada, Borna Abramović, and Tomislav Čvek. Energy-efficient drive of trains. In *22th International Symposium on Electronics in Transport ISEP 2014*, Ljubljana, Sovenija, Mar 2014.

[5] Randall L. Brukardt. *Ada 2012 Annotated Reference Manual*, 2012.

[6] Olaf Brünger and Elias Dahlhaus. Running Time Estimation. In Ingo A. Hansen, Jörn Pachl, and Thomas Albrecht, editors, *Railway, Timetable & Traffic: Analysis, Modelling, Simulation*, pages 58–82. Eurailpress, Hamburg,Germany, 1 edition, 2008.

[7] Peter Buchholz and Peter Kemper. Efficient Computation and Representation of Large Reachability Sets for Composed Automata. *Discrete Event Dyn. Systems*, 12(3):265–286, 2002.

[8] Ginautas Bureika, Rimantas Subacius, and Mika Kumara. Research on Energy Efficient of Rolling-Stock Operation. In *The 6th International Scientific Conference "TRANSBALTICA 2009"*, pages 28–32, Apr 2009.

[9] Gabrio Caimi, Fabián A. Chudak, Martin Fuchsberger, Marco Laumanns, and Rico Zenklusen. A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science*, 45(2):212–227, May 2011.

[10] Gabrio Caimi, Martin Fuchsberger, Dan Burkolter, Thomas Herrmann, Raimond Wüst, and Samuel Roos. Conflict-free train scheduling in a compensation zone exploiting the speed profile. In *3rd International Seminar on Railway Operations Modelling and Analysis (ISROR)*, Zürich, Switzerland, Feb 2009.

[11] Gabrio Caimi, Martin Fuchsberger, Marco Laumanns, and Marco Lüthi. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research*, 39(11):2578–2593, 2012.

[12] Gabrio Caimi, Marco Laumanns, Kaspar Schüpbach, Stefan Wörner, and Martin Fuchsberger. The periodic service intention as a conceptual frame for generating timetables with partial periodicity. In *Transportation Planning and Technology 34(4)*, pages 323–339, 2011.

[13] Jiaxing Cheng, Yelena Davydova, Phil Howlett, and Peter Pudney. Optimal driving strategies for a train journey with non-zero track gradient and speed limits. *IMA Journal of Mathematics Applied in Business & Industry*, 10(0199921504):89–115, 1999.

[14] Rémy Chevrier, Grégory Marlière, Bogdan Vulturescu, and Joaquin Rodriguez. Multiobjective Evolutionary Algorithm for Speed Tuning Optimization with Energy Saving in Railway: Application and Case Study. In *RailRome 2011*, Rome, Italy, 2011.

[15] Colin Cole. Longitudinal Train Dynamics. In Simon Iwnicki, editor, *Handbook of Railway Vehicle Dynamics*, pages 239–278. Taylor & Francis Group, Boca Raton, USA, 2006.

[16] Yong Cui. *Simulation-Based Hybrid Model for a Partially-Automatic Dispatching of Railway Operation*. PhD thesis, Universität Stuttgart, 2010.

[17] Marc Davio. Kronecker Products and Shuffle Algebra. *IEEE Trans. Computers*, 30(2):116–125, 1981.

[18] Caroline Desprez and Housni Djellab. Traction energy saving by speed profile optimization. In *Compendium of Papers for the Euro Working Group of Transportation*, Paris, France, 2012.

[19] Edsger W. Dijkstra. *Over Seinpalen*. 1965.

[20] Edsger W. Dijkstra. Een algorithme ter voorkoming van de dodelijke omarming. n.d.

[21] Markus Ellinger. Energiesparende Fahrweise unter Berücksichtigung betrieblicher und technischer Parameter. Master thesis, Fachhochschule St. Pölten, 2013.

[22] Maria Pia Fanti, Alessandro Giua, and Carla. Seatzu. A deadlock prevention method for railway networks using monitors for colored petri nets. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 2, pages 1866–1873, Oct 2003.

[23] Rüdiger Franke, Markus Meyer, and Peter Terwiesch. Optimal Control of the Driving of Trains. *Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs- und Informationstechnik*, 50(12):606–613, Dec 2002.

[24] Rüdiger Franke, Peter Terwiesch, and Markus Meyer. An algorithm for the optimal control of the driving of trains. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2123–2128, Dec 2000.

[25] Martin Fuchsberger. Solving the train scheduling problem in a main station area via a resource constrained space-time integer multi-commodity flow. Master thesis, ETH Zurich, 2007.

[26] ÖBB-Produktion GmbH. Velocity-braking-force-diagram, 2014.

[27] ÖBB-Produktion GmbH. Velocity-traction-force-diagram, 2014.

[28] ÖBB-Produktion GmbH. Verzeichnis zulässiger Geschwindigkeiten, 2014.

[29] Iakov M. Golovitcher. Energy efficient control of rail vehicles. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 10, pages 658–663, Tucson,USA, 2001.

[30] Alexander Graham. *Kronecker Products and Matrix Calculus with Applications*. Ellis Horwood Ltd., New York, USA, 1981.

[31] Peter Henderson and James H. Morris, Jr. A Lazy Evaluator. In *3rd ACM Symposium on Principles of Programming Languages*, POPL '76, pages 95–103, January 1976.

[32] Phil Howlett. Optimal strategies for the control of a train. *Automatica*, 32(4):519–532, Apr 1996.

[33] Phil Howlett. The Optimal Control of a Train. *Annals of Operations Research*, 98(1-4):65–87, 2000.

[34] Phil Howlett and Peter Pudney. *Energy-Efficient Train Control*. Advances in Industrial Control. Springer London, 1995.

[35] Phil Howlett, Peter Pudney, and Xuan Vu. Local energy minimization in optimal train control. *Automatica*, 45(11):2692–2698, Sep 2009.

[36] Daniel Hürlimann. Opentrack railway technology. Website `http://www.opentrack.at`. Visited: 2014-11-25.

[37] Daniel Hürlimann. *OpenTrack Manual*, 2013.

[38] Wilfried Imrich, Sandi Klavzar, and Douglas F. Rall. *Topics in Graph Theory: Graphs and Their Cartesian Product*. A K Peters Ltd, 2008.

[39] Jürgen Jacobs. Rescheduling. In Ingo A. Hansen, Jörn Pachl, and Thomas Albrecht, editors, *Railway, Timetable & Traffic: Analysis, Modelling, Simulation*, pages 182–191. Eurailpress, Hamburg,Germany, 1 edition, 2008.

[40] Birgit Jaekel and Thomas Albrecht. Interfacing conflict resolution and driver advisory systems in railway operation. In *Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 333–343, Dresden, Germany, Dec 2013.

[41] Eugene Khmelnitsky. On an Optimal Control Problem of Train Operation. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 45(7):1257–1266, July 2000.

[42] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Pearson Education, 1998.

[43] Donald E. Knuth. *Combinatorial Algorithms*, volume 4A of *The Art of Computer Programming*. Addison-Wesley, 2011.

[44] Werner Kuich and Arto K. Salomaa. *Semirings, Automata, Languages*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag GmbH, 1986.

[45] Gerhard Küster. On the Hurwitz Product of Formal Power Series and Automata. *Theor. Comput. Sci.*, 83(2):261–273, 1991.

[46] D.H. Lee, Ian P. Milroy, and K. Tyler. Application of Pontryagin's Maximum Principle to the Semi-automatic Control of Rail Vehicles. In *Proceedings of the Second Conference on Control Engineering*, pages 233–236, Newcastle, Australia, 1982. Institution of Engineers Australia.

[47] Liang Li, Wei Dong, Yindong Ji, and Zengke Zhang. A minimal-energy driving strategy for high-speed electric train. *Journal of Control Theory and Applications*, 10(3):280–286, Aug 2012.

[48] Ulrich Linder. *Optimierung von Fahrweisen im spurgeführten Verkehr und deren Umsetzung*. PhD thesis, Technische Universität Berlin, 2004.

[49] Rongfang (Rachel) Liu and Iakov M. Golovitcher. Energy-efficient operation of rail vehicles. *Transportation Research Part A: Policy and Practice*, 37(10):917–932, Jul 2003.

[50] Quan Lu, Maged Dessouky, and Robert C. Leachman. Modeling train movements through complex rail networks. *ACM Trans. Model. Comput. Simul.*, 14(1):48–75, Jan 2004.

[51] Marco Lüthi. Evaluation of energy saving strategies in heavily used rail networks by implementing an integrated real-time rescheduling system. In Eduardo Pilo, editor, *Power Supply, Energy Management, and Catenary Problems*, pages 75–86. WIT Press, 2010.

[52] Ullrich Martin. *Verfahren zur Bewertung von Zug- und Rangierfahrten bei der Disposition*. Schriftenreihe. Inst. für Eisenbahnwesen und Verkehrssicherung, 1995.

[53] Farhad Mehta, Farhad Rößiger, and Markus Montigel. Latent energy savings due to the innovative use of advisory speeds to avoid occupation conflicts. In B. Ning and C.A Brebbia, editors, *Computers in Railways XII: Computer System Design and Operation in Railways and Other Transit Systems (WIT Transactions on the Built Environment)*, volume 114, pages 99–108. WIT Press, 2010.

[54] Jeff Miller. Earliest Known Uses of Some of the Words of Mathematics, Rev. Aug. 1, 2011. Website: `http://jeff560.tripod.com/k.html`, 2012. Visited: 2014-11-25.

[55] Graham Mills and Peter J Pudney. The effects of deadlock avoidance on rail network capacity and performance. In *Proceedings of the 2003 mathematics-in-industry study group*, pages 49–63. Australian Mathematical Society, 2008.

[56] Robert Mittermayr and Johann Blieberger. Shared Memory Concurrent System Verification using Kronecker Algebra. Technical report, Insitute of Computer-Aided Automation, TU Vienna, Austria, Vienna, Austria, Sep 2011.

[57] Robert Mittermayr and Johann Blieberger. Timing Analysis of Concurrent Programs. In Tullio Vardanega, editor, *12th International Workshop on Worst-Case Execution Time Analysis*, volume 23 of *OpenAccess Series in Informatics (OASIcs)*, pages 59–68, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[58] Robert Mittermayr, Johann Blieberger, and Andreas Schöbel. Kronecker Algebra based Deadlock Analysis for Railway Systems. *PROMET-TRAFFIC & TRANSPORTATION*, 24(5):359–369, 2012.

[59] Masafumi Miyatake and Hideyoshi Ko. Optimization of train speed profile for minimum energy consumption. *IEEJ Transactions on Electrical and Electronic Engineering*, 5(3):263–269, Apr 2010.

[60] Markus Montigel. Innovatives Bahnleitsystem optimiert den Zugverkehr im Lötschberg-Basistunnel. *Signal + Draht*, 6, 2008.

[61] Markus Montigel. Operations control system in the Lötschberg Base Tunnel. *European Rail Technology Review*, 49:42–44, 2009.

[62] Joern Pachl and Thomas White. Analytical capacity management with blocking times. In *Transportation Research Board. 83rd Annual Meeting*, Jan 2004.

[63] Jörn Pachl. *Steuerlogik für Zuglenkanlagen zum Einsatz unter stochastischen Betriebsbedingungen*. PhD thesis, TU Braunschweig, 1993.

[64] Jörn Pachl. Timetable Design Principles. In Ingo A. Hansen, Jörn Pachl, and Thomas Albrecht, editors, *Railway, Timetable & Traffic: Analysis, Modelling, Simulation*, pages 9–42. Eurailpress, Hamburg,Germany, 1 edition, 2008.

[65] Jörn Pachl. Deadlock avoidance in railroad operations simulations. In *Textfassung eines Vortrages auf dem 90th Annual Meeting des Transportation Research Board in Washington DC*, pages 23–27, 2011.

[66] E. Petersen and A Taylor. Line block prevention in rail line dispatch. In *INFOR Journal*, volume 21, pages 46–51, 1983.

[67] Brigitte Plateau. On the Stochastic Structure of Parallelism and Synchronization Models for Distributed Algorithms. In *ACM SIGMETRICS*, volume 13, pages 147–154, 1985.

[68] Brigitte Plateau and Karim Atif. Stochastic Automata Network For Modeling Parallel Systems. *IEEE Trans. Software Eng.*, 17(10):1093–1108, 1991.

[69] Alfons Radtke. Infrastructure Modelling. In Ingo A. Hansen, Jörn Pachl, and Thomas Albrecht, editors, *Railway, Timetable & Traffic: Analysis, Modelling, Simulation*, pages 43–57. Eurailpress, Hamburg,Germany, 1 edition, 2008.

[70] Xiaolu Rao, Markus Montigel, and Weidmann Ulrich. Railway capacity optimization by integration of real-time rescheduling and automatic train operation. In *IT13.RAIL*, Zürich, Switzerland, 2013.

[71] Xiaolu Rao, Markus Montigel, and Ulrich Weidmann. Holistic optimization of train traffic by integration of automatic train operation with centralized train management. In C.A. Brebbia, N. Tomii, J.M. Mera, B Ning, and P. Tzieropoulos, editors, *Computers in Railways XIII: Computer System Design and Operation in the Railway and Other Transit Systems (Wit Transactions on the Built Environment)*, volume 127, pages 39–50. WIT Press, 2012.

[72] Andreas Schöbel, Bernhard Rüger, Andrew Nash, Jürgen Zajicek, Martin Turk, and Hartmut Dannenberg. The potential for saving energy by more precisely calculating station dwell times on commuter rail service. In *3rd International Seminar on Railway Operations Modelling and Analysis: RailZurich2009 Conference*, Zürich, Switzerland, 2009.

[73] Thomas Siefer. Simulation. In Ingo A. Hansen, Jörn Pachl, and Thomas Albrecht, editors, *Railway, Timetable & Traffic: Analysis, Modelling, Simulation*, pages 155–169. Eurailpress, Hamburg,Germany, 1 edition, 2008.

[74] N.S. Szabó and R.I. Tanaka. *Residue arithmetic and its applications to computer technology*. McGraw-Hill series in information processing and computers. McGraw-Hill, 1967.

[75] S. Tucker Taft, Pascal Leroy, Robert A. Duff, Randall L. Brukardt, and Erhard Ploedereder, editors. *Ada 2005 Reference Manual*. Springer-Verlag, 2006.

[76] Mark Volcic, Johann Blieberger, and Andreas Schöbel. Kronecker algebra and its broad applications in railway systems. In *EURO-ŽEL 2013: Recent Challenges for European Railways*, pages 275–282, Žilina, Slovak Republic, June 2013.

[77] Mark Volcic, Johann Blieberger, and Andreas Schöbel. Kronecker algebra as a frame for optimisation of railway operation. In *21st International Scientific Conference – TRANSPORT 2013; Mechanics Transport Communications*, volume 11/3, pages 57–63, Sofia, Bulgaria, October 2013.

[78] Mark Volcic, Johann Blieberger, and Andreas Schöbel. Kronecker algebra based modelling of railway operation. In *Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 345–356, Dresden, Germany, December 2013.

[79] Mark Volcic, Johann Blieberger, and Andreas Schöbel. Kronecker-Algebra und ihre breit gefächerten Anwendungen im Eisenbahnbereich. *Signal + Draht*, 7+8:15–18, 2014.

[80] Mark Volcic, Johann Blieberger, and Andreas Schöbel. Optimisation of railway operation by application of kronecker algebra. In *CETRA 2014*, pages 37–42, Split, Croatia, April 2014.

[81] Mark Volcic, Johann Blieberger, and Andreas Schoebel. Kronecker Algebra based Travel Time Analysis for Railway Systems. In *FORMS/FORMAT 2012 – 9th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 273–281, Braunschweig, Germany, Dec 2012.

[82] Xuan Vu. *Analysis of necessary conditions for the optimal control of a train*. PhD thesis, University of South Australia, Nov 2006.

[83] Michal Žarnay. Solving deadlock states in model of railway station operation using coloured petri nets. In *Proceedings of Symposium FORMS/FORMAT - Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 205–213, Budapest, Hungary, 2008.

[84] Yihui Wang, Bing Ning, Fang Cao, B. De Schutter, and T.J.J. van den Boom. A survey on optimal trajectory planning for train operations. In *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on*, pages 589–594, Beijing,China, Jul 2011.

[85] Johann Georg Zehfuss. Ueber eine gewisse Determinante. *Zeitschrift für Mathematik und Physik*, 3:298–301, 1858.