
# Models For The Train Timetabling Problem

Diplomarbeit bei
Prof. Dr. Dr. h.c. mult. Martin Grötschel

vorgelegt von Berkan Erol
am Institut für Mathematik
der Technischen Universität Berlin

Die selbstständige und eigenhändige Anfertigung dieser Arbeit bestätige ich an Eides statt.

Berlin, 5. August 2009

# Danksagung

An dieser Stelle möchte ich mich bei allen Menschen herzlichst bedanken, die mir während der letzten Monate und Jahre beiseite gestanden und mich bei der Bewältigung des Studiums und dieser Diplomarbeit unterstützt haben.

In erster Linie gilt mein Dank *Prof. Dr. Dr. mult. h.c. Martin Grötschel* und *Dr. Ralf Borndörfer*, die mir die Möglichkeit gaben diese Arbeit am Konrad-Zuse-Institut mit seiner hervorragenden Infrastruktur anzufertigen. Außerdem möchte ich mich bei *Thomas Schlechte* bedanken, der es hoffentlich nicht bereut hat die Arbeit eines Betreuers mit all seinen nervigen Konsequenzen auf sich zu nehmen.

Ich bedanke mich auch bei allen Freunden und Kollegen, insbesondere bei meinen Zimmergesellen *Michael Winkler*, *René Dammer*, *Niklas Forck* und *Cemil Erdoğan*, die ich nun zu meinen engsten Freunden zählen darf, die mir stets mit Rat und Tat beistanden und mir auch halfen den Kontakt zu der Welt außerhalb einer Diplomarbeit aufrecht zu erhalten.

Mein persönlicher und tiefster Dank gilt jedoch meiner Familie, insbesondere meinen Eltern, die mich trotz meiner Launen und persönlichen Krisen in allen Lagen unterstützt haben und auf die ich mich stets verlassen konnte. Ich danke ihnen auch dafür, dass sie mir das Studium überhaupt erst ermöglichten.

Mein abschließender Dank gilt meiner Freundin *Kathrin*, die mich durch das Studium mit seinen Höhen und Tiefen begleitet hat und mir dabei stets zur Seite stand.

Vielen Dank!

# Zusammenfassung

In den letzten Jahrzehnten konnte sich der europäische Schienenverkehrsmarkt nur unterproportional am wachsenden Verkehrsmarkt beteiligen. Dazu trugen vor allem der Wunsch nach individueller Mobilität mit dem eigenen Fahrzeug als auch die damit erzielte Schnelligkeit und Flexibilität bei. Die durch die Just-in-Time Prozesse entstandenen Ansprüche der Industrie konnten somit, insbesondere auf kurzen Distanzen, besser auf der Straße als auf der Schiene befriedigt werden. Die Vorteile, die man noch auf langen Distanzen hatte, konnten auf Grund unterschiedlicher Bahnsysteme in Europa und unzulässiger Marktzutritte durch ausländische Konkurrenten nicht ausgespielt werden. Mit der Reformation des europäischen Schienenverkehrs in den 1990er Jahren beabsichtigte die Europäische Kommission dieser Entwicklung entgegenzutreten: Sie forderte eine Trennung des Bahnbetriebes vom Infrastrukturbetrieb. Somit sollte in einem späteren Schritt eine (Teil-) Privatisierung der staatlichen Bahngesellschaften eingeleitet und eine effiziente Nutzung der Infrastruktur gewährleistet werden. Die Aufgabe der so genannten Netzbetreiber ist es, den Eisenbahnbetrieben eine auf unabhängigen, diskriminierungsfreien, transparenten und profitorientierten Kriterien basierenden Zugang zur Netzinfrastruktur zu ermöglichen. Damit kam dem Problem der Zugfahrplanerstellung (engl. Train Timetabling Problem) neue Bedeutung zu und fand besonders in den letzten Jahren immer mehr Aufmerksamkeit.

Die mathematische Modellierung des Train Timetabling Problems (kurz TTP) stellt das zentrale Thema dieser Arbeit dar. Dazu werden sowohl grundlegende bahntechnische als auch formale mathematische Definitionen eingeführt. Um das TTP formulieren zu können, werden zwei wichtige Eingabedaten benötigt. Auf der einen Seite eine Liste von vorliegenden Trassennachfragen der Bahnbetreiber und auf der anderen Seite die makroskopischen Infrastrukturdaten. Hierzu stellen wir eine von uns entwickelte Implementation vor, die aus den detaillierten Daten eines realen Zugkorridors geeignet abstrahierte Infrastrukturdaten erzeugt.

Neben der Betrachtung der wenigen bisher in der einschlägigen Literatur verwendeten Lösungsansätze wird auch ein neues Modell zur Lösung des Problems entwickelt. Da bei dem TTP fast ausschließlich große ganzzahlige Programme auftreten, wird dieses basierend auf der Technik der Spaltenerzeugung implementiert. Die für diese Technik notwendigen Subroutinen sowie eine kurze polyedrische Analyse werden ebenfalls angegeben. Die Arbeit wird mit den erzielten Rechenergebnissen abgeschlossen.

# Contents

# List of Tables

# List of Algorithms

# List of Figures

# Chapter 1

# Introduction

> Equations are just the boring part of mathematics.
> I attempt to see things in terms of geometry.
>
> *(Stephen Hawking)*

In the past decades the traffic volume has risen immensely due to the world-wide grown international network arising by globalization. In the same time the market share of railway decreased until the early 1990s under 20% due to a greater flexibility and customer focus of road haulage yielding also to massive problems for the national budget in some European countries, e.g. the outstanding debts of the state-owned German rail companies (Deutsche Reichsbahn and Deutsche Bundesbahn) increased until 1993 up to 34 billion € (34,000,000,000 €) and a growth up to 195 billions € till 2003 was forcasted. Since no reversal of this trend was expected, the European Union decided to reform the railway systems for ecological as well as economic reasons. The target was to bring as most as possible of the transportation load on rails and to deregulate the market by opening it step by step for private competitors hoping to increase so the flexibility and attractiveness of railway transportation and to disburden the national finances.

One major step of the reformation was the segmentation of the (at that time state-owned) railway companies into so-called Train Operating Companies (TOCs) and Infrastructure Management Companies (IMCs), whereas at least the TOCs should be passed into private hands in a later stage. The TOCs provide preferred timetables, rolling stock and transportation services and can be separated into operators of passenger trains and operators of cargo trains. The IMCs are responsible for the real-time traffic control management and the allocation of trains to tracks depending on the timetables submitted by the TOCs. The allocation created by an IMC has to be done in an independent, transparent, fair (i.e., non-discriminating) and profit oriented way and should be especially comprehensible.

Current statistics show that these reforms (particularly in Germany) could successfully increase the transportation volume on rails. Thus the goods transported by rail in Germany increased from 296,925 thousand of tons (respectively 78,463 millions of tons-km)

in 2003 to 361,116 thousand of tons (respectively 114,615 millions of tons-km) in 2007. This makes a total growth of 21.62 % (46.08 %) in 4 years.[1]

Since extending the infrastructure is involved with huge investments in capital and time, it is very important to use the existing infrastructure as best as possible. In railway traffic this midterm-planning is mainly achieved in the step of creating a so-called "train timetable" by the infrastructure managers, i.e., a feasible schedule of arrival and departure times for a set of trains on a given set of stations. It is important to study these problems and to develop efficient solving techniques in order to handle large instances of the uprising scheduling tasks. The hope is to be able to model and schedule a lion's share of the public railway traffic for a large network as displayed in Figure 1.1.

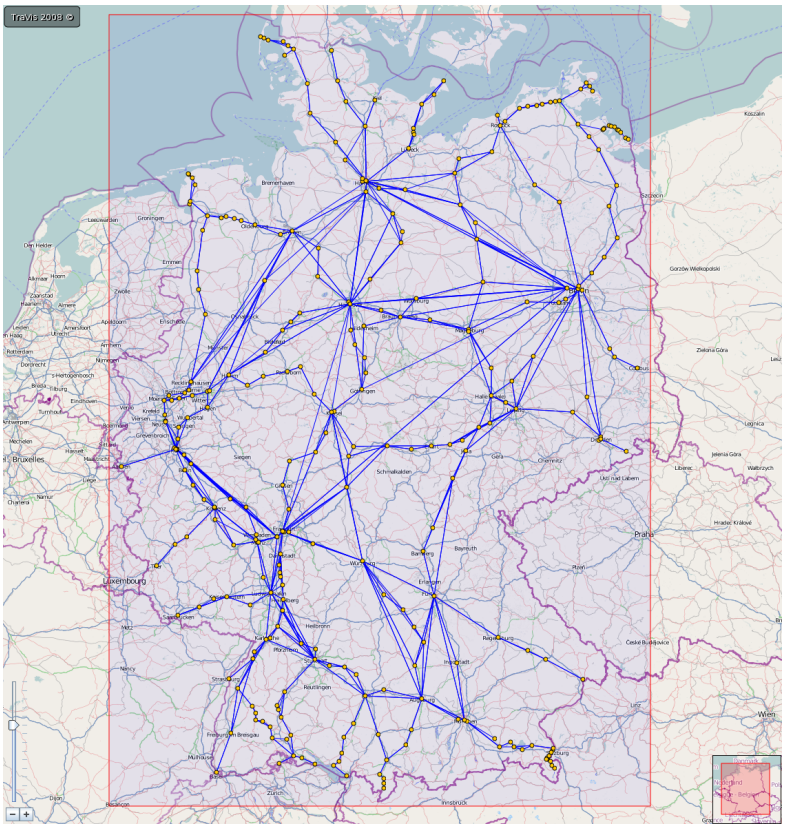



*Figure 1.1:* Macroscopic view to the passenger-train network of the German railway using the visualization tool TraVis.

[1] All statistics are taken from Eurostat (2009) and BMVBS (2009).

**Prerequisite** The mathematical definitions in the following sections contain basic terms of graph theory, and of linear and integer programming. We propose Schrijver (2003), Grötschel (2006, 2007), and Korte and Vygen (2008) for an introduction into this kind of theory.

**Overview** This diploma thesis will introduce a mathematical view to the non-periodic train timetabling problem and is structured as follows. In the next sections of this chapter, we will introduce notational remarks and an introduction to some advanced solving techniques of discrete mathematics needed for this work. In Chapter 2, we describe the technical details that are needed to create a macroscopic infrastructure from microscopic input and introduce a tool to generate such macroscopic networks. After this, in Chapter 3, we introduce formally the Train Timetabling Problem and related definitions. Next, in Chapter 4, we give a literature overview of approaches to solve train timetabling instances and try to classify them. In Chapter 5, we introduce a new model to solve large-scale instances of the (non-periodic) train timetabling problem with the technique of column generation, describe the appropriate pricing problem and give some polyhedral analyzes. Finally in Chapter 6, we provide computational results for our approach and compare them with other existing models.

## 1.1 Notational Remarks

We introduce multiple (mixed) integer programs and use graph theoretical terms and symbols in this work, whose notation is described in this section.

Let $G = (V(G), E(G))$ be an undirected and $D = (V(D), A(D))$ be a directed graph.

For node-sets $X \subseteq V(G)$ and $Y \subseteq V(D)$ we define

- $\delta(X) := \{uv \in E(G) : u \in X, v \in V(G) \setminus X\}$,
- $\delta^+(Y) := \{(uv) \in A(D) : u \in Y, v \in V(D) \setminus Y\}$,
- $\delta^-(Y) := \delta^+(V(D) \setminus Y)$, and
- $\delta(Y) := \delta^+(Y) \cup \delta^-(Y)$.

For singletons, i.e., one-element vertex sets $\{v\}$ and $\{w\}$ with $v \in V(G)$ and $w \in V(D)$, we simply write $\delta(v) := \delta(\{v\})$, $\delta(w) := \delta(\{w\})$, $\delta^-(w) := \delta^-(\{w\})$, and $\delta^+(w) := \delta^+(\{w\})$.

Let $S$ be a set and $x \in \mathbb{K}^S$ be a variable with $\mathbb{K} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$. For a subset $T \subseteq S$ we shortly write

$$x(T) := \sum_{t \in T} x_t.$$

We use the following notation for binary relations. Given to vectors $y, y' \in \mathbb{K}^n$ ($n \in \mathbb{N}$), we write

- $y < y'$, if $y_k < y'_k$ for each $k = 1, \ldots, n$,
- $y \leq y'$, if $y_k \leq y'_k$ for each $k = 1, \ldots, n$,
- $y \not< y'$, if there exists a $k = 1, \ldots, n$ such that $y_k \geq y'_k$, and
- $y \not\leq y'$, if there exists a $k = 1, \ldots, n$ such that $y_k > y'_k$.

In an analogue way, we define $y > y'$, $y \geq y'$, $y \not> y'$ and $y \not\geq y'$.

Let MIP be a (mixed) integer program. We denote the optimal objective value of MIP with $\nu(\text{MIP})$ and the optimal objective value of the linear relaxation $\text{MIP}_{\text{LP}}$ of MIP with $\nu_{\text{LP}}(\text{MIP})$.

The polyhydron associated with a linear program LP is denoted with P(LP) and the one of the linear relaxation of a (mixed) integer program MIP with $\text{P}_{\text{LP}}(\text{MIP})$.

## 1.2 Column Generation

It often happens that linear programs are to large to consider all variables explicitly. The so-called *delayed column generation* is a technique to efficiently solve such large-scale linear programs that have an extraordinary large size of variables but relatively small size of constraints.

Using the simplex algorithm, most of the variables of such a linear program will be non-basic and reach a value of zero in an optimal solution. Hence it is enough to consider only a representative subset of variables explicitly and to solve this subproblem to optimality, hoping to find an optimal solution for the whole problem. Column generation carries this idea out to generate dynamically only such variables which are capable to improve the objective, i.e., variables which have negative reduced costs when minimizing (respectively positive when maximizing), instead of storing all variables statically. Such an approach makes it possible to handle large-scale problems with reducing the computational load. We refer to Desaulniers et al. (2005) for a state-of-the-art survey of column generation.

Assume that we are given the following linear program (MP) (maybe implicitly), which we call *master problem*.

$$\text{(MP)} \qquad \begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

The algorithm works as follows. One initializes the subproblem—the so-called restricted master problem (RMP)—by choosing a set of variables who are assumed to be in an optimal solution or who can be easily generated. In the next step RMP is solved with a standard LP solver. This delivers a dual solution $y^*$ which will be utilized to calculate the reduced cost $\bar{c}_i = c_i - y^* \cdot a_i$ of each variable $x_i$ of MP. This information can now be used

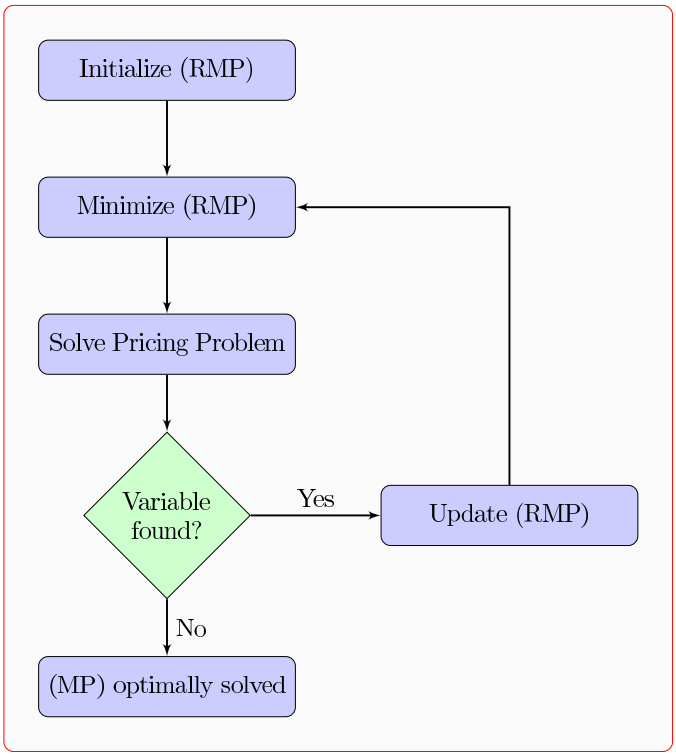



*Figure 1.2:* Column Generation (Flowchart).

to find variables in MP with negative reduced cost which are assumed to improve the objective by becoming basic-variables in the next simplex iteration. The step of creating

new variables can be seen as solving the pricing routine of the simplex algorithm which is why this step is also called *pricing*. The pricing routine of the simplex algorithm is the step that searches a variable with negative reduced cost that should become a basic-variable in the next iteration. The process is continued iteratevely till no variable can be generated, a special solution quality can be guaranteed, or another stop criterion is achieved. This process is also displayed as flowchart diagram in Figure 1.2.

The above described process works fine for linear programs but has in general to be adapted for (mixed) integer programs and delivers in most of the cases only an heuristic solution. These are solved by separating cutting planes and using the branch-and-bound technique. Especially the branch-and-bound technique changes the state-room of the original problem fundamentally and therefore has to be regarded in the pricing routine which leads to a method called branch-and-price.

## 1.3 Dynamic Programming

The concept of dynamic programming was introduced by the American mathematician Richard Bellman in the 1940s and can be shortly described as the sophisticated enumeration of all possible solutions with a divide-and-conquer technique that discards all sub-solutions which will never be contained in an optimal solution. The idea is to find a recursive equation (so-called Bellman equation) which uses information about homogeneous sub-problems to solve the superior problem to optimality. The shortest path problem (SP), the all pair shortest path problem (APSP), the resource constrained shortest path problem (RCSP) and the (0/1) knapsack problem (KP) are the most famous application areas for dynamic programming. For all of these combinatorial optimization problems valid recursions are known whose implementations lead to polynomial—e.g. the Dijkstra algorithm for the (SP) or the Bellman-Moore algorithm for the (APSP)—or at least pseudo-polynomial algorithms—e.g. labeling algorithms for the (RCSP) or the below Algorithm 1 for the (KP).

We try to illustrate the concept on the 0/1 knapsack problem which is defined as follows. We are given a set $N$ of $n$ items, the capacity $c$ of the knapsack, profits $p_i$ and weights $w_i$ for each item $i \in N$. The goal is to fill the knapsack with maximal profit without exceeding the capacity. The problem can also be formulated as an integer program as follows.

$$
\begin{aligned}
\text{(KP)} \qquad & \max \quad p^T x \\
& \text{s.t.} \quad w^T x \leq c \\
& \qquad\quad x \in \{0,1\}^n
\end{aligned}
$$

One can show that this problem is *NP*-hard. Nevertheless the Bellman recursion $z$ for this problem is known to be

$$
z_j(d) := \begin{cases} z_{j-1}(d) & \text{if } d < w_j \ , \\ \max\{z_{j-1}(d),\, p_j + z_{j-1}(d - w_j)\} & \text{else.} \end{cases}
$$

The idea of this equation is to calculate optimal solutions for the subproblems that pack only the first $j-1$ items into either a knapsack of same size as in the current state, or as in the current state reduced about the weight that the current item $j$ takes. The subproblems result of the question whether the $j$-th item should and can be packed or not. This is in a manner of speaking the enumeration of all possible states that the knapsack can take.

The optimal value of a knapsack instance can be solved using this recursion by calculating $z_n(c)$. This can be implemented with a pseudo-polynomial running time of $O(n \cdot c)$ and a space requirement of $O(n + c)$ (see Algorithm 1, taken from: Kellerer et al., 2004, p. 23).

**Algorithm 1**: A Dynamic Program for the Knapsack Problem

**Input**: A set of $n$ items, weights $w_i$ and profits $p_i$ for each item $i = 1, \ldots, n$ , and the capacity $c$ of a knapsack.

**Output**: Optimal solution value $z^*$ of the appropriate knapsack problem.

```
begin
    for d = 0, ..., c do
        z(d) := 0 ;                                   // Initialize z
    for j = 1, ..., n do
        for d = c, c − 1, ..., w_j do
            if z(d − w_j) + p_j > z(d) then
                z(d) := z(d − w_j) + p_j ;            // Item j may be packed
    return z* := z(c);
end
```

**Example 1** Consider the knapsack instance with capacity 10, 7 items and the following profit $p_j$ and weight $w_j$ values (also see Kellerer et al., 2004).

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $p_j$ | 6 | 5 | 8 | 9 | 6 | 7 | 3 |
| $w_j$ | 2 | 3 | 6 | 7 | 5 | 9 | 4 |

The above algorithm would create the following table, whereby only a single row is stored in each iteration $j$.

| | $z_j(d)$ | $d$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| | 2 | 0 | 0 | 6 | 6 | 6 | 11 | 11 | 11 | 11 | 11 | 11 |
| $j$ | 3 | 0 | 0 | 6 | 6 | 6 | 11 | 11 | 11 | 14 | 14 | 14 |
| | 4 | 0 | 0 | 6 | 6 | 6 | 11 | 11 | 11 | 14 | 15 | 15 |
| | 5 | 0 | 0 | 6 | 6 | 6 | 11 | 11 | 12 | 14 | 15 | 17 |
| | 6 | 0 | 0 | 6 | 6 | 6 | 11 | 11 | 12 | 14 | 15 | 17 |
| | 7 | 0 | 0 | 6 | 6 | 6 | 11 | 11 | 12 | 14 | 15 | 17 |

The algorithm is only capable of calculating the solution value but not of recovering the final solution which would pack the items 1, 2 and 5. If we are interested in this, we have to keep a pointer $A_j(d) \in \{0, 1\}$ with the meaning

$$A_j(d) := \begin{cases} 1, & \text{if } z_j(d) = z_{j-1}(d - w_j) + p_j, \\ 0, & \text{if } z_j(d) = z_{j-1}(d) \end{cases}$$

The disadvantage of such an approach is the fact, that the space-requirement is worsen to $O(n \cdot c)$. □

We will later see that our pricing problem is a special case of the Multi-Dimensional 0/1 Knapsack Problem (d-KP) which is defined as

(d-KP)
$$\begin{aligned} \max \quad & p^T x \\ \text{s.t} \quad & W^T x \leq c \\ & x \in \{0, 1\}^n \end{aligned}$$

with a weight matrix $W \in \mathbb{R}^{d \times n}$, a profit vector $p \in \mathbb{R}^n$ and a capacity vector $c \in \mathbb{R}^d$.

The one-dimensional Bellman equation can easily be extended to the $d$-dimensional case (see also Kellerer et al., 2004).

$$z_j(g) := \begin{cases} z_{j-1}(g) & \text{if } g_i \not\geq W_j \ , \\ \max\{z_{j-1}(g),\, p_j + z_{j-1}(g - W_j)\} & \text{else.} \end{cases}$$

with $g \in \mathbb{R}^d$.

However, an analogous implementation for the call of $z_n(c_1, \ldots, c_d)$ using this recursion leads to an exorbitant running time and space requirement of $O(n \cdot c_{\max}^d)$ with $c_{\max} := \max\{c_1, \ldots, c_d\}$.

## 1.4 Multicriteria Optimization

It often happens that an optimization problem should regard more than a single objective function, e.g., modern route guidance systems allow the user to choose a route that is either the shortest, the fastest,the most beautiful, or something between all of these.

*Multicriteria optimization* is a branch of mathematical optimization analyzing techniques that allow to find solutions that are in a special way optimal for a composition of conflicting objective functions. In this section, we will introduce some general definitions and notations used in multicriterial optimization. For a detailed survey of this topic, we refer to Ehrgott and Ruzika (2005).

In the following, let $f : \mathbb{R}^n \to \mathbb{R}^p$ be a continuous functions. A multiobjective program (MOP) is given by

$$\text{(MOP)} \qquad \begin{aligned} \min \quad & f(x) = (f_1(x), \ldots, f_p(x)) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

where $X$ is assumed to be given implicitly in the form

$$\begin{aligned} X := \{x \in \mathbb{R} : \quad & g_j(x) \leq 0 \quad \forall j = 1, \ldots, l; \\ & h_j(x) = 0 \quad \forall j = 1, \ldots, m\}. \end{aligned}$$

**Definition 1** Consider the (MOP). We call $x \in X$

1. a *weakly efficient solution* if there is no $x' \in X$ such that $f(x') < f(x)$,
2. an *efficient solution* if there is no $x' \in X$ such that $f(x') \leq f(x)$.

We call the image of a weakly efficient solution *weakly non-dominated* (*weakly Pareto optimal*) and the image of an efficient solution *non-dominated* (*Pareto optimal*). □

The classic approach for solving MOPs is the so-called *scalarization technique* which transforms the multi-objective function of a related MOP into an single-objective function of a traditional single objective program (SOP).

**Weighted Sum Method** A common scalarization method for solving convex MOPs—and especially multi-objective linear/integer program (MOLP/MOIP), i.e., MOPs of the form

$$\begin{aligned} \min \quad & Cx \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & (x \in \mathbb{Z}^n) \end{aligned}$$

with $C \in \mathbb{R}^{p \times n}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$—is the so-called *adaptive weighted sum method*. This method solves multiple times a parametrized single objective problem of the following form

$$(\text{MOP}_\lambda) \qquad \begin{aligned} \min \quad & \sum_{i=1,\ldots,p} \lambda_i \, f_i(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

for different parameters $\lambda \in \mathbb{R}^p_+$ and $\sum_{i=1,\ldots,p} \lambda_i = 1$.

For the bi-criteria case, i.e., $p = 2$, one can show that the below recursive Procedure 2, the so-called *weighted sum method*, is capable to find all efficient solutions of a MOLP and all extreme supported nondominated solutions of a MOIP when starting with the lexicographic minimal solutions. Before introducing the appropriate procedure, we need the following definition.

**Definition 1 (Extreme Supported Nondominated Points)** Let $x^*$ be an efficient solution of a given multiobjective integer program (MOIP). If there exists some $\lambda \in \mathbb{R}^p$, $\lambda > 0$ such that $x^*$ is an optimal solution of $\text{MOIP}_\lambda$, then $x^*$ is called a *supported effcient solution* and $y^* = c^T x^*$ is called *supported nondominated point*. Moreover, if $y^*$ is an extreme point of $\text{conv}(Y)$, then $x^*$ is called an *extreme supported effcient solution* and $y^*$ an *extreme supported nondominated point*. □

**Procedure** `AdaptiveWeightedSumMethod`($x^1$, $x^2$)

**begin**
  **if** $c^T x^1$ *and* $c^T x^2$ *extreme supported nondominated points* **then**
    **if** $c_1^T x^1 < c_1^T x^2$ *and* $c_2^T x^1 > c_2^T x^2$ **then**
      $\lambda_1 \leftarrow c_2^T x^1 - c_2^T x^2$;
      $\lambda_2 \leftarrow c_1^T x^2 - c_1^T x^1$;
      $x^* \leftarrow$ solution of $\text{MOP}_{(\lambda_1,\lambda_2)}$;
      **if** $c^T x^*$ *new non-dominated point* **then**
        Call `AdaptiveWeightedSumMethod`($x^1$, $x^*$);
        Call `AdaptiveWeightedSumMethod`($x^*$, $x^2$);
**end**

**$\varepsilon$-Constraint Method** Another method based on the scalarization technique is the so-called *$\varepsilon$-constraint method* which can also be used to obtain the efficient solutions of a MOP. The idea of this method is to optimize the regarded problem with respecting a single objective and bounding the other objectives by introducing a new constraint. We obtain the following SOPs

$$(\varepsilon\text{-MOP}_j) \qquad \begin{aligned} \min \quad & f_j(x) \\ \text{s.t.} \quad & f_k(x) \leq \varepsilon_k \quad k \neq j \\ & x \in X \end{aligned}$$

with adequate $\epsilon \in \mathbb{R}^p$.

In contrast to the weighted sum method, the $\varepsilon$-constraint method may change the complexity of the single objective problem, since the additional $\varepsilon$-constraints are knapsack-type constraints, i.e., $\varepsilon$-$\text{MOP}_j$ is in general *NP*-hard.

# Chapter 2

# Macroscopic Railway Infrastructure Modeling

> Die Mathematiker sind eine Art Franzosen:
> Redet man zu ihnen, so übersetzen sie es in ihre Sprache, und dann ist es alsobald ganz etwas anderes.
>
> *(Johann Wolfgang von Goethe)*

While trying to avoid as far as possible technical details about railway and railway design in this work, we need, however, to introduce some basic railway terms, doing so in this chapter. A technical survey of railway timetabling and traffic management can be found in Hansen and Pachl (2008) which also contains more detailed descriptions and explanations of terms used in this works. The authors also compare different modeling depth (microscopic, mesoscopic and macroscopic) of a railway network showing the application area of each.

## 2.1 Motivation

The first and maybe most important step in mathematical modeling is to determine which of the given enormous real-world data should be regarded while creating an appropriate abstract model. The main goal is to retain as most microscopic real-world data as possible into a macroscopic abstract model.

In our case of a railway system, those microscopic data could for example be: incline, acceleration, driving power, power transmission or signal positions. Since most of the data are given in a—from optimization point of view—unhandy way. We have to transform them in more accurate units, e.g., a macroscopic model for timetabling don't needs to know which speed a train can have at the 34-th milestone on a route from station A to B but of course how long the total trip would take. But the question which of these information are important and which can be neglected is not quite easy to answer. Even the question which set of stations should be regarded is non-trivial. Modeling only

stations that actually exist in real-world may be insufficient. It is possible that the line between two stations is in a small area only passable on a single track. Modeling the whole line as single-tracked may reduce the solution-quality immensely since a scheduled train has to wait in this model much longer for a train passing the opposite direction as really necessary. To handle this problem one can introduce so-called *pseudo stations* to divide the whole track into several sections. But inserting to many of these pseudo stations can lead to the situation that we are no longer able to find solutions due to a grown problem size. It is also imaginable that such modeling problems only occur for some type of trains, e.g., most of the deployed passenger trains can pass the tube of our test instance SiT (see Section 2.4 for more details) on two existing tracks but most of the deployed freight trains can't. One side of this tube is undersized for the amply dimensioned freight trains. This leads us to the question how to embed such issues in a solution methodology which we discuss later in Section 2.4.

## 2.2 Infrastructure Topology

A railway network consists mainly of a set of stations, junctions and other important positions of the system that are all connected by railway tracks. The macroscopic model abstracts these aspects in the form of a network which encodes most of the important data and which we call infrastructure graph and introduce formally in Chapter 3.

**Stations** In a microscopic view, a station consists of several internal tracks with platforms that allow passenger to enter/leave a passenger train or to (un-)load a freight train, and tracks beside a station allowing a train to pass this station without holding (which we will, for simplicity, also denote as internal tracks in the following). When regarding a macroscopic design, one has to summarize these station topology into a single vertex of the abstract infrastructure graph. The station characteristics can then integrated as node properties into the model and regarded within a solution procedure. As we will describe later in Section 2.4, it is not enough to abstract only real stations but also has to regard railway junctions or similar positions in the macroscopic graph.

**Tracks and Signals** The tracks are maybe the most important components of a railway network. They are the routeways of a railway system and imply the possible routes in the network of a train. The tracks are through different kind of (fixed) signals implicitly divided into several line sections that we call blocks. One may distinguish between several signal types. The most common types are the *main signals*, the *distant signals* and the *shunt signals*. In a fixed block system, a main signal indicates whether a train may enter the section of the track behind the appropriate signal, i.e., the next block on

the route. A train that passes a main signal, blocks the appropriate block section and, depending on which block system is used, the track behind a train is cleared—i.e., no longer blocked for other trains—either sectionally (fixed block systems) or continuously (moving block systems) (Pachl, 2008). Another train must not enter a block section until its released by the train ahead. On a line segment with long distances between to main signals special distant signals are placed in the braking distance of the next main signal in order to provide an approach (but not a stop) indication. The aspect of shunt signals are integrated into main signals in order to authorize shunting movements and to protect running movements against them. For a detailed description of the different signals and signal systems we refer to Pachl (2008). A macroscopic track does not contain all the microscopic details, especially of the signals anymore. The necessary safety regulation has to be assured through macroscopic headway constraints, which we introduce in the next section. One can picture the blocking and clearing of a train in a space-time diagram as (step) functions. See Figure 2.1 for an example of such a space-time diagram for a fixed block system which will be regarded as default case in the following.

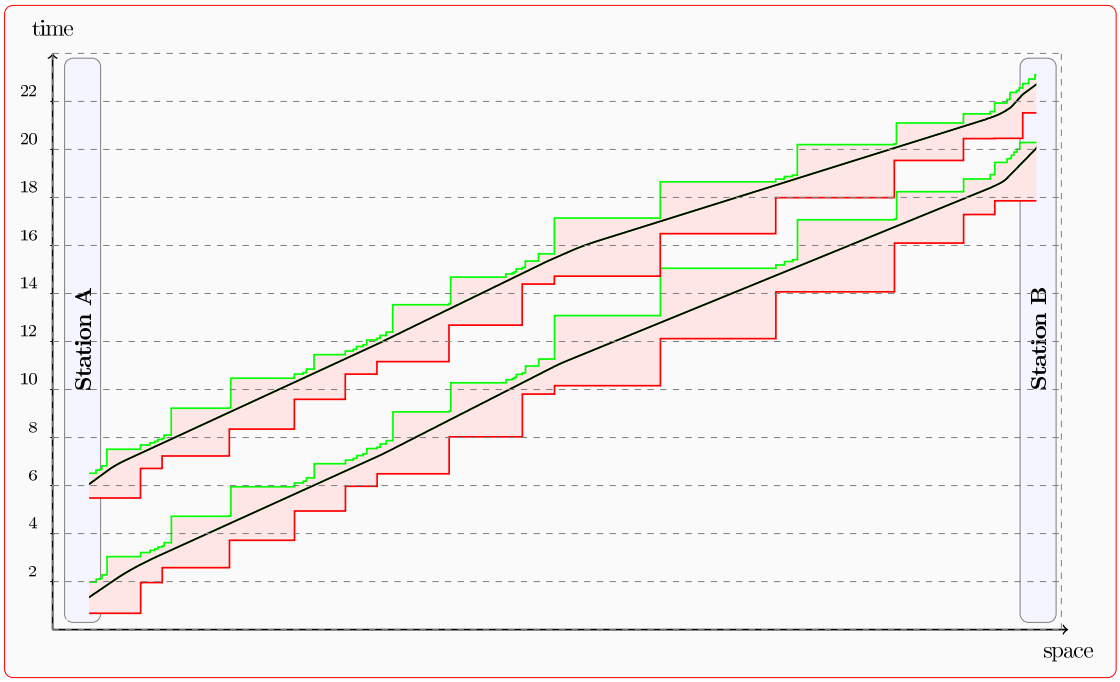



*Figure 2.1:* Route reservation (red) and route release (green) of an Euro-City (above) and a freight train (below) moving from the first to the last station of our SiT test-instance.

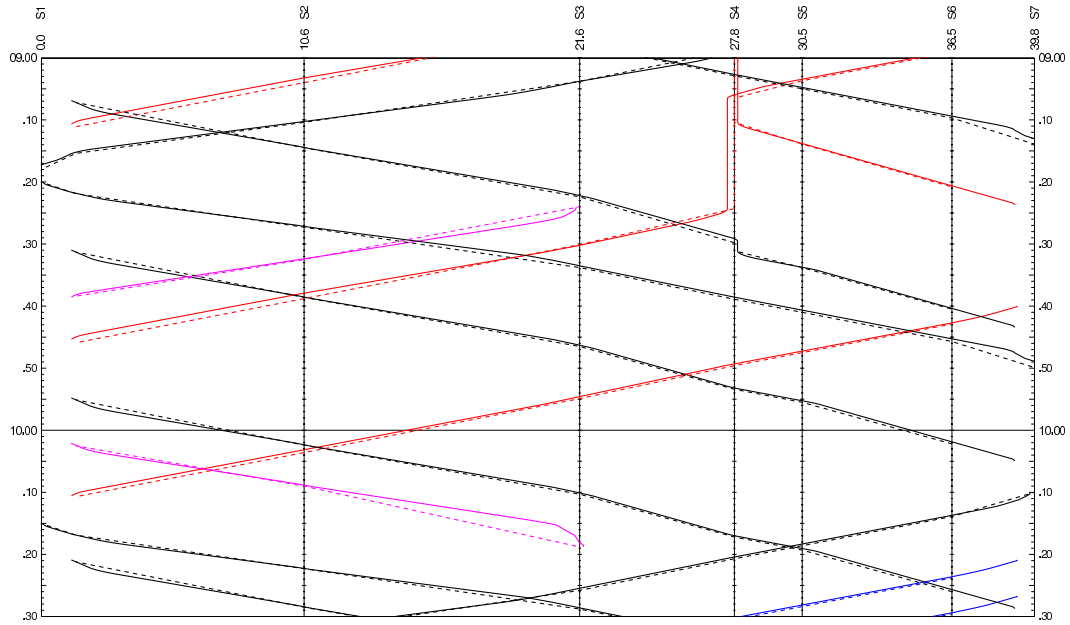



*Figure 2.2:* Traffic diagram in `OpenTrack`.

**Traintypes** The infrastructure also consist of rolling stock which has also to be designed in an abstract model. A main problem is that the calculations of the driving dynamics of a special train are very complex and depend on multiple factors, especially the route that is taken by the regarded vehicle. So, the macroscopic traintypes that we regard are generated by the composition of the possible routes and the values that an appropriate microscopic train takes for this route.

### 2.2.1 Diagramming Railway Traffic

Many train operators use train diagrams in order to support the planning tasks as well as to control the operating processes. The standard approach is to represent the traffic on each train line as time-distance diagram with labeling the distance-axis with the names of the stations and their distance on the regarded line. Figure 2.2 shows such a diagram for a simulation run of our test-instance "SiT" taken from the train simulator `OpenTrack`. The figure contains the planned traffic (continuous line) of different traintypes (each displayed with an own color) and a simulated run of this schedule (dashed line) on a line from station $S_1$ to $S_7$.

These kind of diagrams are suited especially for cyclic timetables, e.g., for passenger traffic, that is in most of the cases based on line plans. In this work, we have a main

focus on the non-cyclic timetabling problem that is particularly qualified for the planning of freight traffic which is inquired by non-recurring requests. In order to support the train scheduling process, we implemented the 3d-visualization and analyzing tool TRAVIS[1] that is capable of displaying requested timetables (see Figure 2.4), a scheduled timetables (see Figure 2.3), and occupation blocks of selected tracks (see Figure 2.5).

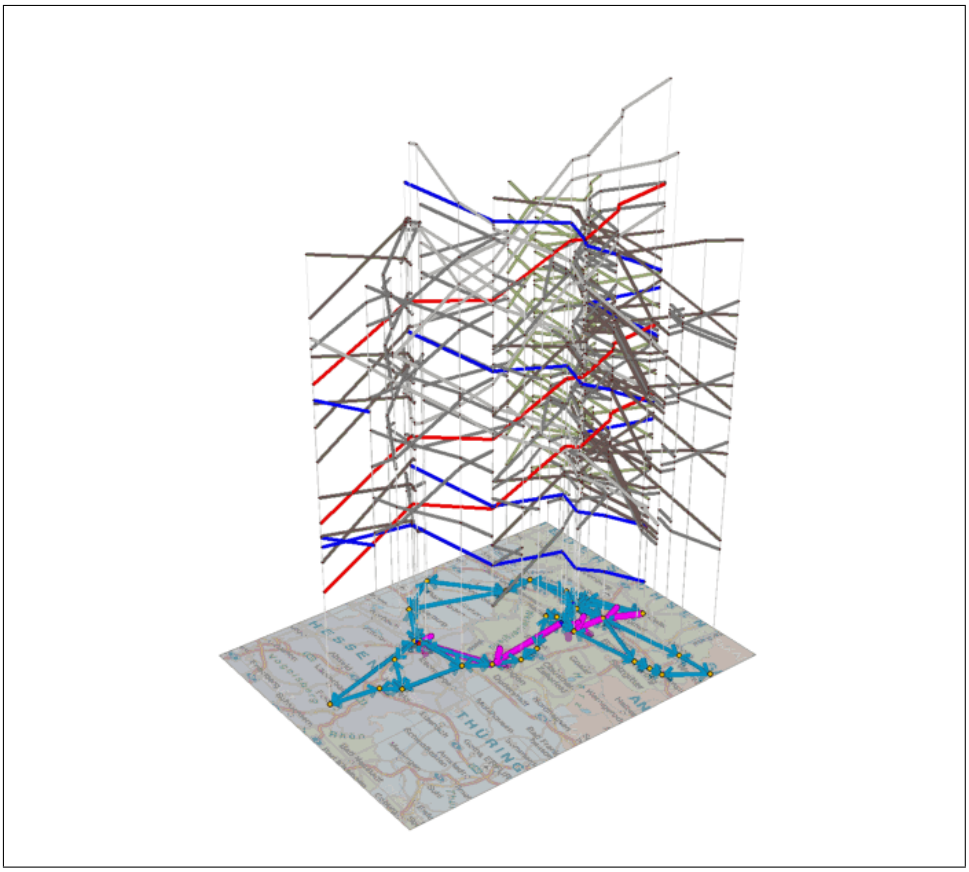


*Figure 2.3:* A planned schedule displayed with TRAVIS.

Another type of diagrams, that is not displayed here, are the station traffic diagrams that show the traffic inside a station separated by traffic associated with each incident track. We refer to Pachl (2008) for a more detailed description of traditional methods for diagramming train traffic.

[1] A public version of TRAVIS can be downloaded from http://ttplib.zib.de/.

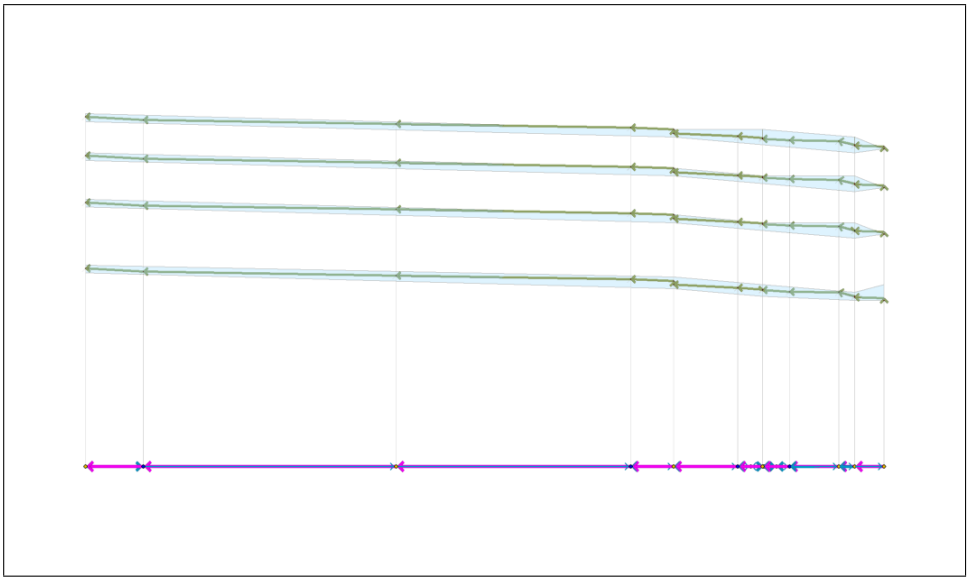


*Figure 2.4:* A requested schedule.

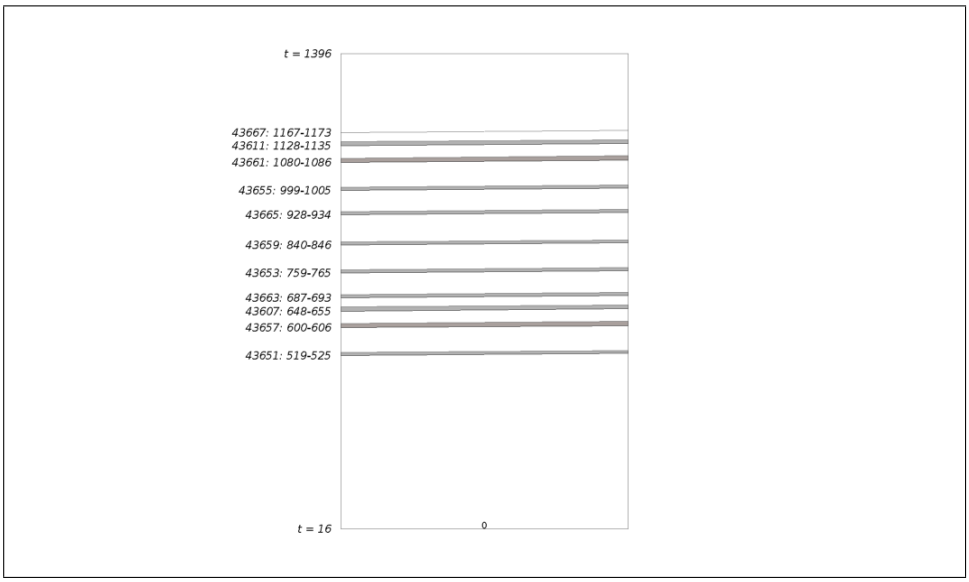



*Figure 2.5:* Block occupation of a track.

## 2.3 Operational Constraints

Suppose we handled all the above mentioned problems (we present in Section 2.4 a way to handle these problems in our case) and have selected sets of stations, rail tracks—each track connecting two stations—and traintypes it left the problem to formulate all important operational constraints and integrate them into our model(s).

In our case, the most important data for an appropriate modeling are

(a) maximal station capacities,

(b) (minimal) driving times,

(c) (minimal) turn-around times, and

(d) minimal headway times.

The *station capacity* (a) of a station describes how many trains of a given type can simultaneously wait in or pass through it. The value is calculated by comparing the number and length of the (reachable) internal tracks inside a station with the default length of the regarded train-type and should deliver an upper bound for the next planning step of the so-called Train Platforming Problem. Unfortunately it is not clear if a so computed capacity value is really valid and has to be checked by running multiple simulations.

The *driving time* (b) is the minimal journey-time that a train needs to pass a track. This value depends on multiple train-type specific factors like acceleration, maximal speed, or gross load weight, and on track specific factors like incline, track quality, or speed-limit. An important aspect for the driving time calculation are the driving dynamics of the regarded vehicle. Especially the question if a train departs and arrives in the start and end points of a track either on-the-fly or with stoppage, has a great effect on the journey time. Due to the complexity of the calculation—one has to solve differential equations that respects as most as possible of the train and track profiles (see Pachl, 2008, for more details). We calculate these values by simulating with (commercial) simulation tools a run of each train starting and ending in two arbitrary stations with affecting the run by different circumstances, e.g. stopping at all, some or none of the stations. The simulations should ideally not consider any detrimental effects derived from other trains and be accomplished in a deterministic, traceable process. The value should only considers the pure running time of a train without any external influences. In contrast to this definition, one can also consider a running time that is increased by additional time that should help to recover lost time for delayed trains.

A *turn-around time* (c) is the time that a train at least needs to spend in a station with the intention to leave a station in the direction it entered. This value is especially important for dead-end stations (like Leipzig central station) in which each train, not

ending in that station, has to turn. It is estimated by the rule-of-thumb and needs subject-specific experience of a planner.

The *minimal headway times* (d) are the most important safety regulation constraint of a timetable. Pachl (2008) describes, that headways can be considered in two different ways: Either assigned to the stations that limit a section, or assigned to the section between two stations. Assigning headways to stations is the more traditional principle (according to Pachl, 2008) and therefore seen as default case in this work. It is defined as the temporary distance between arrival/departure events in a station of two trains running on the physically same track either in the same or on the opposite direction which must at least be holden to avoid rear-end respectively front-end collisions on that track by respecting the specifications of the used block system. So, one can distinguish between four types of headways that can assigned to stations: "depart-depart", "arrive-arrive", "depart-arrive" and "arrive-depart" headways. The headways that can be assigned to the section between two stations, is defined as the temporary difference between two successive trains of either direction enter the section. According to Pachl (2008), this method has become popular in the last years for analyzing the track capacities. The headway times are in general completed by supplemental time that is denoted as buffer time, and should prevent the transmission of small delays from one train to the other. Our definition of headway times does not consider any additional times and consists only of the technically needed, pure headway times.

A final train schedule has to guarantee that no *block-conflict*—i.e., a train trespasses a blocked track section—can arise. First train scheduling models (see Chapter 4) are based on a block-conflict-free formulation which can lead to an huge problem size especially when regarding several tracks. Current models tend to use a headway time based formulation which use so-called headway matrices for each track containing the minimal headway times for each combination of train-types on that track. The following Algorithm 3 shows how departure-departure headways can be obtained if the necessary output data are delivered, for example through a simulation. The graphical interpretation of this algorithm is the idea of shifting the clearing stairway of the train ahead and the blocking stairway of the train below together to meter the necessary temporary difference between the two departures at the start station. We can also see that the above described, four headway types can easily be transformed each in another when we choose the same center of reference for each of them.

An short overview, comparing microscopic and macroscopic models can be seen in Table 2.1.

**Algorithm 3**: Calculation of Minimal Headway Times

**Input**:

- Track $j$, starting in $s_1$ and ending in $s_2$
- Two trains $c_1$ and $c_2$
- Clearing function $c$ of train $c_1$ on track $j$
- Blocking function $b$ of train $c_2$ on track $j$

**Output**: Minimal headway time on track $j$ for the case that train $c_2$ follows train $c_1$

**begin**

$t_1 \leftarrow$ departure time of $c_1$ in $s_1$;
$t_2 \leftarrow$ departure time of $c_2$ in $s_1$;
$\Delta \leftarrow \infty$;
**for** $x = s_1, \ldots, s_2$ **do**
  $\Delta = \min\{b(x) - c(x), \Delta\}$;
**return** $t_2 - t_1 - \Delta$;

**end**

| Microscopic | Macroscopic |
|---|---|
| Station topologies | Station capacities |
| Driving dynamics | Discrete driving times |
| Individual engines | Abstract traintypes |
| Block reservation/release | Minimal headway times |

*Table 2.1:* Comparision: microscopic versus macroscopic

## 2.4 A Macroscopic Network Generator

The problem is to obtain the above discussed macroscopic data from the given microscopic ones. We developed for this purpose `NETCAST`, a prototype of a converter which translates the output data stream of the rail simulator `OpenTrack`[2] into a more appropriate macroscopic model. The main idea is to create firstly a virtual world (using `OpenTrack`) with considering as much real-world data as possible, secondly to run simulations for each train in this world and lastly to analyze and translate the data won by these simulations into a macroscopic data structure.

The so created data can be handled by `TS-OPT` which provided the framework for the implementations of this work. `TS-OPT` is a railway optimization suite developed at Zu-

[2]Producer homepage: `http://www.opentrack.ch/`

se-Institute-Berlin (ZIB) to solve train timetabling problems to optimality using state-of-the-art integer programming models and techniques.

We chose as simulator the tool `OpenTrack` which was developed at the Swiss Federal Institute of Technology and—according to the producer statement—*is used by railways, the railway supply industry, consultancies and universities in different countries.* The data and simulation results are provided by our partners from the working group of *Railway and Track Systems* of the Technical University Berlin (TUB) (German: *Schienenfahrwege und Bahnbetrieb*, shortly SFWBB)

The input data are given by a real-world instance representing a corridor with about 10 train stations. We had to adept the instance and insert some artificial (pseudo) stations such that the final macroscopic network contains 18 stations and 40 tracks. `NETCAST` calculates for six distinguished traintypes about 1000 headway constraints which would have to be calculated by hand without this instrument.

As output format we have chosen the network file specification of the `TTPlib`—a problem library providing at this time over one hundred instances—that was developed within a cooperation of the ZIB, the SFWBB and the Workgroup for Infrastructure Policy (WIP) of the TUB arising in the *Slot Allocation For Railways* project (*German:* Trassenbörse) that is funded by the German Federal Ministry of Economics and Technology (BMWi) and has the goal to analyze if an auction based allocation of railway capacity can lead to a more efficient usage of the railway infrastructure.

The library provides three (macroscopic) file-formats in the flexible markup language XML: one for the infrastructure, one for the requested trains, and one for a final train schedule. The infrastructure file-format of the library consists of three parts: traintype specifications, a set of stations, and a set of tracks, who are explained in the following. Requests are given as a list of slot requests, each containing a stop list of stations that must be visited within given time-windows, penalty values for variations of a given optimal time value of a station-visit, and a basic profit value for the schedule of this request. A final schedule is given as a list of paths, each containing lists of the used tracks and the visited stations together with the scheduled arrival and departure times.

In order to aggregate common properties and restrictions, the traintypes are given as a hierarchically build tree whose nodes represent sets of traintypes, more precisely each non-leaf node is the union of all of his children, and the leaf nodes are singletons representing the actual traintypes of the regarded network. Given such a data structure it is possible to formulate conditions for each trainset that have to be fulfilled for each traintype in that trainset. Each station contains information about his capacity, i.e., how many trains can be simultaneously in that station, and the turnaround times according to this station, both separated by the meant traintype respectively trainset. The tracks consist of driving time specifications on the one side, and the according headway matrices on the other.

The occurring time specifications are all discretized which led to a main problem while scaling the given "in-second" time-values into less accurate, e.g., "in-minute", time units. One can say that the data, especially the driving times, got distorted through this transformation.

**Simple rounding** To clarify this aspect, assume that we are given a minimal example with three waypoints $A, B$ and $C$ for a train that needs 75 seconds for the trip from $A$ to $B$ (short $A \to B$), and 105 seconds for the trip from $B$ to $C$ (short $B \to C$), and that we are targeting a time discretization in one-minute steps. So, a journey starting in $A$ and ending in $C$ would take exactly 3 minutes. To generate macroscopic data that is valid for each possibly requested train, one has to round up each driving time when converting them into other time units. This means for our example that the above regarded train would need 2 minutes for each trip $A \to B$ and $B \to C$, and so 4 minutes for the total journey in a macroscopic world. So, a final timetable would always schedule the above train with a suboptimal total journey time which leads to suboptimal solutions, e.g., regarding a timetable with 60 scheduled trains we may loose an hour. Such a difference may be accepted as scheduled buffer but is not the basic intention of our profit oriented scheduling process. Consider Figure 2.6 for a plot of the driving time on the track $A \to B$ for different time discretization achieved through simply rounding up. One can see that the chosen discretization has a great influence to the accuracy of the final discrete values. The figure also shows, that the obtained discretized times are in a manner globally but not locally monoton, since we achieve amplitudes that increase the higher the time discretization is chosen.

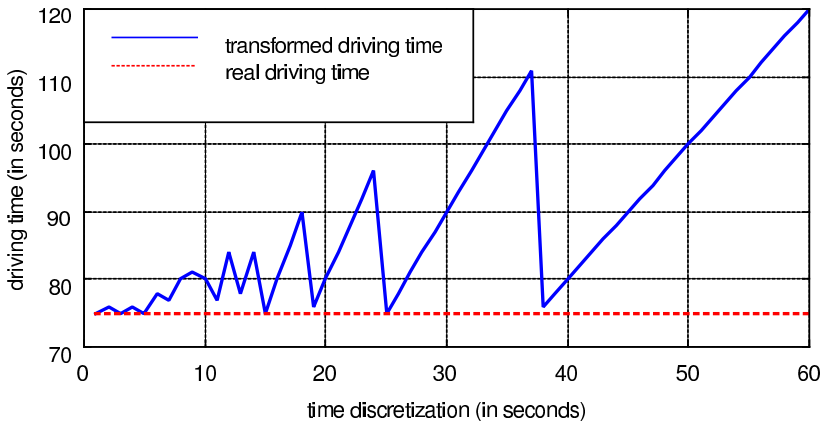



*Figure 2.6:* Driving time on track $A \to B$ for time discretizations between 1 and 60 seconds.

**Smart rounding** In our implementations we use a special rounding technique to prevent the result from such problems. Iterating over the given $m$ track sections we assign the macroscopic driving times of a train as follows.

$$d_i = \begin{cases} \left\lceil \frac{\delta_i}{T} \right\rceil, & \text{if } \xi_i + \left\lceil \frac{\delta_i}{T} \right\rceil - \frac{\delta_i}{T} < T \text{ or } \frac{\delta_i}{T} < 1, \\ \left\lfloor \frac{\delta_i}{T} \right\rfloor, & \text{else} \end{cases}$$

with $\delta_i$ being the given microscopic driving time in-seconds, $T$ being the targeted time discretization given in-seconds, $d_i$ being the wanted macroscopic driving time, and $\xi_i$ being the accumulated "rounding-mistake" in the $i$-th iteration ($\xi_0 := 0$), i.e., $\xi_i := \sum_{j=1}^{i} \delta_i - d_i$ for each track $i = 1, \ldots, m$.

Using this values, it is guaranteed that the total driving time of the whole macroscopic journey differs at most one minute from the given microscopic ones. Since simulations of our solutions led to acceptable schedules, i.e., schedules with less need of rescheduling, and have also not lost much room to benefit from the optimization process, we prefer this values instead of the overcautious variant above.

**Retransformation** Another important feature of NETCAST, which was implemented in cooperation with the SFWBB, is the reinterpretation of solutions obtained by Ts-Opt. The main idea is to remember the original values obtained by the simulation runs and reinsert these data into a microscopic train schedule. This timetable can now be tested by the chosen simulator (OpenTrack) and analyzed with and without regarding external disturbances.

**Pseudo stations** Consider the picture detail displayed in Figure 2.7. We see the station and track topology of a microscopic network, to be more precisely the topology of our SiT test-instance between the stations $S_4$ and $S_5$. The station $S_4$ consists of three internal tracks (dark blue cuboids) with two platforms in total and $S_5$ consists of two internal tracks. The tracks are divided implicitly into several (direction-dependent) block sections by main signals (red circles; yellow cuboids mark distant signals) and some track crossings. As we see, there exist three possible routes for a train from the below platform at station $S_4$ to the below internal track in station $S_5$. Each route has a different driving time which also effects the calculation of the headway times and should therefore be designed—in an accurate model—each as a separate macroscopic traintype.

We regarded six (microscopic) traintypes and up to 50 routes in total in our test-set. Most of the trains are capable to pass each track of the network, but not all. Some freight trains are not able to pass the section after the last junction on a route from station $S_4$ to station $S_5$ that uses the above track. So, we had to insert a pseudo station

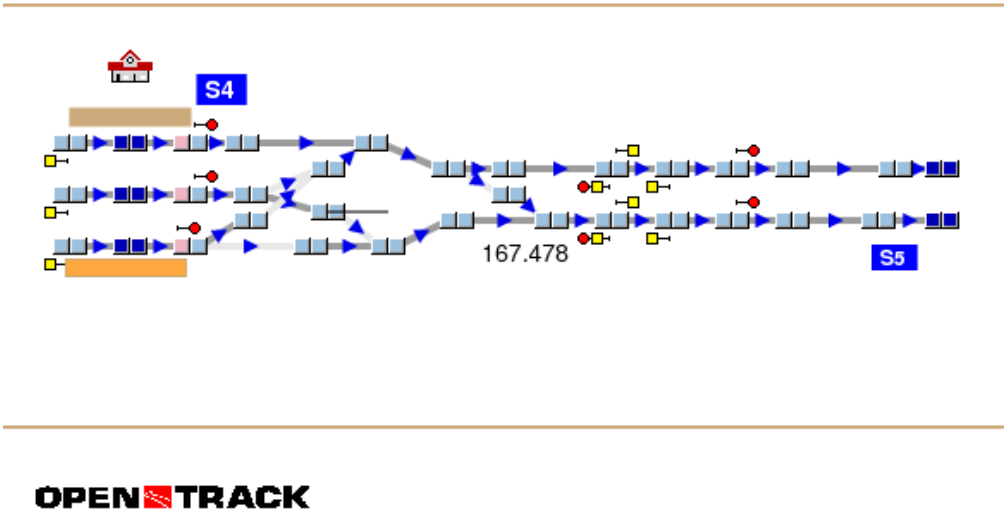



*Figure 2.7:* Screenshot of the railway topology of a microscopic network in the train simulator `OpenTrack`.

at this junction and modeled this part as single-way track for the appropriate trains. A main problem that occurred, results from the fact that no main signals in the close neighborhood of this junction is placed. Tests showed that the arising occupation blocks are too large and, therefore, the calculated headway times have been higher then needed for this section. We solved these problems manually, by choosing the position of the pseudo knots depending on the regarded train and his route, and checking the solutions in multiple simulation runs. We believe, that this procedure is an important step for a full-automated generation of macroscopic networks in future.

# Chapter 3

# The Train Timetabling Problem

In this chapter, we introduce formally the Train Timetabling Problem. The chapter is structured as follows. In the next Section 3.1, we motivate the TTP, give an informal definition of the TTP, and describe the main differences between periodic and non-periodic timetables. The following Section 3.2 introduces the main definitions related with the TTP, define the TTP formally and close the chapter with an example.

## 3.1 Introduction

Railway management provides an huge area of strategic, tactical, operational and in-time planning tasks who are decomposed due to the complexity and diversity of the whole scheduling process. One can distinguish at least the following sub-divisions in infrastructure railway management who are tried to be integrated more and more aiming at better results. The first planning step is the so-called *network planning* which has to be classified as long-term task and whose result provides the infrastructure network for the following planning tasks. This step is followed by the mid-term tasks of the so-called *line planning* which provides the necessary informations for the calculation of periodic train schedules in the so-called *cyclic train timetabling* that take effect in passenger traffic in most of the European countries. After scheduling the regular passenger traffic, one has to manage the freight transportation and long-distance passenger trains in the so-called *non-periodic train timetabling* stage. The so created abstract schedules have to be concertized in the following *vehicle scheduling* steps: *trainplatforming*, *rolling stock circulation* and *train unit shunting*. Finally, a crew has to be assigned to each planned train in the *crew planning* iteration. It is clear, that the steps are not independent of each other and deliver informations for the previous stage. See also Cacchiani (2009) for a short description of each task.

The process of handling a given set of desired train operating schedules and merging these requests as best as possible to a valid timetable is called *train timetabling problem* (TTP).

The result of this step is a train timetable holding technical or safety limitations and operator preferences. One can distinguish between cyclic and non-cyclic timetables.

**Cyclic timetabling** The main advantage of cyclic timetables are the fact that these schedules are easy to operate and remember for passengers, why this method takes effect on scheduling passenger traffic. On the other hand, these timetables are expensive to operate since no distinction between off-peak and peak hours are made. The only way to handle those differences for a train operator is to vary the length of the performed trains. This modifies the variable operating costs—variable rolling stock as well as crew costs are involved—since e.g. shorter trains require less personal to be conducted.

In literature the cyclic TTP is very closely connected to the so-called Periodic Event Scheduling Problem (PESP) and almost always implemented as a PESP. The PESP can be interpreted as a generalization of the cyclic TTP and was first introduced by Serafini and Ukovich (1989). The idea is to schedule the events for one period, e.g. one hour, that is then repeated for the total scheduling period (e.g. one day). Between two conflicting events, e.g., the departure events of two trains that would violate the headway constraints on a track, are defined by so-called *periodic constraints*. For a cyclic timetable these constraints satisfy the needed safety requirements, like respecting of the minimal headway times on a track or the forbid of overtaking on tracks. They can be written as

$$l_{ij} \leq (\pi_j - \pi_i) \bmod(T) \leq u_{ij}$$

with integer variables $\pi_i, \pi_j$ for the events $i, j \in V$, lower bounds $l_{ij}$ and upper bounds $u_{ij}$ for the time interval between the events $\pi_i$ and $\pi_j$, and the total timeperiod $T$.

Since, it is hard to solve a model that contains the modulo operator, one substitutes this operation by introducing binary variables $p_{ij}$ for each constraint, and rewrite these constraints as

$$l_{ij} \leq \pi_j - \pi_i + p_{ij}T \leq u_{ij}.$$

The PESP than be formally defined as follows (taken from Möhring, 2006, chapter 6).

**(PESP)** *Let be given a digraph $D = (V, A)$, edge restrictions $\Delta_{ij} = [l_{ij}, u_{ij}]$ (valid time-windows) and a digit $T$. We search a potential $\pi \in \mathbb{R}^{|V|}$ meeting*

$$l_{ij} \leq \pi_j - \pi_i + p_{ij}T \leq u_{ij} \quad \textit{for each } (i,j) \in A.$$

*Leaning on the practical component of this problem, $T$ is called* timeperiod *and $\pi$* timetable.

**Non-cyclic timetabling** The non-cyclic TTP takes generally effect in long-distance freight haulage that uses infrastructure with limited capacity resulting from a high traffic density or infrastructure on that a high growth of competition in the near future is expected. Therefore, it is also denoted as Freight Transportation Problem by Cacchiani (2009) or Track Allocation Problem by Borndörfer and Schlechte (2007b). It has a lower priority in the scheduling step as the cyclic one, and therefore it is deployed directly after this one. Thus, the regarded data and generated datastructures (mostly the time-space graphs) have to be preprocessed to fulfill each safety regulations and not to destroy the schedules created in the previous step.

Since this diploma thesis has his focus on the non-cyclic TTP we use in the following the abbreviation TTP for this kind of problem types.

## 3.2 Definitions

Before formally defining the TTP we need to introduce some basic terms in this chapter. We would like to point out that the regarded models are all discretized in time (e.g., in one-minute steps), and that the following definitions are adapted to this fact.

**Infrastructure Digraph** We represent the given real-world infrastructure in form of a directed graph $N = (S, J, C)$. Each node $s \in S$ of this graph represents a real station or a pseudo station—i.e., an artificial node that we need to model some railway junctions/crossings (see Section 2.4)—of the regarded infrastructure, the arcs $j \in J$ the (long-distance) railtracks and the elements $c \in C$ represent the regarded traintypes. In the following we call $N$ *infrastructure graph* or *infrastructure network*, the elements of $S$ *stations*, the elements of $J$ *tracks* and the elements of $C$ *traintypes*.

**Running times** As it seems to be clear all trains $c \in C$ have in general different *driving times*, at least on each track $j \in J$ one. Since these values depend on multiple factors (see also Section 2.3), it is hard to model accurately. For simplicity, we assume in this work that each train $c$ has only one journey time on a track $j$ denoted with $d_{c,j} \in \mathbb{R}$ or $d_{c,j} = \infty$ if $c$ is not allowed to pass over $j$.

**Driving dynamics** One may also model this aspect by regarding different so-called driving modes for each track which describe whether a train stops or passes on the arrival and departure stations (the four possible modes are stop-stop, stop-pass, pass-stop and pass-pass). This leads possibly to a better—but also not perfectly satisfying—modeling of the real-world situation and is actually realized in `Ts-Opt`. It takes also effect in our

implementations but is for simplicity not regarded in this thesis. Instead, we assume that each train stops in each visited station.

**Headway times** Similar problems occur for the minimum headway times. The calculation of these times depends—much like the driving times—on multiple factors which was discussed in Chapter 2. We simplify this to the question which train $c_1 \in C$ follow another train $c_2 \in C$ on track $j$ and denote this minimal headway time with $\delta_j(c_1, c_2)$. We assume in this work, that the headway matrices hold the triangle inequality $\delta_j(c_1, c_3) \leq \delta_j(c_1, c_2) + \delta_j(c_2, c_3)$ for each traintypes $c_1, c_2, c_3 \in C$ and track $j \in J$ as described in Lukac (2004).

An infrastructure network can be graphically represented as a hyper-graph like in Figure 3.1. The trip arcs are labeled with the appropriate journey times, and the headway matrices are written on a hyper-arc between two (generally the same) trip-arcs that are in a headway condition.

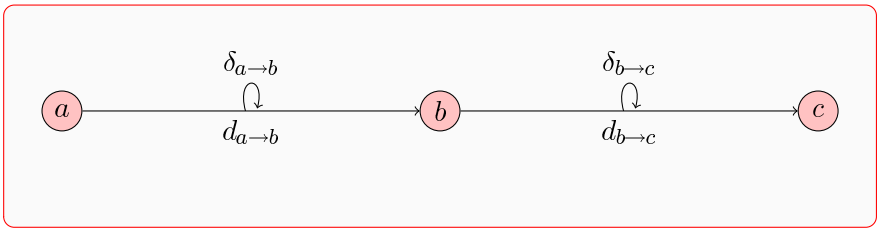



*Figure 3.1:* Graphical representation of an infrastructure digraph.

For the later models it is also important to consider the assumption that trains can only overtake in stations but not on tracks. Overtaking lanes that exist in the real-world on a trackage are modeled as separate tracks with the help of pseudo stations (see also Chapter 2).

**Definition 2 (Stop Request, Train Request)** We call a triple $(s, \Delta^a, \Delta^d)$, being composed of a station $s \in S$ and time windows $\Delta^a = [t^a_l, t^a_u]$ for the arrival and $\Delta^d = [t^d_l, t^d_u]$ for the departure events in $s$, a *stop request*. A *train request* $i = (R, c)$ is a pair of a list—i.e., an ordered set—of stop requests $R_i$ containing at least two elements and a traintype $c(i) \in C$. We call the appropriated list of stations *stop-list* and denote it with $S_i$. □

**Remark 1** In the following we use the terms *train*, *train request*, *slot request* and *request* interchangeably for $i$. □

We interpret a request $i$ as a desire to route a train of type $c(i)$ through a infrastructure $N$ starting in the first and ending in last station of $S_i$. The attached list of stop requests $R_i$ specifies in which stations and in which timewindows intermediate stops must be made. This reduces in a manner the degree of flexibility of routing a train through $N$.

**Graph representation** Each request $i$ can also be represented as a special time expansion $D_i = (V_i, A_i, \sigma_i, \tau_i)$ of an infrastructure $N = (S, J, C)$ in which each $(\sigma_i, \tau_i)$-path is satisfying the requirements of $i$—with $\sigma_i, \tau_i \in V_i$ being artificial source and sink nodes connected with the time-expansions of the first respectively last stop request of $i$.

The nodes $v \in V_i \setminus \{\sigma_i, \tau_i\}$ represent arrivals and departures of a train at a discretized time instant in a station of $S$.

The arcs $a \in A_i$ represent trips between two stations, waiting, starting or ending in a station. To be more accurate, denote the nodes associated with an arrival in a station $s$ with $W_i^s$ and the nodes associated with a departure with $U_i^s$, we distinguish four types of arcs in $A_i$. We call arcs $(u, v) \in A_i$ with ...

(a) ... $u \in U_i^{s_1}$ and $v \in W_i^{s_2}$ *transit* or *journey arcs* $(s_1, s_2 \in S)$,

(b) ... $u \in W_i^s$ and $v \in U_i^s$ *waiting* or *station arcs* $(s \in S)$.

(c) ... $u = \sigma_i$ and $v \in U_i^s$ *artificial source arc*, $s \in S$ first station in $S_i$.

(d) ... $u \in W_i^s$ and $u = \tau_i$ *artificial target arc*, $s \in S$ last station in $S_i$.

After this explanations we can introduce the following definition.

**Definition 3 ((Individual) Train Routing Digraph)** We call the time-expansion $D_i = (V_i, A_i, \sigma_i, \tau_i)$ of the infrastructure $N$ and the request $i$ *individual train routing digraph*, if each node $v \in V_i$ (representing an arrival or an departure event in a station) satisfies a possibly given timewindow restriction of $i$ and each journey arc $a \in A_i$ takes the time specified by the driving time of the traintype $c(i)$.

Given a set $I$ of requests we call the multi-graph $D_I := \cup_{i \in I} D_i$ *train routing digraph.* □

If it is clear which set of requests is meant, we skip the index and simply say that $D$ is a train routing digraph. We denote the station associated with a node $v \in V_i$ with $s(v)$, the time of existence with $t(v) \in \mathbb{Z}$ and the traintype with $c(v)$. The track affiliated with an arc $a \in A_i$ is symbolized with $j(a)$.

**Remark 2** Assuming that all driving times are strict positive one can observe that a Train Routing Digraph is acyclic. □

**Definition 4 (Route)** We call a path $p = (v_1, \ldots, v_l)$ a *route* for the request $i$, if $p$ is a time-expanded $(\sigma_i, \tau_i)$-path meeting all requirements of $i$. We denote the set of all valid paths for a request $i$ with $P_i$ □

**Remark 3** In literature, many authors also use the term *timetable* for a route. To prevent misunderstandings we try to avoid this. □

**Objective** A train operator has the possibility to choose a preferred path contained in a request $i$ by delivering a profit function $w_i : P_i \to \mathbb{R}$. Since the set of all possible paths of a request is generally not known, one is attempt to choose a simple profit function which only uses parameters specified by the request. An often used approach is to attach not only timewindows $\Delta = [t_l, t_u]$ for a stop request, but also optimal scheduling time $t_o$ for each timewindow. Each path for the request is rewarded with a basic value, say $b$, and punished with a penalty function $p$ reducing the profit for each station-visit differing from the preferred optimal schedule. The authors of Erol et al. (2008) and Borndörfer et al. (2006) use a penalty function which is linear in the below described matter.

They define so-called *time-value specifications* $(t_l, t_o, t_u, p_l, p_u)$ that are associated each with a timewindow of stop request, and is needed to define the penalties for arrivals and departures at each station. The first and last station of the appropriate stop-list are associated with exactly one departure/arrival time-value specification and each station in-between with exactly two specifications, one for the arrival and one for the departure at these stations. The penalty for a scheduled time $t \in [t_l, t_u]$ is defined as

$$p(t) = \begin{cases} |t_o - t| \cdot p_l, & \text{if } t < t_o, \\ |t_o - t| \cdot p_u, & \text{else.} \end{cases}$$

See also figure 3.2 for a scheme of the profit function used by the `TTPlib` (see Erol et al., 2008, access at `http://ttplib.zib.de`).

Given such time-value specifications, it is easy to construct a basic value function $b : I \to \mathbb{R}$ for a request-set $I$ and penalty functions $p_i : V_i \to \mathbb{R}$ $(i \in I)$, since each node in $V_i$ is associated with exactly one point in time. One can transform this into an arc-based profit function $w_i : A_i \to \mathbb{R}$ as follows

$$w_i(a) := \begin{cases} -p_i(u) - p_i(v), & a = (u, v) \in A_i \text{ transit or waiting arc,} \\ b(i) - p_i(v), & a = (\sigma_i, v) \in A_i \text{ artificial source-arc,} \\ 0, & \text{else.} \end{cases}$$

One can easily check that the penalty based profit value $b(i) - \sum_{v \in V(p)} p_i(v)$ equals the transit-arc-profit based profit value $\sum_{a \in A(p)} w_i(a)$ of a route $p$.

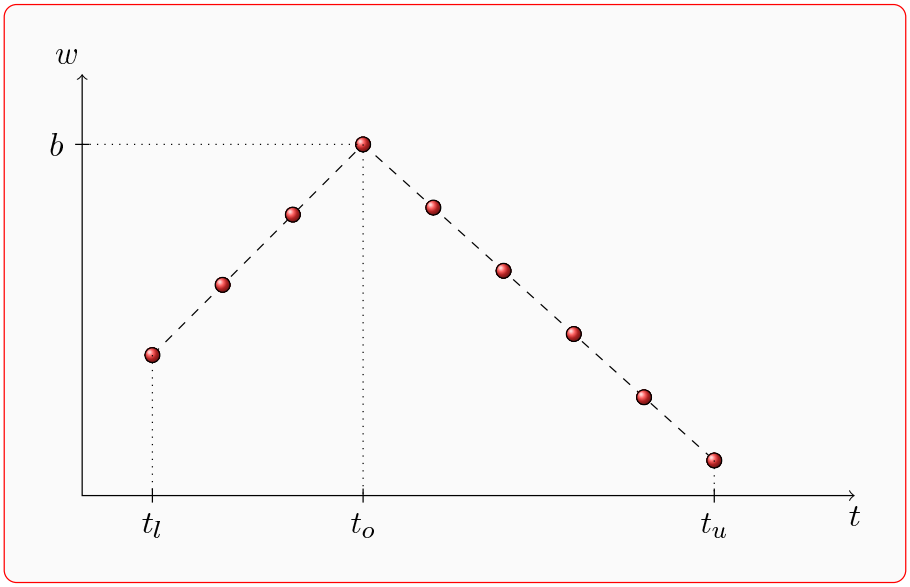



*Figure 3.2:* Profit function used in the `TTPlib` for the starting station.

Given such a transit-arc based profit function, we denote the profit value of a route $p \in P_i$ with $w_i(p) = \sum_{a \in p} w_i(a)$ —since each route is associate with exactly one request, we can even omit the index $i$. Analogous, we denote the profit value $\sum_{p \in X} w(p)$ of a set of routes $X$ with $w(X)$.

As already mentioned in section 2.3, earlier works introduce the so-called block-conflict to handle the safety regulations on each track. The following definition introduces the common conflict types used in newer models and which is needed for this work.

**Definition 5 (Conflict, Timetable)** Given an infrastructure network $N = (S, J, C)$, a set of requests $I$ and the associated train routing digraph $D = (V, A)$.

(a) We say that two track-arcs $(u_1, v_1), (u_2, v_2) \in A$ are in *headway conflict* if they are the expansion of the same track $j$—i.e., $j((u_1, v_1)) = j((u_2, v_2))$ —and are violating the *headway constraint* $\delta_j(c_1, c_2) \leq t(u_2) - t(u_1)$, where $c_1 := c((u_1, v_1))$ and $c_2 := c((u_2, v_2))$.

(b) We say that two routes $p_1$ and $p_2$ are in *headway conflict* if there exist two arcs $a_1 \in p_1$ and $a_2 \in p_2$ which are in conflict.

(c) We say that a set of routes $X$ has a *headway conflict* if two routes $p_1$, $p_2 \in X$ exist that are in conflict.

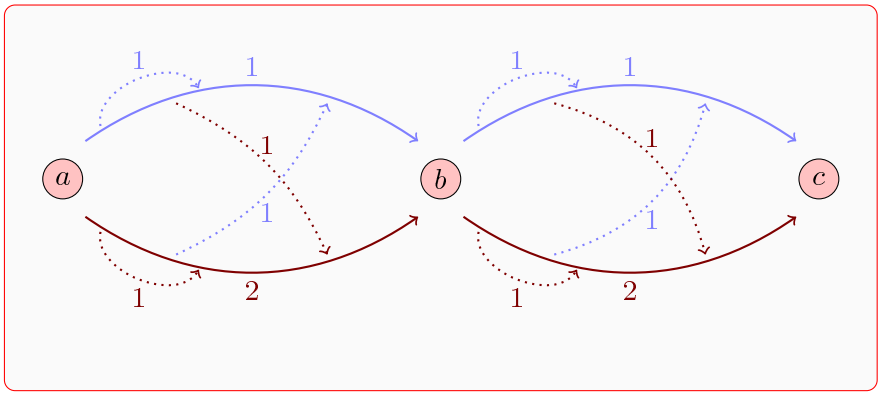



*Figure 3.3:* Graphical representation of an infrastructure digraph.

(d) We say that a set of routes $X$ has a *station conflict* if a station $s \in S$ exist whose *station capacity* $\gamma_s$ ($\in \mathbb{Z}_+$) is exceeded for a time instant $t \in \mathbb{Z}$—i.e., $|\{(u,v) \in A : s = s(u) = s(v),\ t(u) \leq t < t(v) \vee t(u) = t = t(v)\}| \leq \gamma_s$ is violated.

(e) A set of routes $X$ has a *conflict* if it has a headway conflict or a station conflict.

In an analog way, we define the terms *headway-conflict-free*, *station-conflict-free* and *conflict-free*.

We call a set of conflict-free routes *train timetable* or shorter *timetable*. □

After this introduction into the needed data we can finally define the TTP in a formal way.

**Definition 6 (Train Timetabling Problem (TTP))** Let be given an infrastructure digraph $N$, a set of requests $I$ and a profit function $w$. The *train timetabling problem* (TTP) is the question to find a timetable $X^*$ with

$$X^* = \arg\max\{w(X) :\ X \subseteq P, |X \cap P_i| \leq 1\ \forall i \in I,\ X \text{ timetable}\}.$$ □

After these abstract definitions we want to introduce an example of how-to create the train routing graph.

**Example 2 (Constructing the problem graph)** Consider the network graph drafted in Figure 3.3. Assume that the appropriate infrastructure consists of two traintypes (call ICE and ICG) whose trains need one respectively two timeunits to pass the given tracks

and have each to respect a minimal headway of one minute on each track. With mathematical terms, we are given an infrastructure network $N = (S, J, C)$, with

| | |
|---|---|
| stations | $S = \{a, b, c\}$, |
| tracks | $J = \{a \to b, b \to c\}$, |
| traintypes | $C = \{\text{ICE, ICG}\}$, |
| running times | $d_{\text{ICE},j} = 1,\ d_{\text{ICG},j} = 2,$ for all $j \in J$ and |
| minimal headway times | $\delta_j(c_1, c_2) = 1$ for all $j \in J$, $c_1, c_2 \in C$. |

We will try to route two trains in this graph. Both should start in station $a$ and target station $c$, and are allowed to stop in $b$ for an arbitrary time. The first train (blue) should start in the timewindow $[1, 4]$ and arrive in the window $[3, 6]$, where the second train (red) should depart in $[1, 3]$ and arrive in $[5, 7]$. As we see, we obtain a total timehorizon of $T = 7$ for the total train routing graph. In order to complete this example, consider Table 3.1 which also declares the needed valuation (basic value and time-value specifications).

| Request | Basic value | Traintype | Station | Time-value specification $(t_l, t_o, t_u, p_l, p_u)$ |
|---|---|---|---|---|
| blue | 10 | ICE | $a$ | $(1, 3, 4, 1, 2)$ |
| | | | $c$ | $(3, 5, 6, 0, 1)$ |
| red | 10 | ICE | $a$ | $(1, 2, 3, 2, 2)$ |
| | | | $c$ | $(5, 6, 7, 2, 0)$ |

*Table 3.1:* Requestset

Since we distinguish between arrival and departure event in the time-expansion, we split first each station of this network into two sub-nodes like imaged in Figure 3.4.

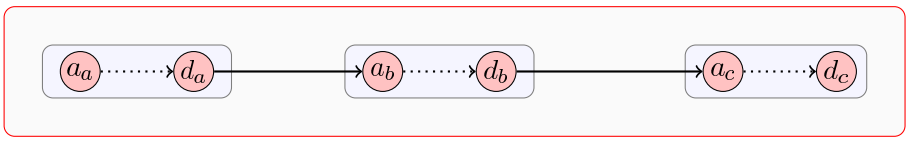



*Figure 3.4:* Infrastructure network (with splitted arrival and departure nodes).

We will now timely expand this graph for each request in order to create the individual train routing graphs which we merge to the train routing graph. For this purpose, we

create first $T$ time-expanded nodes for each sub-node and each train, and artificial source and sink nodes for each requested train. After this step, we can generate the arcs of each individual train routing graph. We connect the artificial source nodes with the possible departure events of each request and the sink node with each appropriate arrival event, generate trip-arcs on each track and each moment that can be used by the requested trains, and generate the possible waiting arcs in each station. As we see, one can now post-process the graph by removing the unused arrival and departure events, and all arcs that can't lie on a $\sigma_i$-$\tau_i$ path for each train $i \in \{\text{red}, \text{blue}\}$. The result of this step is figured in Figure 3.5.

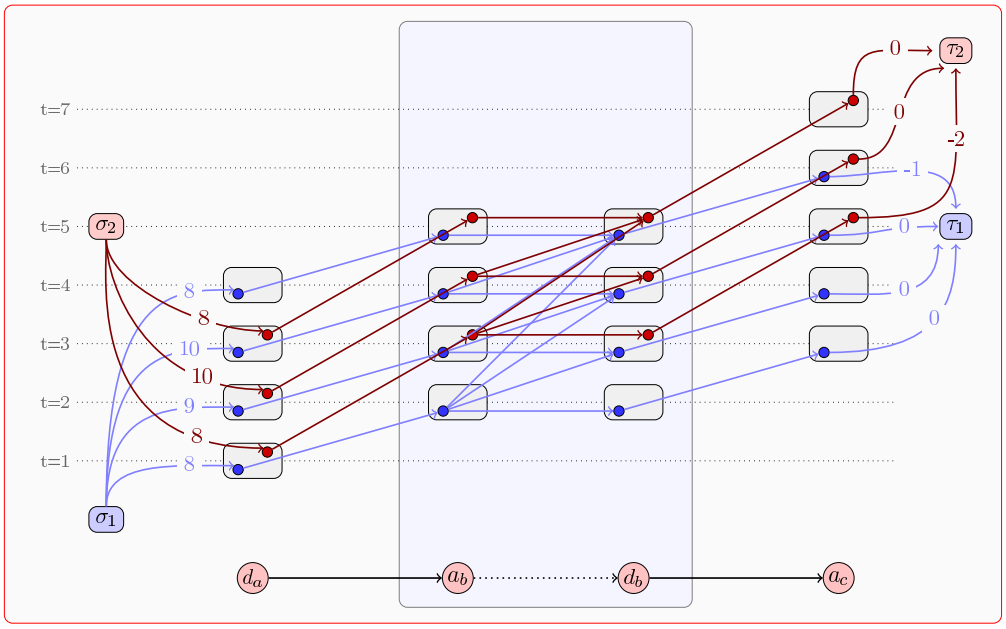



*Figure 3.5:* Train Routing Digraph.

An attentive reader may have noticed that this method is not quite effective for a flow-based definition of train routings. One can improve the construction, i.e., reduce the number of needed arcs, by replacing the waiting arcs (between arrival and departure events) through so-called timeline arcs, i.e., arcs $a = (u, v)$ between two departure (or two arrival) events with appropriate times $t(u)$ and $t(v)$ with $t(u)+1 = t(v)$. The timeline construction of the above train routing digraph is displayed in Figure 3.6.

One can see, that this construction requires $O(n)$ arcs instead of $O(n^2)$ which are needed for a construction with waiting arcs. So, in general this design should be preferred for a (discretized) model. Unfortunately, it is not always possible to use this construction, especially not for our model that we introduce in Chapter 5. □

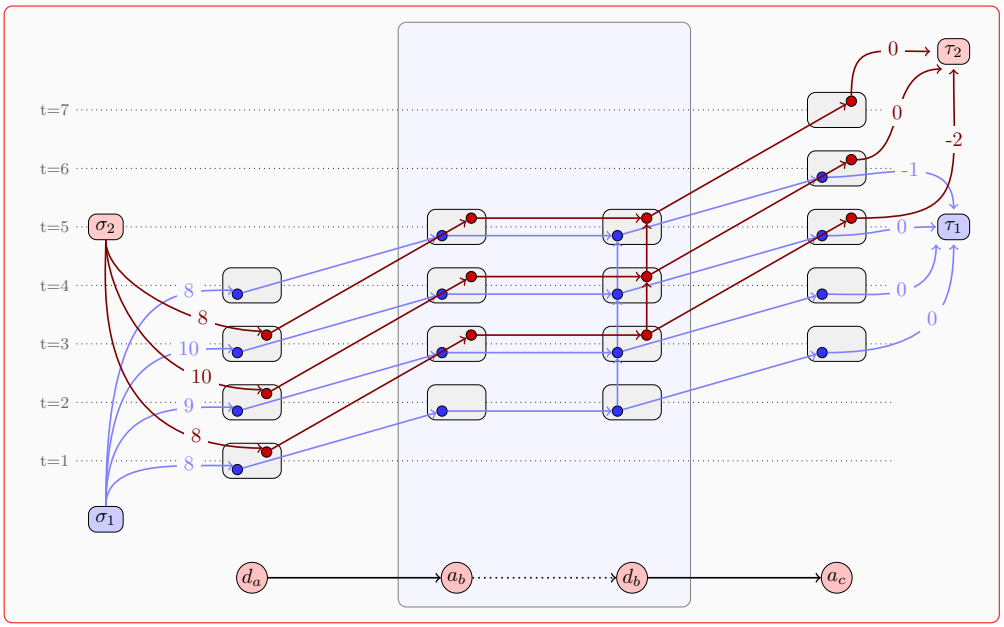



*Figure 3.6:* Train Routing Digraph using a timeline.

# Chapter 4

# Models For The Train Timetabling Problem

When in doubt, expand it out.

*(Albert Einstein)*

In this chapter we introduce models occurring in literature and discuss the pros and cons of each approach. The chapter is structured as follows. The next Section 4.1 introduces the most common approach for solving TTP instances, namely as set packing models. The following Section 4.2 describes a new modeling technique first introduced by Borndörfer and Schlechte (2007b) named as configuration model. The last Section 4.3 of this chapter, we discuss robust timetables and describe a model for these schedules.

## 4.1 Set Packing Models

The most common solving methods for the TTP base on MIP formulations. The models describe either a multi-commodity flow with arc variables or a coressponding column generation approach with path variables. Both variants use additional set packing constraints in order to rule conflicts out. See also Caprara et al. (2002), Borndörfer and Schlechte (2007a), Cacchiani (2009) and Brännlund et al. (1998).

One can distinguish between two basic packing approaches. The first kind are arc-based models similar to the following *arc packing model* (APP). This model is the most basic approach and can therefore be characterized as standard model.

$$
\begin{aligned}
\text{(APP)} \qquad \max \quad & w^T x & & & \\
\text{s.t.} \quad & x(\delta_i^+(v)) - x(\delta_i^-(v)) = 0 & \forall\, i \in I, v \in V_i \setminus \{\sigma_i, \tau_i\} & \text{(i)} \\
& x(\delta_i^+(\sigma_i)) \leq 1 & \forall\, i \in I & \text{(ii)} \\
& x(c) \leq \kappa_c & \forall\, c \in C & \text{(iii)} \\
& x \geq 0 & & \\
& x \in \mathbb{Z}^A & &
\end{aligned}
$$

Each variable $x_a$ of the APP model represents an arc $a \in A$ of the train routing digraph which takes a positive value if a train $i$ is scheduled that uses $a$, and zero else. The model consists of flow-conservation (i) and flow-capacity constraints (ii) which make sure that each train $i \in I$ is scheduled at most once on a $\sigma_i$-$\tau_i$ path, and set packing constraints (iii) which forbid conflicts $c \in C$ in the final schedule. Thereby, the elements of the so-called conflict-set $C \subseteq 2^A$ represent sets of conflicting arcs. A final schedule is only allowed to contain at most $\kappa_c$ arcs of each conflict $c \in C$.

Note that the variables of this model are implicitly binary due to the flow constraints (i and ii) and the definition of the train routing graph as disjoint union of each individual train routing graph.

The second kind of packing models are based on path variables similar to the below *path packing model* (PPP). In general, the variables of these models are generated dynamically and so capable to handle large-scale instances.

$$
\begin{aligned}
\text{(PPP)} \qquad \max \quad & w^T x & & \\
\text{s.t.} \quad & x(P_i) \leq 1 & \forall\, i \in I & \text{(i)} \\
& x(P_c) \leq \kappa_c & \forall\, c \in C & \text{(ii)} \\
& x \geq 0 & & \\
& x \in \mathbb{Z}^P & &
\end{aligned}
$$

The variables $x_p$ of this model correspond to $\sigma_i$-$\tau_i$-paths $p \in P_i$ in the appropriate individual train routing digraph of the request $i \in I$. The PPP model consists of set-packing constraints (i) which make sure that each train is scheduled at most once, and

set packing constraints (ii) forbidding conflicts in the final timetable. In an analogue way to the APP each conflict $c \in C$ implice a maximal number $\kappa_c$ of paths which are allowed to be scheduled simultaneously in a final schedule.

Note again that the variables of PPP are implicitly binary due to the fact that each path is contained in exactly one bundle constraint.

The conflicts that are ruled out differ by each author but can in general be summarized as headway or station conflicts. In the following section we want to have a closer look to the models of other authors and their regarded conflict sets.

### 4.1.1 Literature Overview

To the best of our knowledge, Brännlund et al. (1998) is the first work that analyzes the TTP with combinatorial optimization techniques. The authors resolve the following model that is discretized in minutes and based on block occupation conflicts.

$$\begin{aligned} \max \quad & w^T x & & \\ \text{s.t.} \quad & \sum_i x^i_{bt} \leq 1 & \forall\, b \in B,\, t = 1, \ldots, T & \quad \text{(i)} \\ & x^i \in T^i & \forall\, i \in I & \end{aligned}$$

Each binary variable $x^i_{bt}$ of this model takes the value one iff the train $i \in I$ occupates the block $b \in B$ in the moment $t = 1, \ldots, T$, and zero otherwise. The *track capacity constraints* (i) ensure that no two trains are scheduled that occupy the block $b \in B$ at the same moment $t$. The sets $T^i$ are described as the sets which contain all vectors that result in *technically and logistically feasible schedules for the train $i$, not considering the effect of other trains on the track*, and especially the empty schedule. The authors solve instances of this model on an infrastructure network representing a corridor with 17 stations for 26 and 30 requested trains, by relaxing the constraint (i) into the objective by the Lagrangian method.

Caprara et al. (2002) (see also Cacchiani, 2009) extend the APP approach by new binary variables $z_{iv}$ expressing whether a train $i \in I$ is visiting the station event $v \in V$ or not, and binary variables $y_v$ describing if the station event $v$ is used or not, i.e.,

$$z_{iv} = \sum_{a \in \delta_i^-(v)} x_a \quad \forall i \in I, v \in V_i$$

and

$$y_v = \sum_{i \in I} z_{iv} \quad \forall v \in V.$$

These are used to express the conflict constraints (APP-iii)—called *track capacity constraints* by the authors—that are given in the below listed three classes. In order to express that a node $u$ preceeds another node $v$ in the cyclic order, the authors use the notation $u \preceq v$, and the timely distance between the nodes $u$ and $v$ is written as $\Delta(u, v)$, i.e.,

$$\Delta(u,v) := \begin{cases} t(v) - t(u), & \text{if } t(v) \geq t(u) \\ t(v) - t(u) + q, & \text{otherwise} \end{cases}$$

with $q$ denoting the timehorizon.

Moreover, the set of arrival events at a station $s$ is denoted with $W^s$ and the departure events at $s$ with $U^s$.

(a) *Arrival constraints* for each track that forbid arrivals that are too close in time, due to the minimal headway time requirements for arrival events.

$$\sum_{\substack{w \in W^s : u \preceq w, \\ \Delta(u,w) \leq \delta_s^{\text{arr}}}} y_w \leq 1 \quad \forall s \in S,\ u \in W^s$$

(b) *Departure constraints* for each track that forbid departures that are too close in time, due to the minimal headway time requirements for departure events.

$$\sum_{\substack{w \in U^s : \\ u \preceq w, \Delta(u,w) \leq \delta_s^{\text{dep}}}} y_w \leq 1 \quad \forall s \in S,\ u \in U^s$$

(c) *Overtaking constraints* that forbid overtaking on a track and defined as follows. Considering two trains $i_1$ and $i_2$ and a track $j = (s_1, s_2)$, than are these constraints given for a pair of trip arcs $(v_1, u_1) \in A_{i_1}$ and $(v_2, u_2) \in A_{i_2}$ in the form

$$z_{i_1 v_1} + z_{i_1 u_1} + z_{i_2 v_2} + z_{i_2 u_2} \leq 3$$

with $v_1 \preceq v_2$ and $u_1 \succeq u_2$ for departure events $v_1, v_2 \in U^{s_1}$ and arrival events $u_1, u_2 \in W^{s_1}$. The authors use in there implementations the following tighter form of these constraints

$$\sum_{w \in U^{s_1} \cap V_{i_1} :\, w \preceq v} z_{i_1 w} + \sum_{w \in W^{s_2} \cap V_{i_1} :\, w \succeq u} z_{i_1 w} + \sum_{w \in U^{s_1} \cap V_{i_2} :\, w \succeq v} z_{i_2 w} + \sum_{w \in W^{s_2} \cap V_{i_2} :\, w \preceq u} z_{i_2 w} \leq 3$$

for each departure event $v \in U^{s_1}$ and arrival event $u \in W^{s_2}$ with $v_1 \preceq v \preceq v_2$ and $u_2 \preceq u \preceq u_1$.

Cacchiani (2009) varies this model by aggregating both headway constraints, and introduces so-called *crossing constraints* that forbid crossing between trains traveling in opposite directions.

The authors solved the model by relaxing the above listed track capacity constraints with the Lagrangian method, with a so-called relax-and-cut approach, i.e., with dynamically generating these constraints. They describe as a main advantage of this approach, that the profit function of the $x$ variables is unchanged, and that near-optimal Lagrangian multipliers can be found through subgradient optimization.

The authors report solutions for 17 instances with up to 500 trains for a underlying network, that represents a corridor with up to 73 stations, during a timehorizon of one day, discretized in one minute steps, with optimality gaps between 0.6% and 20.1%.

Borndörfer and Schlechte (2007a) regard an APP approach with special headway conflicts that can be interpreted as block occupation conflicts. They show that the block conflict graph is an interval graph which is known to be perfect. They show that block conflicts are maximal cliques in this graph which can be seperated in polynomial time due to the perfectness. Since optimization and separation are equivalent (see Grötschel et al., 1981) it follows that the linear relaxation of APP with these block conflicts can be solved in polynomial time with respect to the encoding length of the considered train routing digraph. So, the main advantage of this model is that it can easily be used in an exact cut-generation approach.

In the earlier work Borndörfer et al. (2006), an extension of this model (called xOPTRA) is analyzed and proposed for an auction based allocation of track slots. The model in this work includes additional constraints that are used to express AND and OR bids of an advanced auction language.

## 4.2 Configuration Models

Borndörfer and Schlechte (2007a) introduced a new solving approach for the TTP. Instead of separating packing constraints and so forbidding not allowed track schedules, they extend the original model by artificial arcs $\widetilde{A}$ on each track representing the headway condition of two succeeding trains.

Before we explain this in more detail, we want to introduce the following definition.

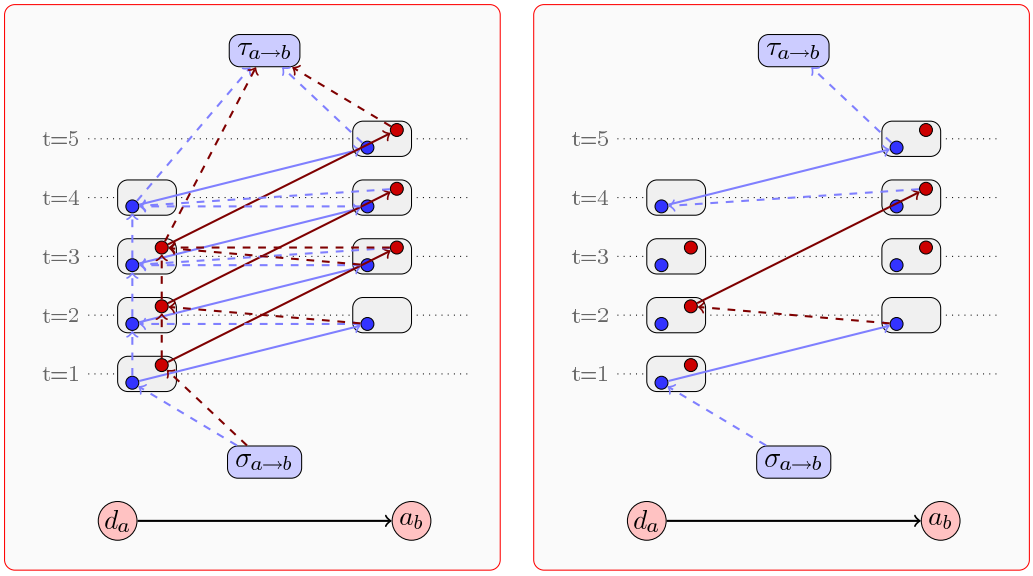
**Definition 7 (Track Configuration)** We call a set $A \subseteq A_j$ of transit arcs associated with a track $j \in J$ *track configuration*, if

$$t(u_1) \leq t(v_1) \Rightarrow\ t(v_1) \geq t(u_1) + \delta_j(c(u_1, u_2), c(v_1, v_2)).$$

for each $(u_1, u_2), (v_1, v_2) \in A$ with $(u_1, u_2) \neq (v_1, v_2)$. □

The authors create for each track $j = xy \in J$ a digraph $D_j = (V_j, \overline{A}_j, \sigma_j, \tau_j)$. Whereby $\sigma_j$ and $\tau_j$ are artificial source and sink nodes for this graph, $V_j := U_x \cup W_y \cup \{\sigma_j, \tau_j\}$ is the union of these artificial nodes together with arrival events $W_y$ and departure events $U_x$ associated with this track, and an arcset $\overline{A}_j := A_j \cup \widetilde{A}_j \cup \widehat{A}_j$ with $A_j$ denoting the transit arcs related with $j$, $\widetilde{A}_j$ being the set of artificial arcs whose elements either connect arrival events in $y$ with possible next departure events in $x$ that does not violate the headway condition or elements of $V_j \setminus \{\sigma_j, \tau_j\}$ with artificial nodes, and $\widehat{A}j$ being the set of timeline arcs that connect an arrival event at time $t$ with a appropriate event at time $t+1$. We call the elements of $\widetilde{A}_j$ connecting departure events in $y$ with arrival events in $x$ also headway arcs. See also Figure 4.1a for an example of this graph.

**Definition 8 (Track Digraph, Configuration Route)** Let $j \in J$ be a track from station $x$ to $y$, and $D = (V, A)$ be the train routing digraph for a regarded set of requests. We call the above described digraph $D_j = (V_j, \overline{A}_j, \sigma_j, \tau_j)$ *track digraph* and a $\sigma_j$-$\tau_j$ path in $D_j$ *configuration route*. □




*(a)* Trackgraph *(b)* Configuration routing

*Figure 4.1:* Track digraph (a) and a configuration routing (b) for track $a \to b$ of Example 2.

One can easily see that the set of transit arcs of each configuration route describes a track configuration. Note at this point, that each headway matrix satisfies the triangle inequality and, therefore, no transitive violation of the headway conditions can occur. The idea is now to decompose the timetabling problem into two elementary subproblems,

firstly to find a valid route for each request, and secondly to find valid track configurations. Coupling both problems within a mixed integer program ACP leads to headway conflict-free timetables.

$$
\begin{aligned}
\text{(ACP)} \qquad \max \quad & w^T x & & \\
\text{s.t.} \quad & x(\delta_i^+(v)) - x(\delta_i^-(v)) = 0 & \forall\, i \in I,\, v \in V_i \setminus \{\sigma_i, \tau_i\} \qquad & \text{(i)} \\
& x(\delta_i^+(\sigma_i)) \leq 1 & \forall\, i \in I \qquad & \text{(ii)} \\
& y(\delta_j^+(v)) - y(\delta_j^-(v)) = 0 & \forall\, j \in J,\, v \in V_j \setminus \{\sigma_j, \tau_j\} \qquad & \text{(iii)} \\
& y(\delta_j^+(\sigma_j)) \leq 1 & \forall\, j \in J \qquad & \text{(iv)} \\
& x_a - y_a \leq 0 & \forall\, j \in J\, \forall\, a \in A_j \qquad & \text{(v)} \\
& x \geq 0 & & \\
& y \geq 0 & & \\
& x \in \mathbb{Z}^A & & \\
& y \in \mathbb{Z}^{\overline{A}} & &
\end{aligned}
$$

The variables $x_a$ of this model represent arcs $a \in A$ of the train routing digraph, taking a positive value if a train $i$ is scheduled that uses the arc $a$ and zero otherwise. Each variable $y_a$ represents an arc $a \in \overline{A}$ of the track digraph of $j \in J$, and takes a positive value if the arc is contained in a final configuration routing of $j$ and zero otherwise. The model consists of flow-constraints (i)-(iv) that ensure that each train $i \in I$ is routed at most once on a $\sigma_i$-$\tau_i$ path, and that each track configuration $j \in J$ is routed at most once on a $\sigma_j$-$\tau_j$ path. The coupling constraints (v) synchronize paths and configurations, i.e., they ensure that there is for each trip arc of a scheduled train an active configuration that also contains this arc. Note that all constraints imply that the variables are implicitly binary.

The authors show that the pricing problem for both subproblems can be solved as a longest-path problem in an acyclic graph which can be solved in polynomial time. The appropriate column generation approach is summarized in the following TCP.

$$
\begin{array}{llrll}
\text{(TCP)} & \max & w^T x & & \\
& \text{s.t.} & x(P_i) \leq 1 & \forall\, i \in I & \text{(i)} \\
& & y(Q_j) \leq 1 & \forall\, j \in J & \text{(ii)} \\
& & x(P_a) - y(Q_a) \leq 0 & \forall\, j \in J\, \forall\, a \in A_j & \text{(iii)} \\
& & x \geq 0 & & \\
& & y \geq 0 & & \\
& & x \in \mathbb{Z}^P & & \\
& & y \in \mathbb{Z}^Q & &
\end{array}
$$

The variables $x_p$ and $y_q$ represent paths in the train routing graph respectively configuration routes in the track digraphs. They take a positive value, if the appropriate path or configuration is routed and zero otherwise. The constraints (i) and (ii) ensure that at most one route per train and one configuration per track is routed. Finally, the coupling constraints (iii) synchronize paths and configurations. Note that, again, all variables are implicitly binary.

In order to shorten the notation, we used the sets $Q_j$, $P_a$, $Q_a$, $Q$ that denote the set of all feasible track configurations on a track $j$, the set of train and configuration routings that contain the trip arc $a$, and the set of all configurations.

Since pricing and optimization is equivalent, the linear relaxation of TCP can be solved in polynomial time. Hence, the TCP seems currently to be the best approach to handle large-scale instances, i.e., instances not manageable by the static models APP and ACP.

The main advantage of both models, ACP and TCP, results directly of the fact that no separation step is necessary to fulfill each headway condition. Since this conditions are guaranteed by the configuration part of the model, each integer solution that is found during the solving process represents a valid timetable.

### 4.2.1 Literature Overview

Erdoğan (2009) analyzes the model ACP with the purpose of finding efficient solving methodologies for an auction based track allocation. Such an approach leads to the necessity of resolving efficiently the TTP multiple times in order to calculate the (Vickrey) prices of the auction and to solve the next round of the auction.

Firstly, he suggests to implement a warmstart technique to improve the recalculation of the problem.

In a second stage, he could improve the model by replacing the coupling constraints ACP-(v) through aggregated coupling constraints, i.e., constraints of type

$$\sum_{\widetilde{a}\in[a]} x_{\widetilde{a}} - y_{[a]} \leq 0 \quad \forall [a] \in [A],$$

with adequate equivalence classes $[a] \subset A$, and $[A]$ denoting the set of all equivalence classes.

He shows that these constraints are tighter then the original ones, i.e., which indicates that the original constraints are not faces of the TTP.

Lastly, he finds some interesting rules to speed up the branch-and-bound process by proposing to branch mainly on configuration-variables on the one side, and improving the conflict analyzes through better rules on the other.

## 4.3 Profit versus Robustness

Using railway roads as efficient as possible is the mayor goal of train timetabling. But what happens if one or more trains are delayed and can't keep with the ideal timetable? This is the rule on stations and tracks operating at full capacity. Since the European Union expects an rise of the freight traffic volume of 64.1% from 1997 to 2015, it seems to be a good idea to analyze this question with methods of mathematical proramming.

In this section we take a look at the aspect of *robustness* for the train timetabling problem and discuss an approach to create so-called *robust timetables*. Since robustness can be mistaken for robustness in (linear) optimization give a short introduction in LP-robustness and try to explain the difference to the robustness term regarded in this section.

### 4.3.1 Robust Optimization

Edward Lorenz found in 1963 "by accident" out that even small changes in the input values can lead to unexpected large differences in the output. He entered a wrong series of numbers (0.506 instead of 0.506127) and got a totally different solution then expected for his weather model. This phenomenon also got known as the *butterfly-effect* which also shows that uncertainties in the input values may hard to be handle. He launched in that way a new branch of mathematics which is also known as theory of complex systems—or more informal as chaos theory

So-called *robust optimization methodologies* try to handle similar problems for combinatorial optimization problems. The prerequisite for the appliance of classical mathematical programming techniques is generally the knowledge of the input data. But if parts of the dataset or even the whole dataset are affected by uncertainty, i.e., are not measurable or predictable, these solving methodolgies reach their limits. In such a case, a common approach is to estimate the data and to search a solution with these values, leading often to solutions that are not valid for the actual, real-world data. Robust models handle this by regarding the given data as independent, symmetric and bounded random variables and deliver solutions, that are valid for each possible value of these variables. So, assume that we are given a problem of the following type

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & l \leq x \leq u \end{aligned}$$

with a matrix $A$ affected by uncertainty. To be more precisely, let $J_i$ be the set of coefficients affected of uncertainty in row $i$, and the entries $a_{ij}, j \in J_i$ be a symmetric, bounden random variable with values in $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$ ($a_{ij}$ is the expected, nominal value and $\hat{a}_{ij}$ is the maximal deviation of this). Then, the following model delivers solutions that are valid for these uncertainties.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & \sum_j a_{ij} x_j + \sum_{j \in J_i} \hat{a}_{ij} |x_j| \leq b_i \quad \forall i \\ & l \leq x \leq u \end{aligned}$$

It is said, that the terms $\sum_{j \in J_i} \hat{a}_{ij} |x_j|$ determine the degree of robustness.

A main problem is to model the uncertainty adequately, especially to find a representative deviation for the variables. Another problem that occurs is the so-called *price of robustness*, i.e., that the found solutions are robust but also far away from optimality for the nominal values. One way to solve the last problem, is select "better" random variables with deviations that cut-off the improbable corners of the state-room, e.g., an elliptic state room that is used by Ben-Tal and Nemirovski (2000).

For more details in robust programming, we refer to El-Ghaoui and Lebret (1997), Ben-Tal and Nemirovski (2002) and Bertsimas and Sim (2004).

### 4.3.2 Robust Train Timetabling

The term *robustness* differs for each application area. In many optimization tasks, and also for the case of timetabling, robustness refers mostly to the demand of guaranteeing

a specific service quality. So, we obtain a second objective beside the main one that increases the revenue of the infrastructure manager. The regarded service for our case of train timetabling is to "be on time" which means not "much too late" but especially not to be early with respect to the given timetables. The first case is the most common and may have multiple reasons. Delays can be classified in primary and secondary delays, expressing if the regarded train is affected directly by disturbances (e.g., by road works or overcrowded stations) or indirectly by the unpunctuality of another, leading train. The other case of unpunctuality, is the earliness of a train which is often the worst case that can happen to a passenger, since this can lead to miss such train as connection. Therefore, it is in the daily-operation often avoided by reducing the speed of the affected train(s), which may also affect the following trains and leading so to delay for these ones. One should also mention that punctuality is defined differently by each train operator. For example, the German railway operator "Deutsche Bahn" specifies a train as too late if the delay exceeds 5 minutes. Others, for example the Swiss railway assess this value by 1 minute, and again others like the Italian railway by 15 minutes.

*Robust train timetabling* has the goal to create schedules that are delay-resistant by considering punctuality as one of the main *key performance indicators* (KPI), which are used for measuring the quality of solutions with respect to different business objectives. There are different possibilities to reduce delays. The common approach to reduce especially primary (and therewith secondary) delay is to create a timetable that allows trains to run faster than planned in order to make up earlier delays. This is achieved by planning trains with a driving time that is higher than the technically minimum running time and obtain time supplements to other processes (as halting at a station). In general, running time supplements lead to better punctuality of the railway services but also to higher planned running times, and therefore to higher planned travel times for the passenger. Running time supplements may also have negative influence on the realized travel times; on the one hand, increases each minute of supplement the risk that it is not needed, and on the other, increase longer planned running times the block occupation times and therewith the track occupation rates. The common approach to reduce the secondary delays, is to plan additional buffer-times between two successive trains such that the delays cannot be passed on the whole day. Both methods have a negative influence on the achieved profit value of the computed schedule.

The question is now, how to obtain punctuality in a smarter way, i.e., without loosing much of optimality with respect to a profit based objective. There are many publications analzing this question, e.g., Liebchen and Möhring (2008) for approaches to the robust cyclic TTP, and Fischetti et al. (2007) for approaches to the robust non-cyclic TTP. We will now summarize a model published by Borndörfer and Schlechte (2008) that reduces secondary delay by planning additional buffer times by extending in an easy way the ACP model.

Consider the definitions and models given in Section 4.2. We described the arc set $\overline{A}_j := A_j \cup \widetilde{A}_j \cup \widehat{A}_j$ of a track digraph as a the union of the trip arcs of the track $j = (s_1, s_2)$, the timeline arcs $\widehat{A}_j$, and artificial arcs that consist mostly of headway arcs each connecting an arrival event in $s_2$ with the next possible departure events in $s_1$ that does not violate the headway conditions. One can extend this graph by adding additional headway arcs that connect not only arrival events with the next possible departure event but also with departure events that come later in time. Using such a headway arc in a configuration routing can be interpreted as adding supplemental buffer time between two departures in $s_1$ on track $j$. The objective of ACP can now be extended by robustness values for each headway arc rewarding additional buffer times. In an optimal way, the chosen robustness function should build a monoton, concave function in order to prefer a distribution of small buffer times that are distributed over the total set of scheduled train against long buffer times that are distributed over a small set of scheduled trains. Borndörfer and Schlechte (2008) use the following robustness function

$$r(a) := \begin{cases} \sqrt{b} & \text{if } a \in \widetilde{A} \text{ headway arc with } \Phi(a) > b, \\ \sqrt{\Phi(a)} & \text{if } a \in \widetilde{A} \text{ headway arc with } \Phi(a) \leq b, \\ 0 & \text{otherwise} \end{cases}$$

with $\Phi(a)$ denoting the buffer implied through the headway arc $a$, and an integer $b \in \mathbb{N}$ denoting the maximal desired buffer time. The function is also plotted in Figure 4.3.2.

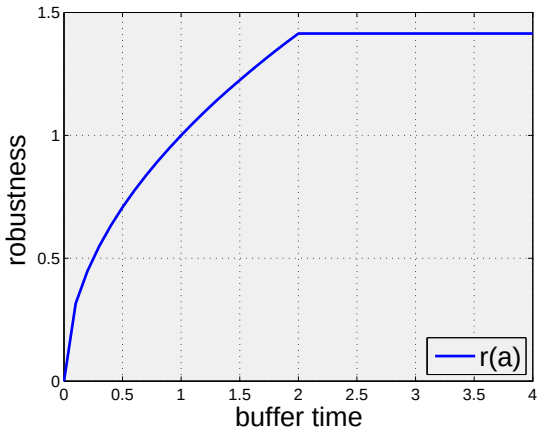



*Figure 4.2:* Robustness function $r(a)$ used in Borndörfer and Schlechte (2008).

Since, it is not clear how important it is to receive a high robustness value for the final

timetable, one has to regard the following bi-criteria optimiziation program.

$$\begin{aligned} \max \quad & w^T x \\ \max \quad & r^T y \\ & (x, y) \in \text{ACP} \end{aligned}$$

One can now solve this MOIP, i.e., obtain the Pareto solutions, by the methods described in Section 1.4.

We implemented in `Ts-Opt` the weighted sum method for the ACP model, i.e., created the IP

$$(\alpha\text{-ACP}) \qquad \begin{aligned} \max \quad & \alpha\, w^T x + (1-\alpha)\, r^T y \\ & (x, y) \in \text{ACP} \end{aligned}$$

and solved it for some $\alpha \in [0, 1]$ obtained by Procedure 2 (see also Figure 4.3).

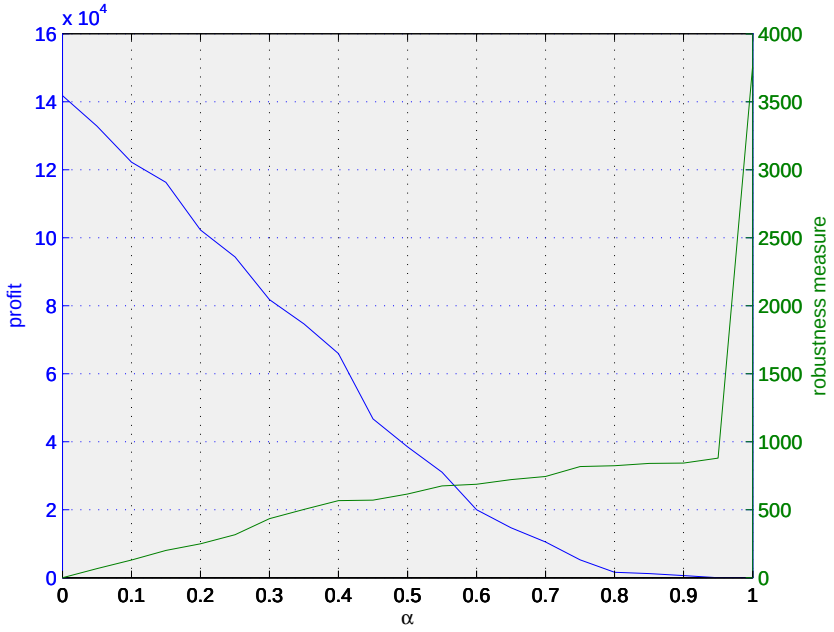



*Figure 4.3:* Tradeoff curve for a run of Procedure 2 for $\alpha$-ACP.

# Chapter 5

# A Station Configuration Model

A mathematician is a device for turning coffee into theorems.

*(Paul Erdős)*

In this chapter, we introduce a new modelling approach for the TTP that was mainly inspired by the TCP model and can handle all of the specifications without attaching a cut-generation step afterwards. The idea is to find a configuration model that considers not only the headway conditions as in the previously introduced configuration models TCP and ACP, but also deals with the station constraints. It can also be seen as the first step of integrating the train platforming problem into the train timetabling step.

This chapter is structured as follows. In the following Section 5.1 we motivate and introduce the new path-based configuration model. The next Sections 5.2 and 5.3 describe the pricing problems of both (path and configuration) variable types. The last Section 5.4 of this chapter gives a short polyhedral characterization of the model.

## 5.1 Motivation

In the previous chapter we have seen that track configuration approaches can successfully handle most of the problem specifications of the TTP. Those restrictions which can't be handled by the basic models—namely the station-capacity restrictions—are regarded afterwards in forms of new constraints in a cut-separation step. But this approach destroys the most important benefit of the configuration approaches, viz. being feasible in each state of the solving process.

In order to find a configuration model respecting the headway constraints as well as station constraints in its basic formulation, remind how we created the train routing digraph and which operational constraints are given. We see that the headway constraints are only affected by the information when two consecutive trains depart, and the station constraint by the information how many trains wait inside a station. Therefore, it seems to be clear that we have to regard the station arcs in a configuration procedure. Including

such an arc $a = (u, v)$ for a train $c$, one has all the necessary information given: first when the train departs, namely $t(v)$, and second in which timewindow the train occupies station resources, namely $[t(u), t(v)]$. So, the idea is to find a set of station arcs for each station that holds both requirements.

The catch is that the departure events as constructed in the previous example are not generated multiple times for each outgoing track and each traintype, which is needed to describe headway conditions between trains for each of these tracks. So, in order to handle this problem, we have to change the construction of the train routing graph which, unfortunately, increases the number of needed nodes and arcs of this graph.

In an analogue way to the track configuration models, we introduce now a definition for similar configuration sets consisting of station arcs of a (modified) train routing graph.

**Definition 9 (Station Configuration)** We call a set $A \subseteq A_s$ of station arcs associated with a station $s \in S$ *station configuration*, if

$$t(u_2) \leq t(v_2) \Rightarrow\ t(v_2) \geq t(u_2) + \delta_j(c(u_1, u_2), c(v_1, v_2))$$

for each $(u_1, u_2), (v_1, v_2) \in A$ with $(u_1, u_2) \neq (v_1, v_2)$ and $j(u_2) = j(v_2) =: j$; and

$$|\{(u, v) \in A : t \in [t(u), t(v)]\}| \leq \gamma_s$$

for each time instant $t$.

We denote the set of all station configurations of a station $s \in S$ with $K_s$, and the set of all station configurations with $K$. □

We can now use this definition in order to introduce a model that couples paths in the train routing graph with station configurations like done in the TCP model with track configurations. So, for a given objective function $w : P \to \mathbb{R}$ we can write down the node configuration model (NCP) as follows.

| | | | | |
|---|---|---|---|---|
| (NCP) | max | $w^T x$ | | |
| | s.t. | $x(P_i) \leq 1$ | $\forall\, i \in I$ | (i) |
| | | $z(K_s) \leq 1$ | $\forall\, s \in S$ | (ii) |
| | | $x(P_a) - z(K_a) \leq 0$ | $\forall\, s \in S,\ \forall a \in A_s$ | (iii) |
| | | $x \in \{0,1\}^P$ | | |
| | | $z \in \{0,1\}^K$ | | |

The model consists of two types of binary variables $x$ and $z$. The $x$ variables correspond to $\sigma_i$-$\tau_i$-paths in the appropriate individual train routing digraph of the request $i \in I$ and the $z$ variables correspond to the station configurations of the regarded instance. A path variable $x_p$ takes the value one if the corresponding path $p$ is contained in the final schedule and zero else. A configuration variable $z_k$ takes the value one if the corresponding configuration $k$ is active in the final schedule and zero else.

The path constraints (i) make sure that at most one train is scheduled for each request $i \in I$. The configuration constraints (ii) make sure that at most one station configuration is active for each station $s \in S$. The coupling constraints (iii) ensure that scheduled trains use only station arcs that are contained in active station configurations.

**Remark 4 (Station Graph)** We should remark that the implementation of a graph with splitted departure events as mentioned above would have needed deep changes in `Ts-Opt`. So, we decided to encapsule this by creating a graph which we call *station graph* that consists of station arcs and copies of arrival and departure events that symbolize this subdivision.

In order to reduce the problem size, we also aggregate parallel arcs of this graph and store a list of these arcs for each station arc that represents them. □

**Example 3 (Station Configuration)** Remember the train routing graph introduced in Example 2. As we see, there are many ways to route the blue and the red train through the routing graph. The tracks are labeled with departure-headway conditions which have to be regarded in a final schedule. Luckily, we are here not given any station capacity constraints, such that the headway condition for departure events is the only restriction. So, we have to find valid station configurations for each station with departure events, that is, station $a$ and station $b$ in this example. An exemplary subset of station configurations of station $b$ is displayed in Figure 5.1.

As a result of the given requests, a configuration is prefered that allows a blue train to arrive and departe immediately at $t = 4$ in $b$ and a red train to arrive also at moment $t = 4$ and departs any moment. Such a configuration would result in a profit of 10 for both of the trains. As we see, the configuration displayed in Figure 5.1d fulfills this requirement and can be active in a final solution of NCP. So, if we assume that this configuration is contained in an optimal solution, a possible return could look like in Figure 5.2.

□

Before we analyze the pricing problem we regard first the dual LP D-NCP of the LP relaxation of the NCP. For this purpose, let $\gamma_i$, $\pi_s$ and $\lambda_a$ be the dual variables to the path constraint for the request $i \in I$, the station constraint of the station $s \in S$ and

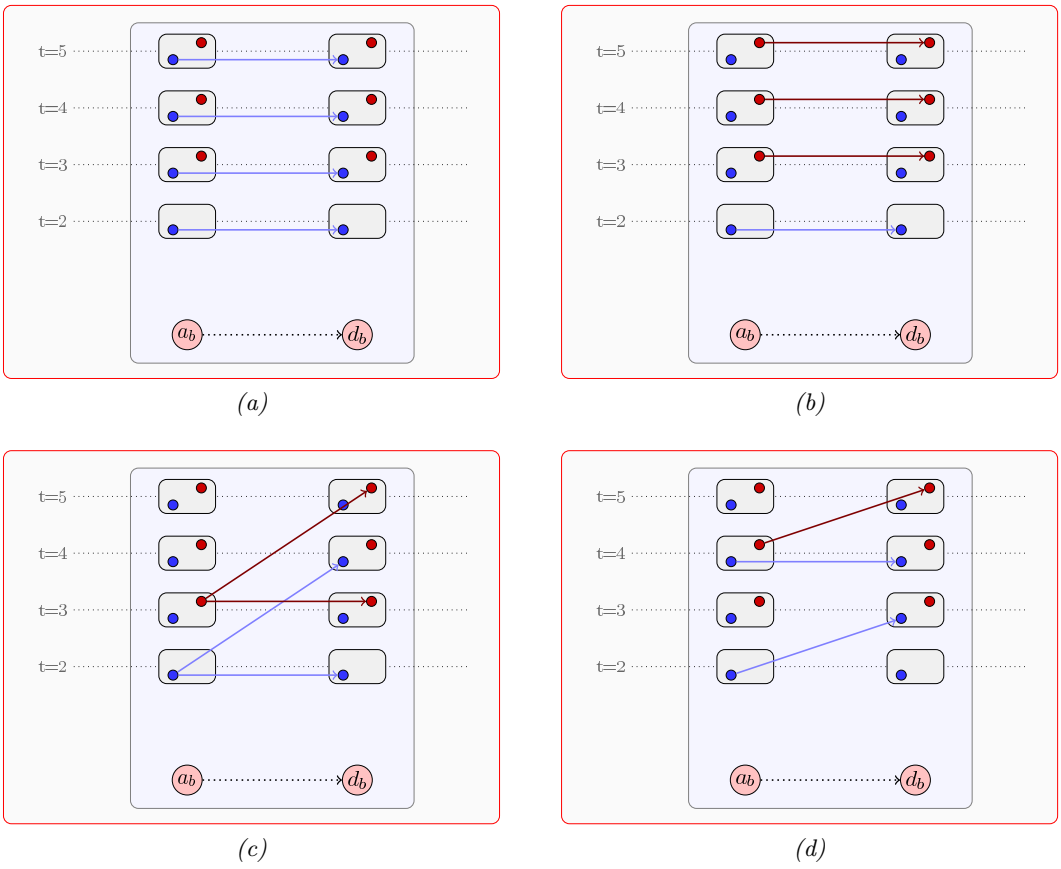



*Figure 5.1:* Some inclusion maximal station configurations for station $b$ of Example 2.

the coupling constraint of the station arc $a \in A$. We can now write down D-NCP as follows.

$$
\begin{aligned}
\text{(D-NCP)} \qquad \min \quad & \mathbb{1}^T\gamma + \mathbb{1}^T\pi & & \\
\text{s.t.} \quad & \gamma_i + \sum_{a \in p} \lambda_a \geq w_p & \forall\, i \in I,\ \forall p \in P_i & \quad \text{(i)} \\
& \pi_s - \sum_{a \in k} \lambda_a \geq 0 & \forall\, s \in S,\ \forall k \in K_s & \quad \text{(ii)} \\
& \gamma, \pi, \lambda \geq 0 & &
\end{aligned}
$$

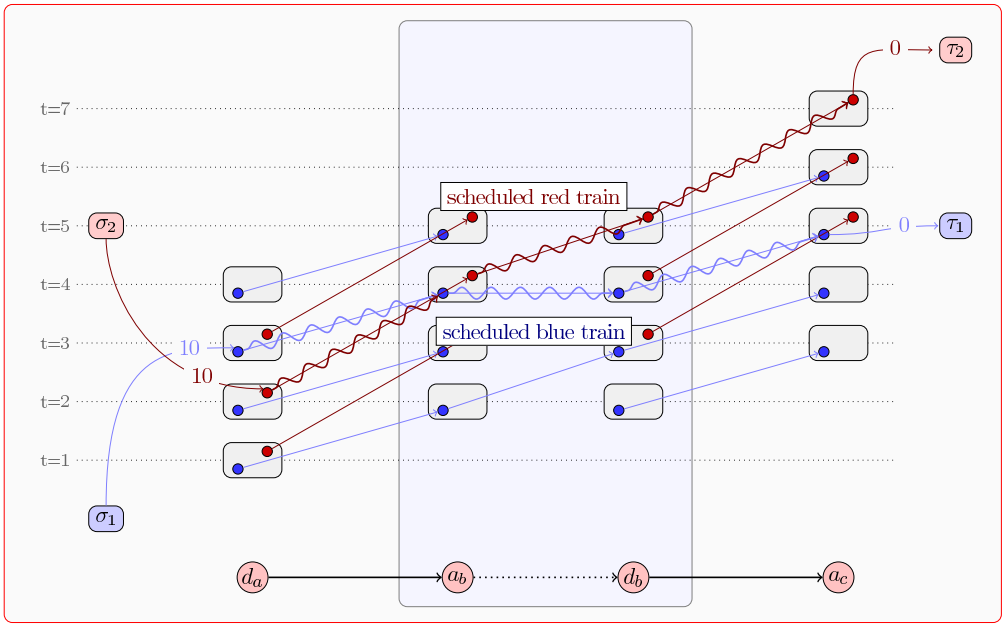



*Figure 5.2:* Possible station-configuration with two planned trains.

## 5.2 Pricing Path Variables

It follows that the pricing problem of the path-variables is equivalent to the question to find a bundle and path that violates a constraint of the D-NCP, i.e., we search a bundle $i \in I$ and a path $p \in P_i$ with

$$\gamma_i + \sum_{a \in p} \lambda_a < w_p.$$

As mentioned before, we use a path-based objective function $w_p$ that can be transformed into an arc-based $w_a$ function with the property $w_p = \sum_{a \in p} w_a$. Therefore, we can reformulate the pricing problem into the question to find a request $i \in I$ and a path $p \in P_i$ with

$$\sum_{a \in p} (w_a - \lambda_a) > \gamma_i.$$

Consider the following weight function $\widetilde{w}$ for each arc.

$$\widetilde{w}_a := \begin{cases} w_a - \lambda_a, & a \text{ station arc}, \\ w_a, & \text{else} \end{cases}$$

Using this function as arc weight, it is sufficient to find a longest $\sigma_i$-$\tau_i$-path in each individual train routing graph $D_i$. So, if the weight of such a path exceeds $\gamma_i$ we found a path that has the ability to improve the objective of the restricted master problem and should therefore be added to the pricing pool of our problem.

Since we know that the individual train routing graphs are acyclic (remember Remark 2), it is possible to find such a path in polynomial time, using for example the Bellman-Ford algorithm or with a dynamic program that uses a topological sort of the arcs.

Since we didn't change the train routing graphs, we needed to adapt this algorithms for our implementations (see Remark 4), which led to the following modified Algorithm 4 of Bellman-Ford.

**Algorithm 4**: Modified Bellman-Ford

**Input**:

- Individual train routing graph $D_i = (V_i, A_i, \sigma_i, \tau_i)$ for a request $i \in I$
- Weights $w_a$ for each arc $a \in A_i$
- Dual costs $\lambda_a$ for each copy $a$ of a station arc arc $\widetilde{a} \in A$

**Output**: Distance and predecessor labels for a longest path with respect to the weight function $\widetilde{w}$ as defined above.

**begin**

```
// Initialize node labels
foreach v ∈ V_i do
    if v is source then
        v.distance ← 0;
    else
        v.distance ← −∞;
    v.predecessor ← NIL;

// Main Loop
for k = 1, ..., |V_i| − 1 do
    foreach station arc a_1 = (u_1, v_1) ∈ A_i do
        foreach station arc copy a_2 = (u_2, v_2) of a_1 do
            if u_1.distance + w_{a_2} − λ_{a_2} > v_1.distance then
                v_1.distance ← u_1.distance + w_{a_2} − λ_{a_2};
                v_1.predecessor ← u_1;
                a_3 = (u_3, v_3) ← transit arc associated with the departure event v_2;
                if u_3.distance + w_{a_3} > v_3.distance then
                    v_3.distance ← u_3.distance + w_{a_3};
                    v_3.predecessor ← u_3;
```

**end**

## 5.3 Pricing Configuration Variables

The pricing problem for the configuration variables is the question of finding a station configuration that violates the appropriate constraint of the D-NCP, i.e., we search a station $s \in S$ and a station configuration $k \in K_s$ with

$$\pi_s - \sum_{a \in k} \lambda_a < 0$$

or equivalent

$$\sum_{a \in k} \lambda_a > \pi_s.$$

The task is now to find an algorithm to solve this question as fast as possible. Our first idea was to find a dynamic program that is capable for this problem. The second one was the implementation of an exact IP-based method that is also capable of handling information obtained in the branch-and-bound routine.

Before explaining these methods, we want to formulate the following remark which leads to an interesting conclusion.

**Remark 5 (Configuration pricing as RCSP)** Let $N = (S, J, C)$ be a infrastructure graph, $D = (V, A)$ be a train routing graph and $s \in S$ be a station.

Then, the pricing problem for the configuration variables (of station $s$) can be modeled as *Resource Constrained Shortest Path Problem* (RCSP) with the following graph construction. Let $W_s$ be the set of arrival events in $s$, $U_s$ be the set of departure events in $s$, and $A_s^j$ be the set of station arcs in $s$ with departure events associated with a track $j \in \delta^+(s)$.

We multiply now each arrival event $w \in W_s$ $|\delta^+(w)|$ times and connect each departure event $u$ that is adjacent to $w$ with exactly one of these copies, and label the so created new arcs with the original arc $(w, u)$. As we see, each arrival event and each station arc in this construction is connected with exactly one departure event who again can be associated with exactly one track. So, we can partition this graph in the following sets for each track $j \in \delta^+(s)$

- $\widetilde{U}_s^j$ containing the the departure events associated with track $j$,
- $\widetilde{W}_s^j$ containing the arrival event copies associated with a departure event to track $j$, and
- $\widetilde{A}_s^j$ containing the station arc copies associated with a departure event to track $j$.

Considering these sets, we can answer the question if a headway condition between a departure event $u \in \widetilde{U}_s^j$ and a departure event that is associated with an arrival event $w \in \widetilde{W}_s^j$ (which we shorten with $u(w)$) is violated or not. So, we can define the set

$$\widehat{A}_s^j := \{(u, w) \in \widetilde{U}_s^j \times \widetilde{W}_s^j : \; t(u) + \delta_j(c(u), c(u(w))) \leq t(u(w))\}.$$

We define a graph station-track routing graph $D_s^j$ as a graph with nodes $V_s^j := \widetilde{U}_s^j \cup \widetilde{W}_s^j \cup \{\sigma_s^j, \tau_s^j\}$ and arcs $\overline{A}_s^j := \widetilde{A}_s^j \cup \widehat{A}_s^j \cup (\{\sigma_s^j\} \times V_s^j) \cup (V_s^j \times \{\tau_s^j\})$.

Let the outgoing tracks $\delta^+(s)$ be given in an arbitrary order, lets say $\delta^+(s) = \{1, \ldots, k\}$. We continue the construction by concatenating the target node $\tau_s^j$ of the graph $D_s^j$ with the source node $\sigma_s^{j+1}$ of the graph $D_s^{j+1}$ $(j = 1, \ldots, k-1)$. We denote the result of this step as *station routing graph* and write shortly $D_s = (V_s, A_s)$.

The major property of the acyclic graphs $D_s^j$ is the fact that, due to the contruction, two consecutive departure events on a $\sigma_s^j$-$\tau_s^j$ path can't violate any headway constraint. This property is transfered to the $D_s$ regarding $\sigma_s^1$-$\tau_s^k$ paths. Note again, that we expect that the headway matrices hold the triangle inequality (see also Section 3.2) such that also no transitive violation of headway constraints can occur.

Given these constructions, we can now formulate the pricing problem for the station configuration variables as RCSP by introducing for each time instant $t = 1, \ldots, T$ (with timehorizon $T$) a station capacity resource $r_t$. Given this, we can allocate to each station arc copy $(v_1, v_2) \in \widetilde{A}_s := \bigcup_j \widetilde{A}_s^j$ the consumed resources $r_t$ $(t \in [t(v_1), t(v_2)])$ and allocating a weight of $\lambda_a$, with $a$ denoting the original station arc of $(v_1, v_2)$. Each solution of this RCSP delivers implicitly a set of station arcs that describes a station configuration.□

Given this remark, we can directly conclude the following theorem.

**Theorem 1** *Disregarding station capacities, the pricing problem (for the configuration variables) can be processed in polynomial time.* □

For the next subsections, we need the following assumption.

**Assumption** In the following subsections, we assume that the departure nodes $U^s$ of a station $s$ are given in an order $v_1, \ldots, v_n$ such that

$$\forall\, i, j \in \{1, \ldots, n\} :\ i < j \Rightarrow t(v_i) \leq t(v_j),$$

that we have a timehorizon of $T$ and that we are given a station capacity of $\gamma_s$ for $s$.

### 5.3.1 A Dynamic Program

As described in Section 1.3, the main idea of dynamic programing is to find a possibility of dividing the whole problem into homogeneous subproblems in order to find an optimal solution.

Before doing so, we first introduce the following vectors for a station $s \in S$.

**Definition 10** We define the *capacity vector* $\gamma^a \in \{0,1\}^T$ for each arc $a = (u,v) \in A_s$, that represents the capacity-usage of $a$, as a vector with entries

$$\gamma_t^a = \begin{cases} 1, & \text{if } t \in [\, t(u), t(v)\,], \\ 0, & \text{else.} \end{cases}$$

We define the *headway vector* $\zeta^v \in \{0,1\}^{U^s}$ for each departure event $v \in U^s$, that stands for the incidence vector of nodes that are blocked through the headway constraints bounded with the usage of $v$, as a vector with following entries

$$\zeta_u^v = \begin{cases} 1, & \text{if } j(v) = j(u) \text{ and } \delta_{j(v)}(c(u), c(v)) + t(u) > t(v), \\ 0, & \text{otherwise.} \end{cases}$$

□

We can now introduce the Bellman equation $z$, that delivers for $z_T(0,0)$ the optimal solution value of the pricing problem in the root-node of the branch-and-bound tree.

$$z_i(\gamma,\zeta) = \begin{cases} 0, & \text{if } i = 0, \\ z_{i-1}(\gamma,\zeta), & \text{if } i > 0 \text{ and } \zeta_{v_i} = 1, \\ z_{i-1}(\gamma,\zeta), & \text{if } i > 0 \text{ and } \gamma_{t(v_i)} = \gamma_s, \\ \max\{\{z_{i-1}(\gamma,\zeta)\} \cup \bigcup\limits_{a \in \delta^-(v_i)} \{z_{i-1}(\gamma + \gamma^a, \zeta + \zeta^{v_{i-1}}) + w_a\}\}, & \text{else.} \end{cases}$$

with $\gamma \in \mathbb{Z}^T$ and $\zeta \in \mathbb{Z}^n$.

The main problem of this approach is the fact that the above recursion leads to $O(\gamma_s^T \cdot 2^n)$ states, which is even for relatively small instances a huge number.

For the case that we are only given one outgoing track $j$ and one traintype $c$, one can reduce this value by discarding the vector $\zeta$ and using instead the information that the departure events are sorted in the recursion. So, the recursion can be read as

$$z_i(\gamma) = \begin{cases} 0, & \text{if } i = 0, \\ z_{i-1}(\gamma), & \text{if } i > 0 \text{ and } \gamma_{t(v_i)} = \gamma_s, \\ \max\{\{z_{i-1}(\gamma)\} \cup \bigcup\limits_{a \in \delta^-(v_i)} \{z_{v_{i-\delta_j(c,c)}}(\gamma + \gamma^a) + w_a\}\}, & \text{else.} \end{cases}$$

This recursion leads to a worst case number of $O(\gamma_s^T)$ states, which is also exorbitant. This is the main reason, why we have not implemented a dynamic program based on this recursions. We use instead an IP-based method which also uses information won of the branch-and-bound tree, and is therefore more appropriate for a branch-and-price routine.

### 5.3.2 An Exact Method

After the disappointment of the last subsection, we introduce here an exact method to price the configuration variables of the NCP.

As mentioned before, this method is based on IP formulations $z$-PRICE($s$) whose solution deliver each an optimal station configuration for the appropriate station $s \in S$. Each formulation can be regarded as multi-dimensional knapsack problem with additional constraints that rule out configurations who cannot be in an optimal solution of the current branch-and-bound node.

Before we describe the model in detail, we need some definitions with the purpose of shorten the notation.

We denote the set of station arcs that consume station resources of $s \in S$ at the moment $t = 1, \ldots, T$ with $A_s^t$, that is,

$$A_s^t := \{(u, v) \in A_s : t \in [t(u), t(v)]\}.$$

The set of station arcs that are in headway conflict with a departure event associated with the station arc $(u, v) = a \in A_s$, is denoted with $A_s^a$, more precisely,

$$\begin{aligned} A_s^a := \{(\overline{u}, \overline{v}) \in A_s : \quad & j(\overline{v}) = j(v), \\ & t(\overline{v}) \leq t(v), \\ & t(v) - t(\overline{v}) \leq \delta(c(\overline{v}), c(v))\} \end{aligned}$$

Note that especially the arc $a$ is also contained in the set $A_s^a$.

Assume that we are given an branch-and-bound node with some fixed variables, we denote the configurations that are fixed to zero with $K_s^0$ and the paths that are fixed to one with $P^1$.

We can now introduce the mentioned integer program $z$-PRICE($s$) for a station $s \in S$ and an appropriate node of the branch-and-bound routine as follows.

$$\begin{aligned} (z\text{-PRICE}(s)) \quad \max \quad & \lambda^T \overline{z} & & \\ & \overline{z}(A_s^t) \leq \gamma_s & \forall\, t = 1, \ldots, T & \quad \text{(i)} \\ & \overline{z}(A_s^a) \leq 1 & \forall\, a \in A_s & \quad \text{(ii)} \\ & \overline{z}_a = 1 & \forall\, p \in P^1\ \forall a \in A_s, a \in p & \quad \text{(iii)} \\ & \overline{z}(k) - \overline{z}(A_s \setminus k) \leq |k| - 1 & \forall\, k \in K_s^0 & \quad \text{(iv)} \\ & \overline{z} \in \{0,1\}^{A_s} & & \end{aligned}$$

The binary variables $\overline{z}$ of this program represent station arcs of the appropriate station $s \in S$. An arc variable $\overline{z}_a$ takes the value one if the arc $a$ is contained in the final configuration, and zero otherwise.

The constraints (i) represent the station capacity of station $s$ and forbid that too many trains wait simultaneously inside a station at each time. The inequalities (ii) forbid that two trains leaving the station $s$ on the same track, violate the headway constraint on that track. The equalities (iii) guarantee that the final configuration can route paths that are fixed to one in the current branch-and-bound node. The conditions (iv) rule configurations out that are fixed to zero and must not contained in the current node. They can equivalently be rewritten as

$$\sum_{a \in k} (1 - x_a) + \sum_{a \notin k} x_a \leq 1.$$

The main disadvantage of this approach is the fact that the number of configurations that were generated in an iteration of the column generation is in general comparatively small. For all instances in our testset, this method works well but in future work it seems to be unavoidable to find an algorithmic (maybe an heuristic method) solving methodology for this problem.

## 5.4 Polyhedral Analysis

After the introduction of the NCP model and his pricing routines, we want to have a look on the polyhydral aspect of this model. In this section we compare the station model of the previous chapter with the other configuration models introduced in section 4.2.

Due to the construction of the configurations, it is clear that the objective values of NCP and TCP are the same, i.e., $\nu(\text{NCP}) = \nu(\text{TCP})$, but how about the LP relaxations? One of the main questions is, which of the models TCP or NCP has a tighter LP-bound. Before we can analyze this, consider the following lemma.

**Lemma 1** *There is a projection*

$$\pi : \mathbb{R}^{P \times K} \rightarrow \mathbb{R}^{P \times Q}$$

*such that*

$$\pi(P_{LP}(NCP)) \subseteq P_{LP}(TCP).$$

Proof Let $k \in K_s$ be a station-configuration of a station $s \in S$. Note that each station arc in $k$ is associated with exactly one departure event which can be allocated to exactly one passing arc. Denote this allocation of station arcs to track arcs with $\phi'$.

Since no headway condition can be violated, it is easy to see that $\phi(k,j) := \phi'(k) \cap A_j$ are valid—possibly empty—track-configurations for each track $j \in J$. Note that there is for each track $j$ at most one station configuration $k$ with $\phi(k,j) \neq \emptyset$ that is active in an integral solution of NCP.

Define now $\pi$ as follows.

$$\pi : \mathbb{R}^{P\times K} \to \mathbb{R}^{P\times Q}, (x,z) \mapsto (x,y)$$

with $y_q := \sum_{k\in K:\, \phi(k,j(q))\neq\emptyset} z_k$ for each track configuration $q \in Q$.

Due to the construction of the station configurations, it holds $\pi(x,z) \in P_{\mathrm{LP}}(\mathrm{TCP})$. ■

Using the projection $\pi$, constructed in the above proof, it directly follows the below Theorem 2.

**Theorem 2** *It holds*

$$\nu_{LP}(NCP) \leq \nu_{LP}(TCP)$$

. □

In order to overcome tailing-off effects in column generation, we state now a simple LP-bound for the NCP formulation out.

**Lemma 2** *Let $\gamma, \pi, \lambda \geq 0$ be dual variables for NCP and $\nu_{LP}(NCP)$ the optimum objective value of the LP-relaxation of NCP. Define*

$$\eta_i := \max_{p\in P_i} \sum_{a\in p} (w_a - \lambda_a) - \gamma_i, \qquad \forall i \in I,$$

$$\theta_s := \max_{k\in K_s} \sum_{a\in k} \lambda_a - \pi_s, \qquad \forall s \in S,$$

$$\beta := \sum_{i\in I} \max\{\gamma_i + \eta_i, 0\} + \sum_{s\in S} \max\{\pi_s + \theta_s, 0\}.$$

*Then*

$$\nu_{LP}(NCP) \leq \beta.$$

Proof We can trivially conclude that for each path $p \in P_i$ of the slot request $i \in I$, the following inequality

$$\gamma_i + \eta_i \geq \sum_{a\in p} (w_a - \lambda_a)$$

and therefore, the inequality

$$\gamma_i + \eta_i + \sum_{a \in p} \lambda_a \geq w_p$$

holds.

It is also clear, that

$$\pi_s + \theta_s \geq \sum_{a \in k} \lambda_a$$

and therefore,

$$\pi_s + \theta_s - \sum_{a \in k} \lambda_a \geq 0$$

holds for each station configuration $k \in K_s$ of the station $s \in S$.

Considering now the vectors $\overline{\gamma} := \max\{\gamma + \eta, 0\}$ and $\overline{\pi} := \max\{\pi + \theta, 0\}$ (with a component-wise taken maximum), it is easy to see that $(\overline{\gamma}, \overline{\pi}, \lambda)$ is dual feasible solution of the LP-relaxation of NCP delivering a dual objective $\beta$. ■

# Chapter 6

# Computational Experience

> Not because they are easy, but because they are hard.
>
> *(John Fitzgerald Kennedy)*

In this chapter, we present our computational results for the implementation of the NCP model. First in Section 6.1, we discuss the implementational issues of the NCP model, describe then our test instances in Section 6.2, and present finally our computational results in Section 6.3.

## 6.1 Implementational Issues

As already mentioned in Section 2.4, our implementations took place in the railway optimization suite `Ts-Opt` that was developed at the Zuse-Institute-Berlin (ZIB) and is maintained by Thomas Schlechte. It can be used to solve instances of the Train Timetabling Problem with the possibility of selecting three different kinds of models, APP, ACP and TCP, which can be controlled with dozens of parameters. `Ts-Opt` is written in `C/C++`, consists of about 70,000 lines of code, and needed about 14,000 lines of changes (new and modified lines) for our implementations.[1] In the context of this work, we implemented as fourth model the NCP that is now fully integrated into `Ts-Opt`. We needed mostly the input/output modules and used an already implemented data structure for sparse-matrices[2] in order to store the NCP model space-efficiently.

In order to solve the occurring integer programs, i.e., the main program NCP and the sub programs ($z$-PRICE($s$)) for the pricing problem of the station configuration variables, we used the call-libraries of `SCIP` 1.1.0.11 that is also developed at ZIB and *is currently one of the fastest non-commercial mixed integer programming solver.*[3] We use `SCIP` as

[1] Code statistics are taken from the log files of the software revision control system CVS at July the 29th, 2009.

[2] We also implemented features like a check for redundancy before adding new columns.

[3] See also the producer's homepage `http://scip.zib.de/`.

branch-and-price framework (for the main program NCP) as well as pure MIP solver (for the sub-problems) with CPLEX 11.2.0 as underlying LP-solver.

The implementation of the other models in Ts-Opt use CPLEX 11.2.0 as underlying MIP solver.

## 6.2 Test Instances

For our computational research we employ instances of the TTPlib that is accessible at http://ttplib.zib.de/ and test sets that are generated for the network instance that is generated with NETCAST. The regarded instances use each time units discretized in one-minute steps.

The instances of TTPlib can be classified in two categories. The first category contains 50 instances whose infrastructure parts represent a real-world, long-distance railway network in the German region of Hannover, Kassel and Fulda, shortly denoted as Ha-Ka-Fu (displayed in Figure 6.1c). The infrastrucuture for these instances is given by two macroscopic networks: one with including station capacity information and the other without these. The data of both Ha-Ka-Fu networks consist of information for about 37 stations, 120 tracks, and 6 train types which leads to 4370 headway conditions and 720 driving time information. The appropriate requests of these instances are based on the official timetables of the year 2002 published by the German railway company Deutsche Bahn AG for this region. Since, the NCP model is especially suited for the regard of station constraints, we include only a subset (the first 29) of the Ha-Ka-Fu instances for the network version that contains the station capacity data and denote them shortly with hakafu_xx (with xx from 01 up to 30). Implementing a new model is connected with an huge amount of time related with programming and thus, with a lot of bug-fixing. Unfortunately, we could not fix each bug till the deadline and had to skip the instances hakafu_11, hakafu_12, hakafu_16 and hakafu_20.

The second category of instances is based on an artificial network (denoted with Wheel) with 1 traintype, 15 stations and 42 tracks leading to 42 headway conditions and 42 driving time information (displayed in Figure 6.1a). The corresponding 11 requests of these instances are fictive scenarios which are automatically generated by *Andreas Tanner* an employer of the WIP who is involved to the Trassenbörse project (see also Section 2.4), and are denoted shortly with wheel_xx with xx from 01 up to 11.

A third class of instances is based on a infrastructure network (denoted with SiT and displayed in Figure 6.1b) that is created with NETCAST from data of a real-world train corridor. The infrastructure part consists of 6 traintypes, 18 stations and 40 tracks leading to 948 headway constraints and 148 driving time information. The corresponding

9 requests are slimed real-world scenarios, and denoted with sit_xx with xx from 01 up to 09.

## 6.3 Computational Results

Our computations took place on a computer with a 64 bit architecture with an Intel® Core™2 Extreme CPU X9650 and 3.00 GHz, 8 GB main memory (RAM), and running the Linux-distribution openSUSE 11.1 as operating system. The source code of `Ts-Opt` has been compiled with the GNU C++ compiler (g++) in version 4.3.2 in optimizations mode.

We started our computations with the parameters listed in Table 6.1 together with the default settings of `SCIP` for the main MIP. The MIPs for the configuration pricing are solved with disabling presolving in `SCIP` and setting a gap limit of 2 %. We presolved the configuration MIPs by discarding variables with negative objective which are not expected to be in a final solution.

| | |
|---|---|
| Time-limit (only MIP solving) | four hours (14,400 seconds) |
| Gap-limit | 2 % |
| Maximal number of branch-and-bound nodes | 1,000 |
| Maximal iterations for column generation | 300 |
| Maximal number of columns (in total / paths / configurations) | 50,000 / 32,500 / 17,500 |

*Table 6.1:* MIP settings.

In order to evaluate our results of the NCP model for the given instances, we compare them with results obtained by the ACP model of Borndörfer and Schlechte (2007a) implemented in `Ts-Opt`. The ACP model is statically constructed and contains right from the beginning of the solving all necessary columns and rows. We would have liked to compare the NCP with the TCP model, but in the current stage of development, the TCP is not able to handle the station constraints efficiently. We present our results in three tables.

The first Table 6.3 summarizes the problem sizes of each instance for both models, ACP and NCP. The first two columns of this table contain the name and the number of slotrequests (Req) of the regarded instance. The following seven columns are related to the ACP model and list the important problem sizing information: total number of rows (r) that are also separated as number of coupling constraints (Cpl), number of set

covering constraints (SC), number of set packing constraints (SP)[4], followed by the total number of columns (c) that are devided into the number of path routing variables ($|A|$) and configuration routing variables ($|\overline{A}|$). The last seven columns are related to the NCP model. The columns show the number of processed column generation iterations (CG), number of processed branch-and-bound nodes (n), number of rows (r) and how many of them are coupling constraints (Cpl), the total number of columns (c) at the end of the generation process with splitting into the number of path variables ($|P|$) and the number of station configurations ($|K|$).

The second Table 6.4 presents the obtained results and is, again, splitted into two section for the results of both models. The columns "t (sec)" display the solving time in seconds, the columns $\nu$(ACP) and $\nu$(NCP) give the obtained final objective and the column "trains" shows how many trains are scheduled in the final solution, each for the appropriate model. The last column ($\Delta$) shows the gap between the solution value of the ACP and the NCP model, calculated with the formula 100 * ($\nu$(ACP) - $\nu$(NCP)) / $\nu$(ACP). Since, we expect that the ACP model delivers optimal solutions, one can regard these values also as optimality gap for the (heuristic) solutions of the NCP. All the other columns, show the same information as explained for the Table 6.3.

A third table (Table 6.5) summarizes the information risen up from the solving of the pricing problem of the configuration variables. We list the name of the regarded instances in the first column, specify in the second column how many MIPs have to be solved for this instance and how many column generation iterations have been processed. The next three columns show the minimal, maximal and average solving time (t) in seconds for these MIPs. The following six columns display the minimal, maximal and average problem sizes (rows r and columns c). The last five columns list how many of the optimal solutions are found through the different heuristics that are described in Table 6.2.

| Name (abbr.) | | Description |
|---|---|---|
| LP | | LP value is already integral. |
| oneopt | (k) | 1-opt heuristic which tries to improve setting of single integer variables. |
| rounding | (R) | LP rounding heuristic with infeasibility recovering. |
| shifting | (s) | LP rounding heuristic with infeasibility recovering also using continuous variables. |
| rens | (E) | LNS exploring fractional neighborhood of relaxation's optimum. |

*Table 6.2:* Description of some heuristcs in `SCIP`.

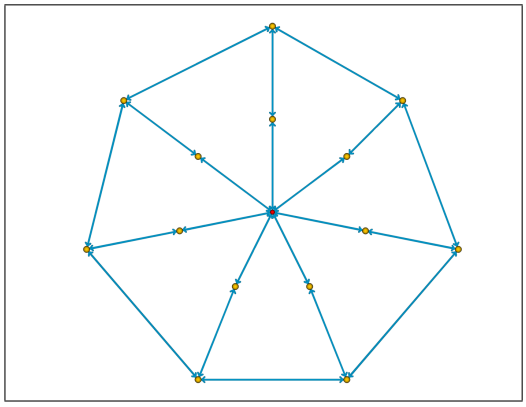
[4]The set covering constraints result from requested minimal waiting times in stations and the set packing constraints result from the station capacity constraints.

## Conclusion

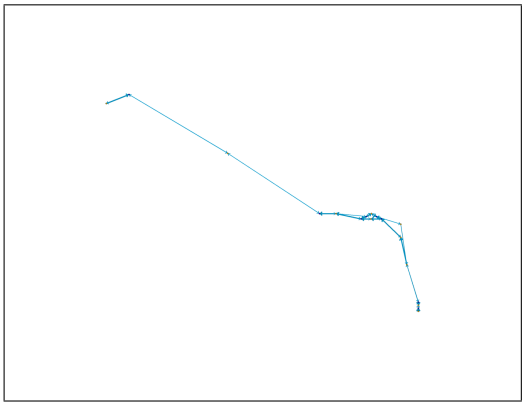
Even if this method is the first to our knowledge that can handle headway times as well as station capacities in a dynamic column generation approach without appending an additional cutting plane stage, it could not lead to the desired improvements. The disappointing results of our calculations show that the approach of the station configuration model for solving the TTP has to be analyzed more deeply, in order to find better methods to unfold the controlling power of this model.

Unfortunately, the results show that the obtained optimization programs of the NCP are harder to solve and deliver worse solutions than the ACP model. Due to the nature of column generation—a column generation approach delivers in general only heuristic solutions—, one could not expect that at least the last problem can be solved as long as a ACP MIP can be created and given to a MIP solver. In our opinion, the first problem is especially rooted in the bad relations between the amount of created path and the one of created configuration variables. In future work, one should examine the effects of heuristically generated station configurations to the solving quality of the NCP.

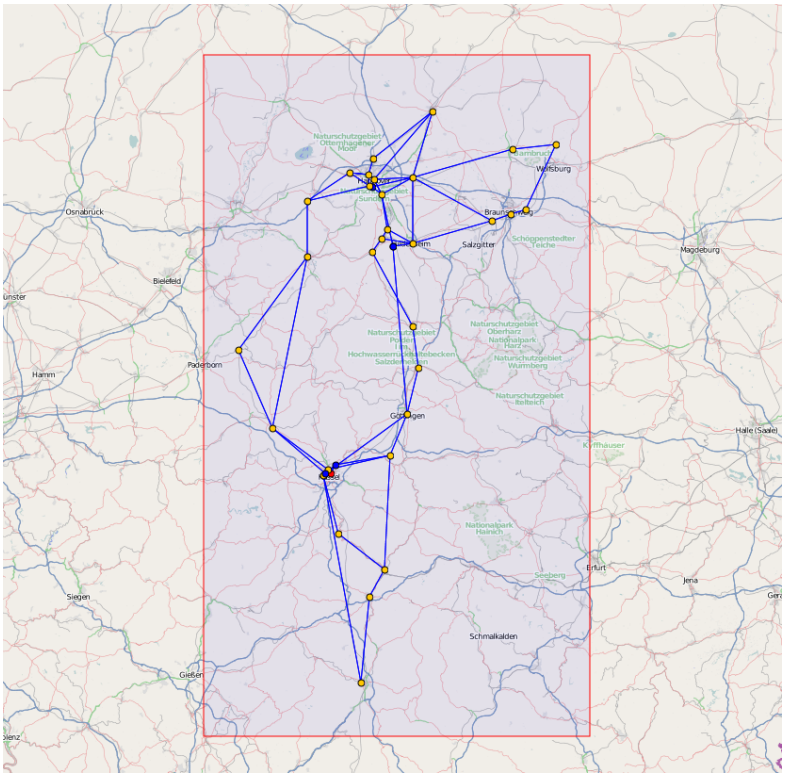
The results show, that most rows of the NCP are related to coupling constraints. A problem without these rows can be trivially solved in polynomial time with a greedy approach. So, in a next stage of development, one should also try to relax these constraints and to solve the result by methods based on the Lagrangian technique.



*(a)* Wheel



*(b)* SiT




*(c)* Ha-Ka-Fu

*Figure 6.1:* Test instances.

| Instance | Req | ACP Model | | | | | | | NCP Model | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | r | Cpl | SC | SP | c | $\|A\|$ | $\|\overline{A}\|$ | CG | n | r | Cpl | c | $\|P\|$ | $\|K\|$ |
| wheel_01 | 8 | 1772 | 241 | 27 | 0 | 1415 | 1032 | 740 | 60 | 1 | 826 | 808 | 872 | 521 | 351 |
| wheel_02 | 11 | 1995 | 254 | 32 | 8 | 1592 | 1214 | 781 | 73 | 1 | 875 | 851 | 1218 | 759 | 459 |
| wheel_03 | 8 | 1731 | 238 | 26 | 0 | 1400 | 992 | 739 | 33 | 1 | 806 | 784 | 663 | 438 | 225 |
| wheel_04 | 19 | 3681 | 497 | 57 | 33 | 2945 | 2166 | 1515 | 248 | 1 | 1683 | 1650 | 2965 | 1457 | 1508 |
| wheel_05 | 15 | 2493 | 308 | 41 | 26 | 1984 | 1550 | 943 | 173 | 1 | 1061 | 1033 | 1957 | 1181 | 776 |
| wheel_06 | 14 | 2843 | 380 | 44 | 7 | 2274 | 1676 | 1167 | 119 | 1 | 1316 | 1288 | 1976 | 1098 | 878 |
| wheel_07 | 46 | 8209 | 939 | 141 | 162 | 6343 | 5364 | 2845 | 300 | 1000 | 3436 | 3375 | 8716 | 6035 | 2681 |
| wheel_08 | 55 | 8995 | 1019 | 156 | 195 | 6985 | 5910 | 3085 | 300 | 1000 | 3787 | 3717 | 9128 | 5937 | 3191 |
| wheel_09 | 106 | 16194 | 1374 | 317 | 519 | 12000 | 12044 | 4150 | 300 | 847 | 5595 | 5474 | 16662 | 13052 | 3610 |
| wheel_10 | 198 | 27862 | 1714 | 597 | 1029 | 19916 | 22692 | 5170 | 300 | 36 | 7628 | 7415 | 26167 | 22119 | 4048 |
| wheel_11 | 288 | 6064 | 860 | 864 | 261 | 6668 | 3456 | 2608 | 300 | 1000 | 1163 | 860 | 2561 | 288 | 2273 |
| hakafu_01 | 285 | 8004 | 1099 | 319 | 198 | 6168 | 2139 | 5865 | 37 | 1 | 1415 | 1105 | 414 | 295 | 119 |
| hakafu_02 | 285 | 62506 | 8204 | 319 | 1463 | 39830 | 19458 | 43048 | 300 | 3 | 22931 | 22621 | 26886 | 19237 | 7649 |
| hakafu_03 | 285 | 100729 | 12890 | 319 | 2920 | 62450 | 32269 | 68460 | 164 | 1 | 51064 | 50754 | 37054 | 32500 | 4554 |
| hakafu_04 | 285 | 240500 | 27289 | 319 | 9706 | 133318 | 74280 | 166220 | 70 | 1 | 177996 | 177686 | 34783 | 32500 | 2283 |
| hakafu_05 | 194 | 5442 | 774 | 229 | 122 | 4378 | 1479 | 3963 | 20 | 1 | 999 | 775 | 303 | 213 | 90 |
| hakafu_06 | 213 | 46862 | 6273 | 234 | 932 | 30106 | 14341 | 32521 | 300 | 75 | 17447 | 17208 | 20003 | 13519 | 6484 |
| hakafu_07 | 184 | 67726 | 8894 | 216 | 1806 | 42635 | 21602 | 46124 | 300 | 31 | 35453 | 35241 | 28664 | 23041 | 5623 |
| hakafu_08 | 199 | 177459 | 20637 | 215 | 5743 | 99240 | 53595 | 123864 | 173 | 1 | 132504 | 132272 | 36739 | 32500 | 4239 |
| hakafu_09 | 114 | 3190 | 468 | 141 | 56 | 2728 | 907 | 2283 | 8 | 1 | 618 | 471 | 178 | 120 | 58 |
| hakafu_10 | 109 | 25111 | 3572 | 129 | 564 | 17022 | 7958 | 17153 | 300 | 3 | 9704 | 9562 | 9340 | 5959 | 3381 |
| hakafu_13 | 28 | 814 | 134 | 35 | 25 | 841 | 255 | 559 | 6 | 1 | 195 | 134 | 62 | 31 | 31 |
| hakafu_14 | 33 | 6301 | 1033 | 27 | 168 | 4956 | 2207 | 4094 | 56 | 1 | 2851 | 2787 | 2181 | 1529 | 652 |
| hakafu_15 | 31 | 11973 | 1879 | 40 | 187 | 8771 | 4255 | 7718 | 73 | 1 | 7387 | 7322 | 3842 | 3197 | 645 |
| hakafu_17 | 285 | 7154 | 986 | 0 | 171 | 5261 | 1874 | 5280 | 45 | 1 | 1319 | 1006 | 505 | 314 | 191 |
| hakafu_18 | 285 | 59058 | 7605 | 0 | 1560 | 36551 | 17296 | 41762 | 300 | 43 | 20480 | 20167 | 25369 | 18714 | 6655 |
| hakafu_19 | 285 | 97488 | 12249 | 0 | 3004 | 58600 | 28410 | 69078 | 210 | 1 | 45054 | 44741 | 37326 | 32500 | 4826 |
| hakafu_21 | 209 | 4934 | 707 | 0 | 136 | 3834 | 1343 | 3591 | 17 | 1 | 957 | 719 | 324 | 223 | 101 |
| hakafu_22 | 212 | 43014 | 5594 | 0 | 1019 | 26851 | 12604 | 30410 | 300 | 147 | 14928 | 14686 | 18208 | 12913 | 5295 |
| hakafu_23 | 199 | 74027 | 9377 | 0 | 2105 | 44363 | 21136 | 52891 | 300 | 22 | 34235 | 34002 | 32121 | 26702 | 5419 |
| hakafu_24 | 194 | 169257 | 19946 | 0 | 6399 | 93720 | 46038 | 123219 | 222 | 1 | 118777 | 118546 | 36813 | 32500 | 4313 |
| hakafu_25 | 117 | 2929 | 421 | 0 | 65 | 2332 | 786 | 2143 | 9 | 1 | 578 | 429 | 186 | 127 | 59 |
| hakafu_26 | 118 | 21917 | 3040 | 0 | 441 | 14666 | 6873 | 15044 | 300 | 1000 | 8172 | 8022 | 9833 | 6598 | 3235 |

*continued on next page ...*

*Table 6.3:* Problem sizes of the regarded instances.

| **Instance** | **Req** | **ACP Model** | | | | | | | **NCP Model** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | r | Cpl | SC | SP | c | $\|A\|$ | $\|\overline{A}\|$ | CG | n | r | Cpl | c | $\|P\|$ | $\|K\|$ |
| hakafu_27 | 118 | 38304 | 5031 | 0 | 1116 | 23849 | 11179 | 27125 | 300 | 1000 | 18032 | 17882 | 13812 | 11144 | 2668 |
| hakafu_28 | 102 | 83780 | 10621 | 0 | 2484 | 49389 | 23661 | 60119 | 300 | 1000 | 62787 | 62648 | 35597 | 32500 | 3097 |
| hakafu_29 | 20 | 435 | 69 | 0 | 17 | 447 | 127 | 308 | 7 | 1 | 118 | 69 | 44 | 22 | 22 |
| sit_01 | 106 | 16021 | 2085 | 24 | 24 | 8858 | 2532 | 13489 | 300 | 1000 | 2842 | 2745 | 4566 | 613 | 3953 |
| sit_02 | 227 | 24532 | 2881 | 27 | 3 | 12048 | 3246 | 21286 | 300 | 16 | 4009 | 3836 | 4527 | 667 | 3860 |
| sit_03 | 187 | 24250 | 2799 | 18 | 1 | 11662 | 3104 | 21146 | 300 | 41 | 3788 | 3627 | 4902 | 555 | 4347 |
| sit_04 | 67 | 15980 | 2141 | 8 | 36 | 8968 | 2475 | 13505 | 300 | 1000 | 3020 | 2942 | 5079 | 799 | 4280 |
| sit_05 | 86 | 16576 | 2194 | 8 | 40 | 9223 | 2563 | 14013 | 300 | 602 | 3448 | 3358 | 5333 | 1148 | 4185 |
| sit_06 | 30 | 6743 | 988 | 4 | 15 | 4149 | 1133 | 5610 | 300 | 1000 | 1388 | 1344 | 3708 | 286 | 3422 |
| sit_07 | 193 | 40132 | 5298 | 27 | 75 | 22178 | 6216 | 33916 | 206 | 1 | 8265 | 8086 | 5827 | 2314 | 3513 |
| sit_08 | 111 | 17176 | 2084 | 0 | 1 | 8668 | 2282 | 14894 | 300 | 42 | 3611 | 3482 | 4994 | 1277 | 3717 |
| sit_09 | 207 | 36729 | 4424 | 0 | 2 | 18292 | 4802 | 31927 | 262 | 1 | 6719 | 6494 | 5448 | 1736 | 3712 |

| **Instance** | **Req** | **ACP Model** | | | **NCP Model** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $t$ (sec) | $\nu$(ACP) | trains | CG | n | $t$ (sec) | $\nu$(NCP) | trains | $\Delta$ (in %) |
| wheel_01 | 8 | 0.13 | 3500.00 | 8 | 60 | 1 | 7.08 | 3426.08 | 8 | 2.11 |
| wheel_02 | 11 | 0.10 | 4221.02 | 11 | 73 | 1 | 10.38 | 4158.98 | 11 | 1.47 |
| wheel_03 | 8 | 0.10 | 3400.00 | 8 | 33 | 1 | 4.29 | 3322.55 | 8 | 2.28 |
| wheel_04 | 19 | 0.18 | 7526.34 | 19 | 248 | 1 | 68.70 | 7464.43 | 19 | 0.82 |
| wheel_05 | 15 | 0.16 | 5456.78 | 15 | 173 | 1 | 33.04 | 5453.41 | 15 | 0.06 |
| wheel_06 | 14 | 0.16 | 5787.24 | 14 | 119 | 1 | 23.69 | 5742.25 | 14 | 0.78 |
| wheel_07 | 46 | 0.61 | 16889.40 | 42 | 300 | 1000 | 4384.42 | 9300.00 | 24 | 44.94 |
| wheel_08 | 55 | 1.16 | 17944.74 | 46 | 300 | 1000 | 4956.13 | 8201.56 | 21 | 54.3 |
| wheel_09 | 106 | 4.51 | 25935.34 | 63 | 300 | 847 | 14441.72 | 9896.85 | 25 | 61.84 |
| wheel_10 | 198 | 3.01 | 31118.27 | 74 | 300 | 36 | 14457.96 | 8052.24 | 21 | 74.12 |
| wheel_11 | 288 | 0.74 | 26300.00 | 62 | 300 | 1000 | 446.87 | 22200.00 | 54 | 15.59 |
| hakafu_01 | 285 | 3.96 | 1046520.00 | 153 | 37 | 1 | 37.28 | 889250.00 | 140 | 15.03 |
| hakafu_02 | 285 | 11.40 | 1271618.10 | 214 | 300 | 3 | 14550.65 | 659392.72 | 123 | 48.15 |
| hakafu_03 | 285 | 22.30 | 1303995.79 | 217 | 164 | 1 | 14594.99 | 238232.90 | 64 | 81.73 |
| hakafu_04 | 285 | 116.37 | 1344014.58 | 223 | 70 | 1 | 14740.36 | 208982.94 | 58 | 84.45 |
| hakafu_05 | 194 | 3.21 | 771820.00 | 121 | 20 | 999 | 26.31 | 672950.00 | 112 | 12.81 |
| hakafu_06 | 213 | 6.56 | 972842.94 | 163 | 300 | 75 | 14504.83 | 639918.85 | 110 | 34.22 |
| hakafu_07 | 184 | 24.50 | 806103.53 | 135 | 300 | 31 | 14549.65 | 530176.93 | 97 | 34.23 |
| hakafu_08 | 199 | 42.82 | 1012391.67 | 160 | 173 | 1 | 14698.39 | 165263.03 | 45 | 83.68 |
| hakafu_09 | 114 | 2.46 | 584930.00 | 76 | 8 | 1 | 18.96 | 523020.00 | 73 | 10.58 |
| hakafu_10 | 109 | 2.75 | 494819.56 | 86 | 300 | 3 | 742.87 | 400680.42 | 77 | 19.02 |
| hakafu_13 | 28 | 0.90 | 113710.00 | 20 | 6 | 1 | 5.16 | 113710.00 | 20 | 0.00 |
| hakafu_14 | 33 | 1.01 | 181190.00 | 24 | 56 | 1 | 43.23 | 175904.30 | 23 | 2.92 |
| hakafu_15 | 31 | 0.87 | 123530.00 | 23 | 73 | 1 | 73.99 | 103178.76 | 21 | 16.47 |
| hakafu_17 | 285 | 4.67 | 1298910.00 | 181 | 45 | 1 | 50.92 | 1172920.00 | 171 | 9.7 |
| hakafu_18 | 285 | 11.03 | 1438922.94 | 218 | 300 | 43 | 15464.65 | 921086.08 | 145 | 35.99 |
| hakafu_19 | 285 | 18.25 | 1499800.70 | 223 | 210 | 1 | 15059.72 | 259259.39 | 78 | 82.71 |
| hakafu_21 | 209 | 2.47 | 918550.00 | 139 | 17 | 1 | 17.1100 | 837050.00 | 132 | 8.87 |
| hakafu_22 | 212 | 7.71 | 1020845.88 | 168 | 300 | 147 | 14521.88 | 817499.43 | 133 | 19.92 |
| hakafu_23 | 199 | 14.22 | 1141208.38 | 157 | 300 | 22 | 14571.90 | 845572.06 | 122 | 25.91 |
| hakafu_24 | 194 | 39.59 | 1133136.75 | 158 | 222 | 1 | 14700.44 | 185998.47 | 51 | 83.59 |
| hakafu_25 | 117 | 2.16 | 615050.00 | 82 | 9 | 1 | 14.79 | 568510.00 | 80 | 7.57 |
| hakafu_26 | 118 | 2.59 | 652444.64 | 97 | 300 | 1000 | 3495.53 | 537525.00 | 87 | 17.61 |
| hakafu_27 | 118 | 3.88 | 531879.95 | 88 | 300 | 1000 | 3693.88 | 526278.89 | 88 | 1.05 |

*continued on next page ...*

*Table 6.4:* Compuational results for the regarded instances.

| **Instance** | **Req** | **ACP Model** | | | **NCP Model** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $t$ (sec) | $\nu$(ACP) | trains | CG | n | $t$ (sec) | $\nu$(NCP) | trains | $\Delta$ (in %) |
| hakafu_28 | 102 | 10.30 | 623992.36 | 84 | 300 | 1000 | 12146.99 | 427788.13 | 79 | 31.44 |
| hakafu_29 | 20 | 0.48 | 124670.00 | 14 | 7 | 1 | 5.89 | 121030.00 | 13 | 2.92 |
| sit_01 | 106 | 0.99 | 108008.00 | 59 | 300 | 1000 | 9940.86 | 89968.00 | 54 | 16.70 |
| sit_02 | 227 | 1.98 | 715470.00 | 117 | 300 | 16 | 14427.45 | 561430.00 | 98 | 21.53 |
| sit_03 | 187 | 1.81 | 172550.00 | 117 | 300 | 41 | 14452.71 | 157260.00 | 105 | 8.86 |
| sit_04 | 67 | 0.72 | 58500.00 | 41 | 300 | 1000 | 9574.63 | 31930.00 | 19 | 45.42 |
| sit_05 | 86 | 1.03 | 69230.00 | 52 | 300 | 602 | 14435.34 | 26200.00 | 16 | 62.16 |
| sit_06 | 30 | 0.25 | 31300.00 | 17 | 300 | 1000 | 1176.79 | 26330.00 | 13 | 15.88 |
| sit_07 | 193 | 2.46 | 183880.00 | 107 | 206 | 1 | 14430.75 | 83690.00 | 64 | 54.49 |
| sit_08 | 111 | 1.29 | 557940.00 | 71 | 300 | 42 | 14419.64 | 454810.00 | 65 | 18.48 |
| sit_09 | 207 | 7.37 | 628910.00 | 108 | 262 | 1 | 14464.44 | 431760.00 | 125 | 31.35 |

| Instance | MIPs | Iter | Solving Times | | | Problem size | | | | | | Solution found through ... | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $t^{\min}$ (sec) | $t^{\max}$ (sec) | $t^{\text{avg}}$ (sec) | $c^{\min}$ | $r^{\min}$ | $c^{\max}$ | $r^{\max}$ | $c^{\text{avg}}$ | $r^{\text{avg}}$ | LP | k | R | s | E |
| wheel_01 | 418 | 60 | 0.00 | 0.02 | 0.01 | 7 | 28 | 223 | 382 | 105 | 208 | 418 | 0 | 0 | 0 | 0 |
| wheel_02 | 535 | 73 | 0.00 | 0.02 | 0.01 | 7 | 28 | 265 | 520 | 100 | 226 | 533 | 2 | 0 | 0 | 0 |
| wheel_03 | 229 | 33 | 0.00 | 0.02 | 0.01 | 7 | 31 | 224 | 497 | 84 | 248 | 229 | 0 | 0 | 0 | 0 |
| wheel_04 | 1906 | 248 | 0.00 | 0.02 | 0.01 | 7 | 28 | 514 | 754 | 194 | 384 | 1903 | 3 | 0 | 0 | 0 |
| wheel_05 | 2044 | 173 | 0.00 | 0.02 | 0.01 | 8 | 32 | 356 | 596 | 84 | 191 | 2044 | 0 | 0 | 0 | 0 |
| wheel_06 | 1038 | 119 | 0.00 | 0.02 | 0.01 | 7 | 56 | 378 | 648 | 132 | 321 | 1038 | 0 | 0 | 0 | 0 |
| wheel_07 | 3967 | 300 | 0.00 | 0.10 | 0.01 | 14 | 152 | 1011 | 1329 | 251 | 457 | 3950 | 15 | 0 | 2 | 0 |
| wheel_08 | 4175 | 300 | 0.00 | 0.20 | 0.01 | 6 | 125 | 1202 | 1487 | 256 | 469 | 4155 | 17 | 0 | 3 | 0 |
| wheel_09 | 4463 | 300 | 0.00 | 0.83 | 0.02 | 5 | 185 | 1703 | 2030 | 342 | 588 | 4428 | 33 | 0 | 1 | 1 |
| wheel_10 | 4450 | 300 | 0.00 | 2.14 | 0.04 | 5 | 235 | 2298 | 2631 | 469 | 735 | 4387 | 37 | 0 | 2 | 24 |
| wheel_11 | 4092 | 300 | 0.00 | 0.02 | 0.01 | 23 | 48 | 215 | 533 | 59 | 309 | 4092 | 0 | 0 | 0 | 0 |
| hakafu_01 | 691 | 37 | 0.00 | 0.02 | 0.02 | 12 | 6612 | 114 | 13698 | 39 | 9902 | 691 | 0 | 0 | 0 | 0 |
| hakafu_02 | 10180 | 300 | 0.00 | 0.26 | 0.03 | 1 | 7004 | 2830 | 14140 | 347 | 10393 | 10086 | 48 | 41 | 5 | 0 |
| hakafu_03 | 5723 | 164 | 0.01 | 1.01 | 0.09 | 1 | 7367 | 6612 | 19002 | 981 | 11268 | 5456 | 230 | 16 | 21 | 0 |
| hakafu_04 | 2397 | 70 | 0.01 | 6.95 | 0.64 | 252 | 9246 | 23974 | 36664 | 5055 | 16182 | 1591 | 771 | 0 | 35 | 0 |
| hakafu_05 | 302 | 20 | 0.00 | 0.02 | 0.01 | 5 | 4595 | 83 | 13691 | 26 | 8835 | 302 | 0 | 0 | 0 | 0 |
| hakafu_06 | 10046 | 300 | 0.00 | 0.20 | 0.03 | 0 | 4522 | 2240 | 13952 | 260 | 9360 | 9940 | 86 | 12 | 8 | 0 |
| hakafu_07 | 10118 | 300 | 0.00 | 0.57 | 0.05 | 0 | 7281 | 4540 | 15280 | 512 | 10123 | 9977 | 133 | 0 | 8 | 0 |
| hakafu_08 | 5843 | 173 | 0.00 | 4.54 | 0.29 | 0 | 6421 | 18706 | 30436 | 2507 | 12932 | 4824 | 970 | 5 | 44 | 0 |
| hakafu_09 | 118 | 8 | 0.00 | 0.02 | 0.01 | 4 | 3634 | 50 | 11482 | 15 | 8432 | 118 | 0 | 0 | 0 | 0 |
| hakafu_10 | 8761 | 300 | 0.00 | 0.08 | 0.03 | 0 | 3898 | 1065 | 12764 | 251 | 8687 | 8737 | 22 | 0 | 2 | 0 |
| hakafu_13 | 49 | 6 | 0.00 | 0.02 | 0.01 | 1 | 61 | 18 | 9362 | 4 | 3968 | 49 | 0 | 0 | 0 | 0 |
| hakafu_14 | 1296 | 56 | 0.00 | 0.05 | 0.01 | 7 | 217 | 444 | 10431 | 103 | 4704 | 1296 | 0 | 0 | 0 | 0 |
| hakafu_15 | 1226 | 73 | 0.00 | 0.12 | 0.03 | 11 | 341 | 1241 | 9431 | 343 | 5507 | 1195 | 30 | 0 | 1 | 0 |
| hakafu_17 | 854 | 45 | 0.00 | 0.02 | 0.02 | 4 | 5890 | 95 | 13675 | 36 | 9794 | 854 | 0 | 0 | 0 | 0 |
| hakafu_18 | 10356 | 300 | 0.00 | 0.22 | 0.03 | 0 | 6097 | 2543 | 14375 | 315 | 9887 | 10278 | 64 | 5 | 9 | 0 |
| hakafu_19 | 6937 | 210 | 0.00 | 0.82 | 0.07 | 1 | 6854 | 5798 | 18188 | 850 | 10928 | 6523 | 388 | 3 | 23 | 0 |
| hakafu_21 | 286 | 17 | 0.00 | 0.02 | 0.01 | 2 | 4442 | 72 | 13668 | 25 | 9420 | 286 | 0 | 0 | 0 | 0 |
| hakafu_22 | 9759 | 300 | 0.00 | 0.16 | 0.02 | 3 | 4988 | 2004 | 13314 | 253 | 9280 | 9669 | 68 | 10 | 12 | 0 |
| hakafu_23 | 9529 | 300 | 0.00 | 0.57 | 0.05 | 0 | 5111 | 4748 | 17138 | 539 | 10351 | 9224 | 300 | 1 | 4 | 0 |
| hakafu_24 | 7699 | 222 | 0.00 | 11.90 | 0.25 | 0 | 7354 | 17214 | 28194 | 2049 | 12463 | 6373 | 1309 | 0 | 17 | 0 |
| hakafu_25 | 115 | 9 | 0.00 | 0.02 | 0.01 | 1 | 421 | 46 | 10945 | 16 | 7801 | 115 | 0 | 0 | 0 | 0 |
| hakafu_26 | 8851 | 300 | 0.00 | 0.09 | 0.02 | 4 | 4366 | 1162 | 10762 | 226 | 8040 | 8837 | 12 | 0 | 2 | 0 |

*continued on next page ...*

*Table 6.5:* MIP statistics for the configuration pricing of the solved instances.

| Instance | MIPs | Iter | Solving Times | | | Problem size | | | | | | Solution found through ... | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $t^{\min}$ (sec) | $t^{\max}$ (sec) | $t^{\text{avg}}$ (sec) | $c^{\min}$ | $r^{\min}$ | $c^{\max}$ | $r^{\max}$ | $c^{\text{avg}}$ | $r^{\text{avg}}$ | LP | k | R | s | E |
| hakafu_27 | 7678 | 300 | 0.00 | 0.26 | 0.05 | 11 | 701 | 2618 | 14048 | 623 | 8936 | 7062 | 607 | 0 | 9 | 0 |
| hakafu_28 | 8655 | 300 | 0.00 | 2.92 | 0.24 | 2 | 4773 | 8139 | 18039 | 1988 | 10642 | 7194 | 1435 | 0 | 26 | 0 |
| hakafu_29 | 43 | 7 | 0.00 | 0.02 | 0.01 | 1 | 31 | 10 | 5890 | 2 | 3342 | 43 | 0 | 0 | 0 | 0 |
| sit_01 | 4985 | 300 | 0.00 | 0.11 | 0.01 | 0 | 0 | 450 | 3288 | 140 | 1236 | 4797 | 167 | 2 | 19 | 0 |
| sit_02 | 5310 | 300 | 0.00 | 0.02 | 0.01 | 5 | 5 | 358 | 4630 | 181 | 1833 | 5216 | 94 | 0 | 0 | 0 |
| sit_03 | 5330 | 300 | 0.00 | 0.02 | 0.01 | 4 | 4 | 349 | 4621 | 171 | 1815 | 5330 | 0 | 0 | 0 | 0 |
| sit_04 | 5165 | 300 | 0.00 | 0.06 | 0.01 | 0 | 1 | 361 | 1906 | 137 | 741 | 5033 | 86 | 6 | 40 | 0 |
| sit_05 | 5131 | 300 | 0.00 | 0.08 | 0.01 | 1 | 1 | 462 | 2592 | 160 | 1037 | 5017 | 79 | 21 | 14 | 0 |
| sit_06 | 4666 | 300 | 0.00 | 0.03 | 0.01 | 1 | 16 | 184 | 784 | 75 | 363 | 4521 | 106 | 32 | 7 | 0 |
| sit_07 | 3642 | 206 | 0.00 | 0.39 | 0.02 | 2 | 2 | 1283 | 5507 | 407 | 2075 | 3556 | 79 | 3 | 3 | 0 |
| sit_08 | 5002 | 300 | 0.00 | 0.03 | 0.01 | 3 | 24 | 410 | 4495 | 181 | 1909 | 4755 | 205 | 12 | 30 | 0 |
| sit_09 | 4470 | 262 | 0.00 | 0.03 | 0.01 | 0 | 0 | 614 | 4819 | 317 | 2053 | 4347 | 117 | 3 | 3 | 0 |

# Bibliography


Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805. Available from: http://mor.journal.informs.org/cgi/content/abstract/23/4/769.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions to uncertain programs. *Operations Research Letters*, 25:1–13.

Ben-Tal, A. and Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424.

Ben-Tal, A. and Nemirovski, A. (2002). Robust Optimization—Methodology and Applications. *Mathematical Programming*, 92(3):453–480. Available from: http://www.springerlink.com/content/jk7bqe1jyvexk24u/.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71. Available from: http://www.springerlink.com/content/5nhd1pk9bpakh0c2/.

Bertsimas, D. and Sim, M. (2004). The Price of Robustness. *Operations Research*, 52(1):35–53. Available from: http://or.journal.informs.org/cgi/content/abstract/52/1/35.

BMVBS (25. February 2009). Homepage of the German Federal Ministry of Transport, Building and Urban Affairs (bundesministerium für Verkehr, Bau und Stadtentwicklung). Available from: http://www.bmvbs.de.

Borndörfer, R., Grötschel, M., Lukac, S., Mitusch, K., Schlechte, T., Schultz, S., and Tanner, A. (2006). An auctioning approach to railway slot allocation. *Competition and Regulation in Network Industries*, 1(2):163–196. Available from: http://opus.kobv.de/zib/volltexte/2005/878.

Borndörfer, R. and Schlechte, T. (2007a). Models for Railway Track Allocation. In Liebchen, C., Ahuja, R. K., and Mesa, J. A., editors, *ATMOS 2007—7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum f"ur Informatik (IBFI), Schloss Dagstuhl, Germany. Available from: http://drops.dagstuhl.de/opus/volltexte/2007/1170.

Borndörfer, R. and Schlechte, T. (2007b). Solving Railway Track Allocation Problems. In Kalcsics, J. and Nickel, S., editors, *Operations Research Proceedings 2007*, pages 117–122. Available from: `http://opus.kobv.de/zib/volltexte/2007/974`.

Borndörfer, R. and Schlechte, T. (2008). Balancing efficiency and robustness. In Ehrgott, M., Naujoks, B., Stewart, T., and Wallenius, J., editors, *MCDM for Sustainable Energy and Transportation Systems*. Lecture Notes in Economics and Mathematical Systems. Available from: `http://opus.kobv.de/zib/volltexte/2008/1105/`.

Brännlund, U., Lindberg, P. O., Nou, A., and Nilsson, J.-E. (1998). Railway Timetabling Using Lagrangian Relaxation. *Transportation Science*, 32(4):358–369. Available from: `http://transci.journal.informs.org/cgi/reprint/32/4/358`.

Cacchiani, V. (2009). *Models and algorithms for combinatorial optimization problems arising in railway applications*. PhD thesis, Università di Bologna. Available from: `http://www.springerlink.com/content/f62838x63546vp7q/`.

Cacchiani, V., Caprara, A., and Toth, P. (2008). A column generation approach to train timetabling on a corridor. *4OR: A Quarterly Journal of Operations Research*, 6(2):125–142. Available from: `http://www.springerlink.com/content/c638656380058577/`.

Caprara, A., Fischetti, M., and Toth, P. (2002). Modeling and Solving the Train Timetabling Problem. *Operations Research*, 50(5):851–861.

Caprara, A., Kellerer, H., and Pferschy, U. (2000). The Multiple Subset Sum Problem. *SIAM Journal of Optimization*, 11:308–319.

Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P. (2007). Passenger Railway Optimization. In Barnhart, C. and Laporte, G., editors, *Handbooks in Operations Research and Management Science*, volume 14, chapter 3, pages 129–187. Elsevier. Available from: `http://arrival.cti.gr/index.php/Documents/0035`.

Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors (2005). *Column Generation*. Gerad 25th Anniversary. Springer Berlin / Heidelberg.

Ehrgott, M. and Ruzika, S. (2005). An Improved $\varepsilon$-Constraint Method for Multiobjective Programming. Report in wirtschaftsmathematik (wima report), Technischen Universität Kaiserslautern. Available from: `http://kluedo.ub.uni-kl.de/volltexte/2005/1893/`.

El-Ghaoui, L. and Lebret, H. (1997). Robust solutions to least-square problems to uncertain data matrices. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064.

El-Ghaoui, L., Oustry, F., and Lebret, H. (1998). Robust solutions to uncertain semidefinite programs. *SIAM Journal of Optimization*, 9(1):33–52.

Erdoğan, C. (2009). Computing Prices For Track Allocation. Master's thesis, Technische Universität Berlin.

Erol, B., Klemenz, M., Schlechte, T., Schultz, S., and Tanner, A. (2008). TTPLIB 2008—A Library for Train Timetabling Problems. Available from: `http://opus.kobv.de/zib/volltexte/2008/1102/`.

EUR-Lex (2009). Homepage of the Office for Official Publications of the European Communities. Available from: `http://eur-lex.europa.eu`.

Eurostat (2009). Homepage of the Statistical Office of the European Communities. Available from: `http://epp.eurostat.ec.europa.eu/`.

Fischetti, M., Zanette, A., and Salvagnin, D. (2007). 10. fast approaches to robust railway timetabling. In Liebchen, C., Ahuja, R. K., and Mesa, J. A., editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. Available from: `http://drops.dagstuhl.de/opus/volltexte/2007/1176`.

Grötschel, M. (2006). *Graphen- und Netzwerkalgorithmen (lecture notes)*. Technische Universität Berlin. Available from: `http://www.zib.de/groetschel/teaching/skript-SS2006.pdf`.

Grötschel, M. (2006/2007). *Linear and Integer Programming (lecture notes)*. Technische Universität Berlin. Available from: `http://www.zib.de/groetschel/teaching/skriptADMII.pdf`.

Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197. Available from: `http://www.zib.de/groetschel/pubnew/biblio.htm`.

Hansen, I. A. and Pachl, J., editors (2008). *Railway Timetable & Traffic: Analysis—Modelling—Simulation*. Eurailpress, 1 edition.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer Berlin-Heidelberg.

Korte, B. and Vygen, J. (2008). *Combinatorial Optimization*. Algorithms and Combinatorics 21. Springer Berlin-Heidelberg, fourth edition edition. Available from: `http://www.springerlink.com/content/978-3-540-71843-7`.

Kroon, L. G., Dekker, R., and Vromans, M. J. C. M. (2007). Cyclic Railway Timetabling: A Stochastic Optimization Approach. *Lecture Notes in Computer Science*, 4359/2007:41–66. Available from: `http://hdl.handle.net/1765/6957`.

Liebchen, C. and Möhring, R. H. (2008). The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables—and Beyond. In *Computer-aided Systems in Public Transport*, volume 600 of *Lecture Notes in Economics and Mathematical Systems*, pages 117–150. Springer Berlin Heidelberg. Available from: `http://www.springerlink.com/content/n301j051u066l655/`.

Lukac, S. G. (2004). Holes, Antiholes, and Maximal Cliques in a Railway Model for a Single Track. Technical Report ZR-04-18, Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany. Available from: `http://opus.kobv.de/zib/volltexte/2004/794/pdf/ZR-04-18.pdf`.

Möhring, R. (2006). *Angewandte Netzwerkoptimierung (lecture notes)*. Technische Universität Berlin. Available from: `http://www.math.tu-berlin.de/~moehring/adm3/`.

Pachl, J. (2008). Glossary of railroad operation and control. Available from: `http://joernpachl.gmxhome.de/glossary.htm`.

Schrijver, A. (2003). *Combinatorial Optimization—Polyhedra and Efficiency*, volume A-B-C of *Algorithms and Combinatorics 24*. Springer Berlin-Heidelberg.

Serafini, P. and Ukovich, W. (1989). A Mathematical Model for Periodic Scheduling Problems. *SIAM J. Discrete Math.*, 2(4):550–581.

Soyster, A. L. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157. Available from: `http://www.jstor.org/pss/168933`.