Investigating Robustness, Public Transport Optimization, and their Interface

Mathematical Models and Solution Algorithms

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades

„Doctor rerum naturalium“

der Georg-August-Universität Göttingen

im Promotionsprogramm „PhD School of Mathematical Sciences“ (SMS)

der Georg-August University School of Science (GAUSS)

vorgelegt von

Julius Pätzold

aus Verden (Aller)

Göttingen, 2019

Betreuungsausschuss

Prof. Dr. Anita Schöbel
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen
seit 2019: Fachbereich Mathematik
Technische Universität Kaiserslautern

Jun.-Prof. Dr. Anja Fischer
Juniorprofessur Management Science
Technische Universität Dortmund

Mitglieder der Prüfungskommission

Referentin
Prof. Dr. Anita Schöbel
Fachbereich Mathematik
Technische Universität Kaiserslautern

Korreferent
Prof. Dr. Marc Goerigk
Institut für Betriebswirtschaftslehre
Universität Siegen

Weitere Mitglieder der Prüfungskommission

Prof. Dr. Stefan Halverscheid
Mathematisches Institut
Georg-August-Universität Göttingen

Jun.-Prof. Dr. Christoph Lehrenfeld
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Prof. Dr. Gerlind Plonka-Hoch
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Prof. Dr. Stephan Westphal
Institut für Mathematik
Technische Universität Clausthal

Tag der mündlichen Prüfung: 28.06.2019

Contents

1. Introduction

Mathematical optimization represents a key branch of applied mathematics thanks to its potential to model and solve real world decision processes. The topics *public transport optimization* and *robust optimization*, which constitute two emerging subfields of mathematical optimization, define the scope of this dissertation: Public transport optimization investigates the design of a public transport system, such as a bus or train network. Its objective is to formulate mathematical models that provide plans of public transport system with increased quality. Whereas public transport optimization thus investigates a concrete application, research in the field of robust optimization is concerned with providing solutions to decision making processes under uncertainty. It explores concepts and methods for obtaining solutions that minimize a certain robustness measure for a given problem. The robustness measure is determined by a set of scenarios for which it is unknown which scenario might be revealed. To illustrate this scheme, consider the application of robustness to public transport, in particular, to timetabling. Finding a robust timetable for a public transport system can be conceived of as finding a timetable that is resistant with respect to one or more delay scenarios that have not yet been revealed. Exactly this application of robustness methods to public transport is a central point in this dissertation.

More generally, by investigating robustness, public transport optimization, and their interface, this dissertation contributes research to the field of mathematical optimization that satisfies the claim of applicability in three different ways. First, by considering public transport optimization, a socially highly relevant application is investigated. With the growing demand for mobility as well as ecological awareness, public transport has become an increasingly important topic for the modern world society. Thus, there exists a need for public transport systems that are passenger-convenient while keeping costs reasonable low. Second, the investigation of robust optimization and its incorporation into public transport optimization enables the inclusion of naturally arising uncertainties into optimization models, which leads to more realistic and consequently more applicable mathematical models. Third, by presenting solution algorithms as well as computational experiments on their respective implementations, the applicability of the models introduced in this dissertation is emphasized. In fact, every contribution contains efficient implementations of the presented algorithms, constituting a proof-of-concept of their adaptability to real world problems.

Via the cumulation of five individual but thematically connected publications, this dissertation thus makes a novel contribution to the field of applied mathematics. The first three publications focus on public transport optimization, the fourth on robust optimization, and the fifth on the incorporation of robust optimization into public transport optimization. More concretely, the first two publications are concerned with designing cost-minimal public transport systems by integrating several subproblems of public transport optimization. The paper *Look-Ahead Approaches for Integrated Planning in Public Transportation* does this heuristically, whereas *Cost-Minimal Public Transport Planning* provides exact models for finding cost-minimal public transport systems. The third public transport contribution, *The Trickle-in Effect: Modeling Passenger Behavior in Delay Management*, considers passenger-convenience by integrating passenger movements at train stations into a delay management model. Publication four, *Approximate Cutting Plane Approaches for Exact Solutions to Robust Optimization Problems*, is methodology- and not application-driven. It focuses on cutting plane techniques that are used to solve robust optimization problems and introduces speed-up techniques for these problems by approximatively solving occurring subproblems, which are induced by the cutting plane scheme. The fifth publication, *Finding Robust Periodic Timetables by Integrating Delay Management*, then combines the topics of public transport and robust optimization through formulating a robust timetabling problem in order to find delay-resistant timetables. By doing so it integrates public transport problems – as done in the first two publications – considers delay management, as in the third publication, and makes use of speed-up techniques for cutting plane algorithms for its solution algorithms – as presented in the fourth contribution.

The remainder of this dissertation is structured as follows: Chapter 2 introduces the topics of public transport- and robust optimization, and provides a survey of the relevant literature. Chapter 3 summarizes the publications of the dissertation, which are further discussed in Chapter 4. The dissertation concludes in Chapter 5 and the author's contributions to the publications are summarized in Chapter 6. Finally, all five publications are provided in the Appendix.

2. Literature Review

This section briefly introduces the investigated topics and provides a literature overview for each of them. First, research in public transport optimization is presented in Section 2.1. Section 2.2 then provides a concise overview on robust optimization before Section 2.3 combines both research areas via discussing robustness considerations in public transport optimization.

2.1. Public Transport Optimization

Over the last decades, different endeavors have been made to transform aspects of public transportation planning into mathematical models. Relevant works that not only consider a particular problem but also formulate a chain of planning problems include Ceder and Wilson (1986), Bussieck, Winter, and Zimmermann (1997), Huisman, Kroon, Lentink, and Vromans (2005), Desaulniers and Hickman (2007). One version of such a planning chain, which this thesis follows, is presented in Figure 1.

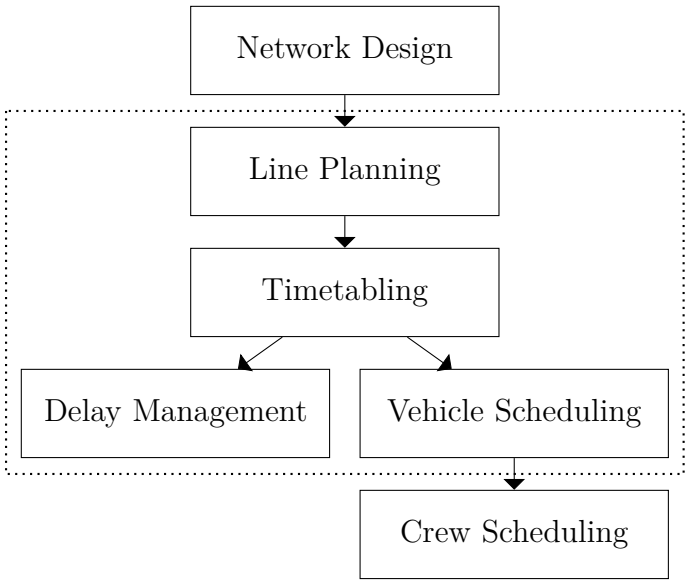


Figure 1: Planning Chain in Public Transport

In this planning chain, *Network design* investigates the location of stops or stations

and tracks between them to create or extend a public transportation network. *Line planning* focusses on creating lines, i.e., fixed repeating transportation routes, in the network. *Timetabling* then schedules departures and arrivals for all lines at all stations the lines are passing. Following timetabling, two different stages can be considered independently. On the one hand, *delay management* is concerned with rescheduling the timetable after some delays have been revealed. On the other hand, *vehicle scheduling* assigns vehicles to execute the lines as specified in the timetable. Finally, *crew scheduling* assigns personnel to each vehicle of the vehicle schedule.

Much of the literature in public transport optimization considers one individual problem of the planning chain. A recent approach, however, is to *integrate*, i.e., combine, successive planning problems, resulting in more difficult optimization problems. Examples for successfully approaching these complex integrated problems in order to obtain better overall solutions are given in Schmidt (2014), Schöbel (2017), Burggraeve, Bull, Vansteenwegen, and Lusby (2017), Schiewe (2018), Schiewe (2019). The first two publications of this dissertation fall into this category of integrated public transport planning as well. Notwithstanding the approaches of integrating several planning problems, each problem from the planning chain is now introduced separately, its relevant literature is presented, and connections to its adjacent problems are illustrated. The scope of considered public transport problems is restricted in this thesis to line planning as a starting point and delay management and vehicle scheduling, respectively, as an ending point. For research about the preceding problem of network design, confer to literature overviews in Laporte, Mesa, and Ortega (2000), Kepaptsoglou and Karlaftis (2009), Laporte, Mesa, Ortega, and Perea (2011). A survey on the subsequent problem of crew scheduling can be found in Van den Bergh, Beliën, De Bruecker, Demeulemeester, and De Boeck (2013).

As a prerequisite for line planning it is commonly assumed that the network design has been completed and a *public transportation network* (PTN) is given. A PTN is a graph (V, E) with nodes modeling bus stops or train stations and edges modeling streets or tracks between them. Furthermore, it is assumed that an *OD-matrix* $W \in \mathbb{R}^{|V|\times|V|}$, denoting for each pair of stops the number of passengers that want to travel between them, has already been obtained.

Line Planning

The Line planning problem decides on which routes and how often rides should be offered. Given a PTN and an OD-matrix, line planning is preceded by estimating a passenger distribution for every edge in the PTN, called *load*. Models for this passenger distribution are given in Desaulniers and Hickman (2007), Patriksson (2015), Friedrich, Hartl, Schiewe, and Schöbel (2017). With a given passenger load it is then determined how often per time period, e.g. an hour, every edge $e \in E$ of the PTN needs to be served by some vehicle (cf. Schöbel, 2012) which is called minimal

frequency f_e^{min}. Additionally, it is also assumed that for every edge there exists some maximal frequency f_e^{max}, induced, for example, by security regulations. The goal of line planning is to determine a *line concept* $(\mathcal{L}, f)$, consisting of a set of *lines* $\mathcal{L}$, which are defined as simple paths on the PTN graph (V, E), and a *frequency* f_l for every line denoting how often it has to be executed every hour. The line concept has to respect minimum and maximum frequency bounds on every edge in the PTN, i.e.,

$$f_e^{min} \leq \sum_{l \in L : e \in l} f_l \leq f_e^{max} \quad \forall e \in E. \tag{2.1}$$

A line concept can be determined with respect to different objective functions. Cost-oriented models, for example, estimate the incurring costs of the resulting public transport system. Cost-oriented models are investigated in Zwaneveld (1997), Claessens, van Dijk, and Zwaneveld (1998), Goossens, Van Hoesel, and Kroon (2004), Goossens, van Hoesel, and Kroon (2006). Passenger-oriented models, on the other hand, estimate the value of a line concept with respect to passenger-convenience. Convenience for passengers has been modeled by optimizing the number of direct travelers (see Bussieck, 1998), estimated passenger travel times (see Schöbel and Scholl, 2006; Borndörfer, Grötschel, and Pfetsch, 2007), or the number of transfers (see Harbering, 2016). Passenger route choice can also be modeled as a separate subproblem, as in Borndörfer and Karbstein (2012), Schmidt and Schöbel (2015a), Goerigk and Schmidt (2017). In order to reduce the problem complexity of line planning, a common simplification is to restrict the set of all possible lines to a smaller subset, called *line pool*. Algorithms for determining line pools are presented in Gattermann, Harbering, and Schöbel (2017). With a line pool at hand, the line planning problem reduces to assigning a frequency to each line in the line pool. A detailed overview on line planning models is given in Schöbel (2012) and an experimental comparison of different line planning models is presented in Goerigk, Schachtebeck, and Schöbel (2013).

Timetabling

The next step in the planning chain is timetabling. Timetabling determines departure and arrival times for every line of the line concept. In order to model train departure and arrivals, the line concept is converted into an *event-activity-network (EAN)* – a directed graph $(\mathcal{E}, \mathcal{A})$ whose nodes are called *events* and edges called *activities*. For every line l, every departure and arrival for every stop the line passes is modeled as a different event. Formally, if $(s_{l,1}, \ldots, s_{l,n})$ is the sequence of stops line l passes, then we construct events $\{s_{l,1}^{\text{dep}}, s_{l,2}^{\text{arr}}, s_{l,2}^{\text{dep}}, \ldots, s_{l,n-1}^{\text{dep}}, s_{l,n}^{\text{arr}}\} \subset \mathcal{E}$ denoting departure and arrival events at every stop and add driving activities $(s_{l,i}^{\text{dep}}, s_{l,i+1}^{\text{arr}}) \in \mathcal{A}$ for

$i \in \{1, \dots, n-1\}$ and waiting activities $(s^{\text{arr}}_{l,i}, s^{\text{dep}}_{l,i}) \in \mathcal{A}$ for $i \in \{2, \dots, n-1\}$. Further activities can be inserted into the model, including changing activities (passengers transferring from one train to another), synchronization activities (modeling line frequencies) and headway activities (modeling minimum time distances between two consecutive trains on the same track), see, e.g., Peeters (2003). Every activity $a \in \mathcal{A}$ of the EAN is equipped with lower und upper bounds $[L_a, U_a]$ on their duration and some weight w_a denoting the number of passengers using that activity.

In accordance with the preceding planning stages, i.e., estimating the passenger volume for a certain time period and determining a line concept for this time period, the timetabling problem finds a periodic timetable that is repeatedly operated for a given *time period* $T \in \mathbb{N}$.

A *timetable* $\pi \in \mathbb{N}^{|\mathcal{E}|}$ assigns an integral point of time to every event $e \in \mathcal{E}$ of the EAN such that the duration of every activity $a \in \mathcal{A}$ lies in $[L_a, U_a]$, i.e.,

$$L_a \leq (\pi_j - \pi_i - L_a) \bmod T + L_a \leq U_a \quad \forall a = (i,j) \in \mathcal{A}. \tag{2.2}$$

Note that due to the periodicity it is not possible to measure the duration between two events i and j simply as $\pi_j - \pi_i$; but instead we have to incorporate the modulo operator and take its representative in $\{0, \dots, T-1\}$ (cf. Pätzold and Schöbel, 2016). The usually considered objective function of periodic timetabling is to minimize the sum of weighted activity durations, leading to the *Periodic Event Scheduling Problem (PESP)*. (PESP) has been introduced in Serafini and Ukovich (1989) and since then been thoroughly studied, see Odijk (1996), Nachtigall (1998), Peeters (2003), Liebchen (2007). Important insights are the cycle-base formulation (see, e.g., Peeters and Kroon, 2001; Borndörfer, Hoppmann, Karbstein, and Lindner, 2018), the modulo simplex algorithm (Nachtigall and Opitz, 2008; Goerigk and Schöbel, 2013) and SAT-Formulations (Großmann et al., 2012). An experimental comparison of different timetabling algorithms has been given in Siebert and Goerigk (2013). Current state-of-the-art solutions (measured by their performance on PESP instances from Goerigk, 2018) include Pätzold and Schöbel (2016), Goerigk and Liebchen (2017) and Borndörfer, Lindner, and Roth (2019). An important shortcoming of PESP, however, is that the number of passengers w_a is fixed for all $a \in \mathcal{A}$ before the timetable is known. After the timetable has been determined, passengers may switch their routes to a shorter one resulting in a change of weights w_a for some activities $a \in \mathcal{A}$. To resolve this issue integrated models of timetabling and passenger routing are presented in Schmidt (2014), Schmidt and Schöbel (2015a, 2015b), Gattermann, Großmann, Nachtigall, and Schöbel (2016b), Borndörfer, Hoppmann, and Karbstein (2017), Schiewe (2018).

Furthermore, timetabling models do not necessarily have to be induced by a line concept and periodicity requirements. Aperiodic timetabling models work in a slightly different setting by scheduling a given set of rides instead of determining

a regular and periodic transportation supply. Surveys on various timetabling problems are given in Caprara, Fischetti, and Toth (2002), Lusby, Larsen, Ehrgott, and Ryan (2011), Cacchiani and Toth (2012).

Vehicle Scheduling

After departure and arrival times have been determined for every line of the line concept, the vehicle scheduling step of the planning chain assigns vehicles to execute the rides specified by the line concept and timetable. To obtain a vehicle scheduling instance, periodic timetable and EAN are *rolled out* for a certain number of time periods $p \in \mathbb{N}$, e.g., $p = 24$ for a whole day. This means that an aperiodic EAN is created by sequentially connecting p copies of the periodic EAN. The roll-out hence converts the periodic timetable and EAN to an aperiodic timetable and EAN. By doing so one receives a set of trips $\mathcal{T}$ containing every execution of each line $l \in \mathcal{L}$ over all time periods. In vehicle scheduling it is looked for a set of rides $\mathcal{R}$ such that every trip $t \in \mathcal{T}$ is contained in exactly one ride $r = (t_1, \ldots, t_n) \in \mathcal{R}$. Every pair of consecutive rides (t_i, t_{i+1}) has to satisfy certain feasibility conditions, e.g., the time at which trip t_i finishes has to be smaller than the time at which trip t_{i+1} starts. Additionally, every pair of consecutive trips (t_i, t_{i+1}) induces costs $c_{t_i,t_{i+1}}$, which model the costs of a vehicle driving from the end of trip t_i to the start of trip t_{i+1}, and that need to be minimized. A survey of vehicle scheduling models is given in Bunte and Kliewer (2009) and more recent works, which include the integration of related subproblems, are Borndörfer, Reuther, Schlechte, and Weider (2011), Giacco, D'Ariano, and Pacciarelli (2014), Borndörfer, Reuther, Schlechte, Waas, and Weider (2015). Next to minimizing occurring costs, another common objective of vehicle scheduling is to minimize the number of required vehicles to operate the timetable. This has been proven to be successful in practice, see Liebchen (2008). Vehicle scheduling can also be defined on a periodic EAN such that the roll-out is omitted: Periodic vehicle scheduling decides on connecting lines (i.e., last event of one line to the first event of another line) by adding *turnaround* activities to the periodic EAN, instead of constructing and connecting trips $\mathcal{T}$. In Borndörfer, Karbstein, Liebchen, and Lindner (2018) the two problems of aperiodic and periodic vehicle scheduling yield similar vehicle schedules if the number of time periods p for the aperiodic EAN is chosen to be large enough. Nevertheless, the problem of aperiodic vehicle scheduling can be straightforwardly extended with additional real-world details – like introducing vehicle depots, at which vehicles have to start and return at the end of the day – and can also be integrated with crew scheduling, cf. Huisman, Freling, and Wagelmans (2005), Steinzen, Gintner, Suhl, and Kliewer (2010).

Delay Management

After choosing a line plan, a timetable, and possibly also a vehicle schedule, the execution of the resulting public transport plan is not guaranteed to work smoothly due to potentially occurring delays. Delays, so-called *source delays*, can occur due to construction work, weather conditions, sudden infrastructure unavailabilities, or vehicle malfunctions. Delay management is concerned with coping with source delays by rescheduling the timetable. Mathematically, source delays $s_i \geq 0$ for $i \in \mathcal{E} \cup \mathcal{A}$ are defined on events and activities of an aperiodic EAN $(\mathcal{E}, \mathcal{A})$ and the goal of delay management is to find a *disposition timetable* $d \in \mathbb{R}^{|E|}$ that reschedules the timetable π by satisfying

$$d_i \geq \pi_i + s_i \quad \forall i \in \mathcal{E} \tag{2.3}$$

$$d_j - d_i \geq s_a + \pi_j - \pi_i \quad \forall a = (i,j) \in \mathcal{A}', \tag{2.4}$$

with $\mathcal{A}' \subseteq \mathcal{A}$ being all driving, waiting and headway activities from $\mathcal{A}$, cf. Schachtebeck (2010). In addition to satisfying these minimum requirements, delay management aims at keeping the induced passenger delay minimal. The first models with this objective have been presented in Schöbel (2001), Suhl, Biederbick, and Kliewer (2001) and integer programming models have been developed in Schöbel (2007), De Giovanni, Heilporn, and Labbé (2008). In order to make the delay management models more realistic, capacities along tracks have been included in Schachtebeck and Schöbel (2010), capacities at stations have been included in Dollevoet, Huisman, Kroon, Schmidt, and Schöbel (2015) and passenger re-routing has been studied in Dollevoet, Huisman, Schmidt, and Schöbel (2012), Schmidt and Schöbel (2015a), Rückert, Lemnian, Blendinger, Rechner, and Müller-Hannemann (2017). Rescheduling of timetables, rolling stock and crew is studied in Dollevoet, Huisman, Kroon, Veelenturf, and J.C.Wagenaar (2017). Albert, Kraus, Müller, and Schöbel (2018) simulate the trickling effect, i.e., the passenger behavior at train stations which will further be investigated in the third publication in this dissertation. A recent survey of state of the art delay management is given in Dollevoet, Huisman, Schmidt, and Schöbel (2018).

2.2. Robust Optimization

Research in robust optimization started with Soyster (1973) and began growing two decades later with the works of Ghaoui and Lebret (1997), Ben-Tal and Nemirovski (1998, 2000), Bertsimas and Sim (2004), Ben-Tal, Ghaoui, and Nemirovski (2006). Robust optimization builds mathematical models around the fact that in reality almost every optimization problem is influenced by some uncertainty that is unknown

at the time a solution has to be chosen. Uncertainty can arise because input parameters of an optimization problem might not be exactly known or a solution to an optimization problem cannot be realized as intended. Formally, we consider an optimization problem

$$\min_{x \in X} g(x) \tag{P}$$

with $X \coloneqq \{x \in \mathbb{R}^n | F(x) \leq 0\}$ for some $F : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}$. Uncertainty is incorporated into the model as a set of scenarios $\mathcal{U}$ such that only after a solution x is chosen, a scenario $u \in \mathcal{U}$ is revealed. Contrary to the idea of stochastic optimization, robust optimization makes no distribution assumption on the uncertainty set $\mathcal{U}$. Instead, robust optimization considers a family of optimization problems $\mathrm{P}(\mathcal{U}) = (\mathrm{P(u)})_{u \in \mathcal{U}}$, i.e.,

$$\min_{x \in \mathcal{X}(u)} g(x, u)^1 \tag{P(u)}$$

with $X(u) \coloneqq \{x \in \mathbb{R}^n | F(x, u) \leq 0\}$. Hence, every $u \in \mathcal{U}$ leads to a different optimization problem, but when deciding on a solution x it is not known which scenario will be revealed. Nevertheless, since a solution has to be chosen regardless, the idea of robustness concepts was introduced, which lead to mathematical models, called *robust counterparts*, for obtaining a solution.

An important traditional robustness concept is *strict robustness*, which requires a solution x to be feasible for all scenarios $u \in \mathcal{U}$, and then chooses a solution x^* that has the minimal objective value in its worst-case, i.e., $\sup_{u \in \mathcal{U}} g(x^*, u)$. See Ben-Tal, Ghaoui, and Nemirovski (2009) for a compendium on results for strict robustness. Hence, we define the strictly robust counterpart as

$$\begin{aligned} \min_{x \in X} \sup_{u \in \mathcal{U}} & g(x, u) \\ \text{s.t.} \quad & F(x, u) \leq 0 \quad \forall u \in \mathcal{U}. \end{aligned} \tag{Strict RC}$$

See Kouvelis and Yu (1997), Aissi, Bazgan, and Vanderpooten (2009) for algorithms and applications for strict robustness on combinatorial optimization problems.

Another popular robustness concept is *adjustable robustness*, which partitions the solution x into two parts $x = (x^1, x^2) \in X_1 \times X_2$ (cf. Ben-Tal, Goryashko, Guslitzer, and Nemirovski, 2004). The first part x^1 is called *here-and-now*-variables which have to be determined before scenario u is revealed. The second part x^2 is called *wait-and-see*-variables which can be chosen after the scenario $u \in \mathcal{U}$ is known. Most importantly, x^1 has to be chosen such that there exists an x^2 such that the overall solution (x^1, x^2) remains feasible for every $u \in \mathcal{U}$. Hence, define

$$\mathcal{X}_1 \coloneqq \{x^1 \in X_1 | \, \forall u \in \mathcal{U} \, \exists \, x^2 \in X_2 : F(x^1, x^2, u) \leq 0\}.$$

[1] Notwithstanding the chosen layout paradigm (consistent notation throughout the thesis while minimizing inconsistencies with the notation used in the publications), the author is aware of the standard notation used in the robust optimization literature ($f(x, \xi)$ instead of $g(x, u)$).

The adjustable robust counterpart then states as

$$\min_{x^1 \in \mathcal{X}_1} \sup_{u \in \mathcal{U}} \inf_{x^2 \in X_2} g(x^1, x^2, u) \qquad \text{(Adjustable RC)}$$
$$\text{s.t.} \quad F(x^1, x^2, u) \leq 0 \quad \forall u \in \mathcal{U}.$$

In addition to the presented robustness concepts, there exist many more approaches to capture robustness, e.g., regret robustness, light robustness or recoverable robustness. To this end, confer to Goerigk (2012), Goerigk and Schöbel (2016) for an overview.

Despite of the differences of the two introduced robustness concepts, cutting planes approaches, due to Kelley (1960), have been investigated for both of them. Cutting plane algorithms work iteratively by starting with a small subset $\mathcal{U}_0 \subset \mathcal{U}$, finding a solution for the robust counterpart with respect to $\mathcal{U}_0$, calculating a worst-case scenario $u \in \mathcal{U}$ to x and adding u to $\mathcal{U}_0$, see Figure 2. This procedure is repeated until some stopping criterion for the solution x is met.

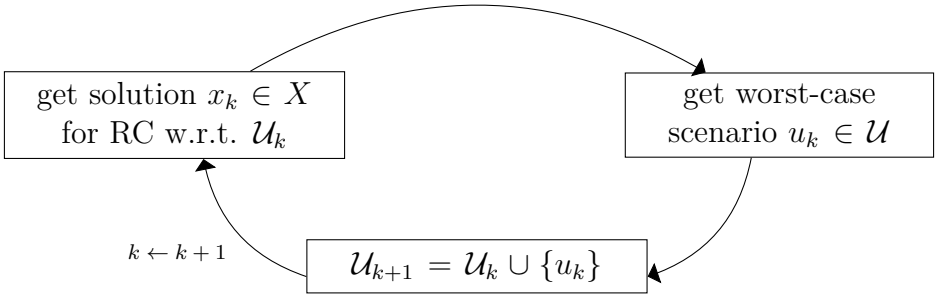


Figure 2: Cutting Plane Method for Solving Robust Counterparts

This approach has been widely studied in previous literature und has been proposed and used under many different names: Outer approximation method (Reemtsen, 1994; Bürger, Notarstefano, and Allgöwer, 2014; Goerigk and Schöbel, 2016), (modified) Benders decomposition approach (Montemanni, 2006; Siddiqui, Azarm, and Gabriel, 2011), implementor-adversarial framework (Bienstock, 2007), cutting set/plane method (Mutapcic and Boyd, 2009; Terry, 2009; Fischetti and Monaci, 2012) or scenario relaxation procedure (Assavapokee, Realff, Ammons, and Hong, 2008; Aissi et al., 2009). A convergence proof can be found in Mutapcic and Boyd (2009), which originates from the proof of the original cutting plane method given in Kelley (1960). The applicability of cutting plane methods and their computational competitiveness holds for a wide range of optimization problems, such as robust versions of combinatorial problems (see Aissi et al., 2009), mixed-integer programming (Fischetti and Monaci, 2012; Bertsimas, Dunning, and Lubin, 2016) and convex programming (Mutapcic and Boyd, 2009).

In the fourth publication we present speed-up techniques for this cutting plane method for strict robustness, and in publication five we apply a version of cutting planes to an adjustable robust counterpart.

2.3. Robustness in Public Transport Optimization

The incorporation of robust optimization into the field of public transport optimization is a canonical extension of traditional public transport optimization problems. Uncertainties occur naturally at several stages of the public transport planning process: Varying number of traveling passengers, weather conditions, infrastructure construction work, crew members falling behind schedule, and others are only a small excerpt of all aspects that are uncertain when planning a public transport system. For mathematical models which assimilate uncertainties like these, there exists a wide range of literature. A recent and thorough survey of robustness considerations in public transport optimization is given in Lusby, Larsen, and Bull (2018). Robustness extensions for the early planning stages of network design and line planning include Marín, Mesa, and Perea (2009), Laporte, Mesa, and Perea (2010), but are scarce in general, potentially due to the lack of precise timetable information. The next stage of timetabling, however, has been widely studied regarding robustness. For a survey on robustness focused on timetabling, see Cacchiani and Toth (2012). Liebchen, Lübbecke, Möhring, and Stiller (2009) introduce the concept of recoverable robustness in which a solution needs to be able to recover to a feasible solution in all scenarios. Recoverable robustness has since then been applied to timetabling, see Cicerone et al. (2009), Goerigk and Schöbel (2014). Light robustness, as proposed in Fischetti and Monaci (2009) and further investigated in Schöbel (2014), imposes a trade-off between stochastic programming and traditional robust optimization approaches and is applied to timetabling in Fischetti and Monaci (2009), Goerigk, Knoth, Müller-Hannemann, Schmidt, and Schöbel (2014). Furthermore, there exist several approaches to extend robustness to periodic timetabling: For example, Kroon, Maróti, Helmrich, Vromans, and Dekker (2008) use stochastic programming, Goerigk (2015) applies the concept of recovery robustness for periodic timetabling, and Polinder, Breugem, Dollevoet, and Maróti (2019) use adjustable robustness to obtain robust periodic timetables. The existence of robustness in delay management as a research topic for itself is doubtful since in delay management the uncertain scenario of delays is already known. Nevertheless, delay management plays an important role in connection with robust timetabling (see Cicerone, D'Angelo, Di Stefano, Frigioni, and Navarra, 2007) and as an evaluation procedure for some robust timetabling approaches, e.g., in Liebchen, Schachtebeck, Schöbel, Stiller, and Prigge (2010). Robustness considerations have also been applied to vehicle scheduling problems, or rolling stock planning, respectively. Cacchiani et al. (2012) consider

robustness as the ability of a vehicle schedule to cope with large disruptions and propose a two-stage optimization model, similar to a recovery approach, whereas Cadarso and Marín (2014) define robustness as a measure of expected propagated delays. Finally, Amberg, Amberg, and Kliewer (2018) propose a robust integrated vehicle and crew scheduling problem that similarly minimizes expected propagated delays. For robustness in crew scheduling, refer to Lusby et al. (2018).

3. Summary of the Publications

This chapter provides a summary for each of the five publications which constitute the core of this dissertation. As mentioned before, the first two papers are concerned with finding a public transport system with low cost, the third publication incorporates passenger behavior at train stations into delay management. Publication four proposes speed-up techniques for cutting plane methods in robust optimization and publication five unites the previous topics by finding robust periodic timetables.

3.1. Look-Ahead Approaches for Integrated Planning in Public Transportation

This paper introduces three different improvements to finding public transport systems with reduced operational cost. A public transport system is hereby viewed as a triple $(\mathcal{L}, \pi, \mathcal{R})$, consisting of a line plan $\mathcal{L}$, a timetable π, and vehicle schedule $\mathcal{R}$. The first part of the paper describes which models are used to obtain these three objects by following the planning sequence from Figure 1. Afterwards, three different "look-ahead" heuristics are proposed in order to improve the costs of the resulting public transport system. In the last part of the paper, the quality of the proposed improvements is emphasized via computational experiments.

Traditional Sequential Planning

Given a PTN (V, E) and frequency bounds $f_e^{\min}, f_e^{\max}$ for every edge $e \in E$, a line concept has to satisfy (2.1). Equipped with an objective function and only allowing frequencies in $\{0, 1\}$, a feasible line concept can hence be found by solving

$$\begin{aligned} \min_{f \in \{0,1\}^{|\mathcal{L}^0|}} \quad & \sum_{l \in \mathcal{L}^0} \text{cost}_l f_l, && \text{(LP)} \\ \text{s.t.} \quad & f_e^{min} \leq \sum_{l \in \mathcal{L}^0 : e \in l} f_l \leq f_e^{max} && \forall e \in E, \end{aligned}$$

with a given line pool $\mathcal{L}^0$. The EAN $(\mathcal{E}, \mathcal{A})$ is constructed as described in Section 2.1 and therewith a periodic timetable π is found via solving

$$\begin{aligned} \min_{\pi \in \mathbb{N}^{|\mathcal{E}|}} \quad & \sum_{a=(i,j)\in\mathcal{A}} w_a \left((\pi_j - \pi_i - L_a) \bmod T + L_a\right) \\ \text{s.t.} \quad & L_a \leq (\pi_j - \pi_i - L_a) \bmod T + L_a \leq U_a \quad \forall a = (i,j) \in \mathcal{A}. \end{aligned} \tag{PESP}$$

After rolling out the timetable and creating trips $\mathcal{T}$ the vehicle schedule $\mathcal{R}$ is obtained by using a flow-based model (VS) presented in Bunte and Kliewer (2009). Hence, the public transport system $(\mathcal{L}, \pi, \mathcal{R})$ is constructed as follows:

get $\mathcal{L}$ via (LP) $\underset{\text{construct EAN}}{\longrightarrow}$ get π via (PESP) $\underset{\text{construct trips}}{\longrightarrow}$ get $\mathcal{R}$ via (VS)

Figure 3: Obtaining a Public Transport System

For evaluating the public transport system $(\mathcal{L}, \pi, \mathcal{R})$ we consider two different objective functions which model operational cost and passenger travel time.

$$\begin{aligned} g^{\text{cost}}(\mathcal{L}, \pi, \mathcal{R}) \coloneqq\ & 2p \sum_{l\in\mathcal{L}} (c_1\ \text{dur}_l + c_2\ \text{length}_l) + c_3|\mathcal{R}| \\ & + \sum_{r\in\mathcal{R}} \sum_{i=1}^{n_r} \left(c_4\ \text{dur}_{t_i,t_{i+1}} + c_5\ \text{length}_{t_i,t_{i+1}}\right) \end{aligned} \tag{3.1}$$

$$g^{\text{time}}(\pi) \coloneqq \sum_{a=(i,j)\in\mathcal{A}} w_a \left((\pi_j - \pi_i - L_a) \bmod T + L_a\right) + \sum_{a\in\mathcal{A}^{\text{change}}} w_a \text{pen}. \tag{3.2}$$

The cost g^{cost} of a public transport system is given by several components: the cost of the length and duration of all lines (factored by $2p$ because of p time periods of the rollout and times 2 for forward and backward direction of a line), the cost of the the number of required vehicles $|\mathcal{R}|$, and the cost of the length and duration of empty rides between two trips (i.e., lines) induced by the vehicle schedule. The second objective g^{time} models passenger travel times by summing up the weighted duration of all activities $a \in \mathcal{A}$ and additionally penalizing every passenger transfer by pen≥ 0.

Look-Ahead Improvements

When a public transport system is found using the sequence in Figure 3, the resulting travel times for passengers are observed to be fairly good, but the cost-aspect needs to be improved. Hence, three ideas are proposed to improve the cost of a public

transport system by looking ahead. "Looking ahead" means that the objective the public transport system will be evaluated by is already kept in mind in earlier planning stages.

Improved Line Costs

When finding a line concept in (LP) the costs cost_l of a line l are defined according to Schöbel (2012) as $\text{cost}_{\text{fix}} + \text{cost}_{\text{length}} \cdot \text{length}_l$, i.e., some fixed costs plus some costs that are linear in the length of the line. The function g^{cost}, however, evaluates the costs of a line differently. Thus, the first proposed improvement is to change the parameters cost_l to coefficients that approximate the incurring cost of a line as specified in (3.1). To this end, we assume that each vehicle of the vehicle schedule executes a single line alternately in forward and backward direction, which is called *line-pure vehicle schedule*. Consequently, for every line l we can estimate the cost contribution to g^{cost} by

$$\text{cost}_l := 2(c_1 \text{dur}_l + c_2 \text{length}_l) + \frac{c_3}{p} \left\lceil 2\frac{\text{dur}_l + L^{\text{turn}}}{T} \right\rceil + c_4 \left(T - 2\text{dur}_l \bmod T\right).$$

The first two summands measure costs induced by duration and length of the line in forward and backward direction. The third summand estimates how many vehicles are required to serve the line (with L^{turn} being the minimum time a vehicle has to wait at the end of a line and T being the time period), and the last summand estimates how many minutes the vehicle has to wait at the end of a line in order to satisfy the periodicity of the timetable. The costs c_5 of g^{cost} do not occur here as no vehicle has to travel to a different station to start the next trip.

New Line Pool

The line pool $\mathcal{L}^0$ used in (LP) is found by tree-based heuristics from Gattermann et al. (2017). We want to improve this procedure by considering only lines that result in cost-efficient line-pure vehicle schedules. To achieve this, we require for the duration of line l to satisfy

$$2\left(\text{dur}_l + L^{\text{turn}}\right) \bmod T \in [T - 2\alpha, T]$$

with some buffer time $\alpha \in [0, \frac{T}{2}]$. By requiring the total time of the vehicle schedule to be close to a multiple of T, we keep the waiting time of a vehicle between two trips low. Additionally, for every line satisfying this property, it can be ensured that the number of vehicles required to serve this line is kept low, even for a line-pure vehicle schedule. This is the case because the waiting time of a vehicle between two trips is bounded by $2(\alpha + L^{\text{turn}})$.

Vehicle Scheduling First

The third improvement changes the order of Figure 3 by inserting a preliminary vehicle scheduling step after the line planning step. The reason for this change is that the timetabling step potentially eliminates cost-efficient vehicle schedules due to its passenger-oriented objective. Preliminary vehicle scheduling consists of adding turnaround activities to the EAN that connect the last event of a line in forward direction with the first event of the same line in backward direction. We additionally require bounds on the duration of the turnaround activities of $[L^{\text{turn}}, L^{\text{turn}} + 2\alpha]$. These bounds ensure short waiting times at the end of lines and hence prevent additional costs. A timetable found by merely solving (PESP) would not consider the benefit of short turnaround times since (PESP) optimizes passenger travel times. This improvement is in particular beneficial if the lines have a small α (from the second improvement).

Computational Results

We implemented the three proposed improvements in LinTim (see Schiewe et al., 2018) and tested them on the datasets *bahn* (German ICE network) and *grid* (a bus network with 25 stops, see DFG Research Unit FOR 2083, 2019) and retrieved the results for grid given in Figure 4.

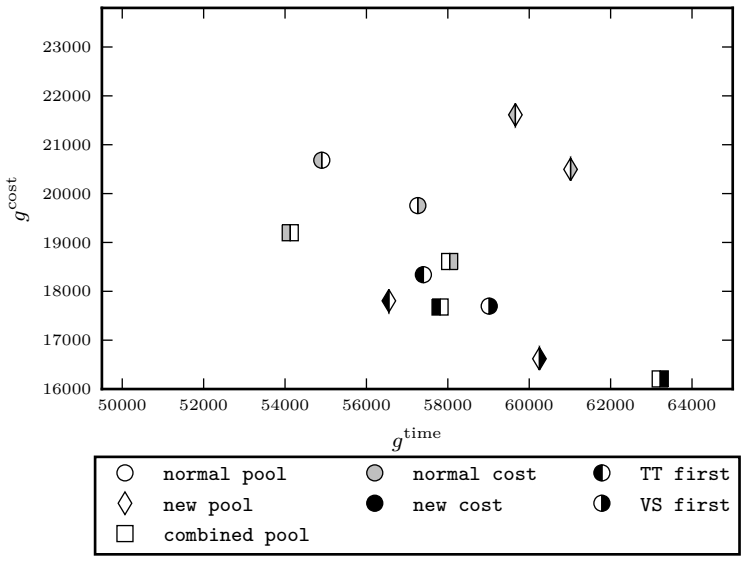


Figure 4: Results for grid

The clearest observation is the effect of the improvement "Vehicle Scheduling First": If switched on (`VS first`), the cost of the solution increases in comparison to when switched off (`TT first`). In this case, however, the passenger travel times are better. For the improvement "Improved Line Costs" it can be seen that, when switched on (`new cost`) the solutions (colored in black) lie below the solutions when switched off (`normal cost`, grey colored), which means they have lower operational cost. For the improvement "New Line Pool" we retrieve the best results if (`combined pool`) is used, i.e., if normal and new line pool are combined into one large line pool. The new line pool itself already leads to lower costs when combined with (`new cost`). Thus, in summary, all three ideas improve the operational cost of a public transport system and hence work as initially intended.

3.2. Cost-Minimal Public Transport Planning

As opposed to the previous publication that introduces heuristics to reduce the costs of a public transport system, this publication takes one step further and presents models for cost-minimal solutions. To this end, we first state how a public transport system is defined here (which slightly differs from the last contribution) and then present models for finding a cost-minimal public transport system.

Planning a Public Transport System

We again consider the three planning steps of line planning, timetabling and vehicle scheduling, but make two important changes in the definition of a public transport system.

First, for obtaining a line concept the frequency bounds $f_e^{\min}$ are assumed not to be given anymore. Instead, we consider the requirements according to which they were constructed in Section 2.1. Minimum frequencies are found by distributing a passenger load from a given OD-matrix $W \in \mathbb{R}^{|V|\times|V|}$ to every edge in the PTN (V, E). A line concept is then defined to be feasible if all passengers are able to travel from their origin to their destination. Formally, a line concept $\mathcal{L}$ is a set of simple paths l that is feasible if for every pair $(u, v) \in V \times V$ there exists a set of directed paths P_{uv} from u to v in the PTN, $P_{\text{all}} = \bigcup_{u,v\in V} P_{uv}$ and weights w_p for each path $p \in P_{uv}$ with $\sum_{p\in P_{uv}} w_p = W_{uv}$ such that

$$\sum_{p\in P_{\text{all}}:e\in p} w_p \leq \text{Cap} \cdot |\{l \in \mathcal{L} : e \in l\}| \quad \forall e \in E, \tag{3.3}$$

with Cap being the same passenger capacity for every vehicle.

The second difference to Section 3.1 is that we consider periodic vehicle scheduling. A vehicle schedule $\mathcal{R}$ is hence a set of rides r such that every directed line from the set of trips (the set of all lines converted into directed lines in forward and backward direction) is contained in exactly one ride r. Consequently, no rollout of the timetable is required since a vehicle schedule can be determined by adding turnaround activities to the periodic EAN. We additionally assume that changing activities in the EAN have no upper bounds and also that no other activities exist, except driving and waiting activities. Under these assumptions there always exists a feasible timetable for the EAN, even with the added turnaround activities. This provides motivation to neglect timetable π and EAN as separate objects in our planning process since the main concern is cost-minimization.

Hence, we define the problem of finding a cost-minimal public transport system $(\mathcal{L}, \mathcal{R})$ as

$$\min_{\mathcal{L},\mathcal{R}} g(\mathcal{L}, \mathcal{R}) := c_{\text{time}}\text{dur}(\mathcal{L}, \mathcal{R}) + c_{\text{length}}\text{length}(\mathcal{L}, \mathcal{R}), \tag{cost-opt}$$

with

$$\mathrm{dur}(\mathcal{L},\mathcal{R}) \coloneqq \sum_{r\in\mathcal{R}} \left\lceil \sum_{i=1}^{n_r} \mathrm{dur}_{l'_i} + \mathrm{dur}_{l'_i,l'_{i+1}} \right\rceil_T$$

$$\mathrm{length}(\mathcal{L},\mathcal{R}) \coloneqq \sum_{r\in\mathcal{R}} \sum_{i=1}^{n_r} \mathrm{length}_{l'_i} + \mathrm{length}_{l'_i,l'_{i+1}}$$

where the $l'_i \in \mathcal{L}'$ are the trips induced by the line concept $\mathcal{L}$, cf. Section 2.1. Again, a ride $r = (l'_1, \ldots, l'_{n_r}) \in \mathcal{R}$ is a sequence of trips such that – due to periodicity of the vehicle scheduling – trip l'_1 is also executed after trip l'_{n_r}, meaning that $n_r + 1 = 1$ in the definition of $\mathrm{dur}(\mathcal{L},\mathcal{R})$. Furthermore, $\lceil x \rceil_T$ is defined as rounding up $x \in \mathbb{R}$ to the next multiple of T. This rounding up is necessary since every (periodic) vehicle ride $r \in \mathcal{R}$ needs to be executed periodically with time period T and hence the duration of a ride r needs to be a multiple of T. In general, g can be viewed as a modified version of g^{cost} from Section 3.1. Costs of a vehicle for riding trips or riding between two trips are assumed to be equal ($c_1 = c_4$ and $c_2 = c_5$ for the cost coefficients c_i from g^{cost}). Additionally, the number of required vehicles is determined by $\frac{\mathrm{dur}(\mathcal{L},\mathcal{R})}{T}$ and hence the costs c_3 of g^{cost} are assumed to be contained in c_{time}.

For this definition of a public transport system we propose three models that are capable of solving (cost-opt) to optimality. The difference between the three models consists in the additional assumptions made under which an cost-optimal public transport system can be obtained:

- Model 1 gives a lower bound of (cost-opt). Solutions of a modified version (Model 1*) can be extended to feasible solutions of (cost-opt) and hence yield an upper bound. If $L^{\mathrm{turn}} = L^{\mathrm{wait}}$ (i.e., minimal turnaround time equals minimal waiting time of vehicles), or if lines are not simple and do not have to be operated in both directions, Model 1* solves (cost-opt) to optimality.
- Model 2 extends Model 1 and gives a stronger lower bound of (cost-opt). A modified version (Model 2*) yields an upper bound to (cost-opt) and solves it to optimality if only line-pure vehicle schedules (i.e., vehicles operate only one line by alternating in forward and backward direction) are allowed.
- Model 3 solves (cost-opt) to optimality under no additional assumptions.

Model 1: Cost-Efficient Load Generation

The crucial component for the first proposed model is the load of a line concept $\mathcal{L}$, also known as the number how often every edge $e \in E$ is covered by the line concept, i.e.,

$$f_e \coloneqq |\{l \in \mathcal{L} \ : \ e \in l\}|.$$

On the one hand, these $(f_e)_{e\in E}$ determine the feasibility of the line concept according to (3.3). On the other hand, they give an estimate on $g(\mathcal{L},\mathcal{R})$ since

$$g(\mathcal{L},\mathcal{R}) \approx c_{\text{time}} \left\lceil \sum_{e\in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) f_e \right\rceil_T + c_{\text{length}} \sum_{e\in E} 2\text{length}_e f_e. \tag{3.4}$$

L_e^{drive} denotes the minimal duration for a vehicle driving edge $e \in E$ and L^{wait} denotes the minimal waiting time for every vehicle at every station. Hence, g is approximated in (3.4) by two summands. The first summand estimates the minimal amount of time required of any line concept to serve every edge $e \in E$ at least f_e times. The second summand estimates the minimal length of serving every edge $e \in E$ at least f_e times.

In the first model we now find a load $(f_e)_{e\in E}$ that satisfies (3.3), which is achieved by modeling the passenger paths as a multi-commodity flow. Moreover, Model 1 also minimizes the estimation of (3.4). To this end, define the flow variables $f_{(u,v),w}$ denoting how many passengers – originating from $w \in V$ – travel along edge $u, v \in E$ from u to v. Model 1 is then given by

$$\begin{aligned}
\min_{f} \quad & c_{\text{time}}\text{dur} + c_{\text{length}} \sum_{e\in E} 2\text{length}_e f_e && && \text{(Model 1)}\\
\text{s.t.} \quad & \sum_{e\in E} 2f_e(L_e^{\text{drive}} + L^{\text{wait}}) \le T\text{dur} && &&\\
& \sum_{u\in V} f_{(i,j),u} \le f_e \cdot \text{Cap} && && \forall e=\{i,j\}\in E\\
& \sum_{i\in V:\{i,v\}\in E} f_{(i,v),u} = W_{uv} + \sum_{i\in V:\{v,i\}\in E} f_{(v,i),u} && && \forall u\in V\ , v\in V\backslash\{u\}\\
& \sum_{i\in V:\{u,i\}\in E} f_{(u,i),u} = \sum_{v\in V} W_{uv} && && \forall u\in V\\
& f_{(u,v),w}, f_e, \text{dur} \ge 0 && && \forall w\in V, \{u,v\}, e\in E
\end{aligned}$$

Hence, Model 1 finds frequencies $(f_e)_{e\in E}$ that allow a feasible passenger flow required in (3.3) and minimize (3.4). In Theorem 4 of the publication it is shown that Model 1 yields a lower bound on (cost-opt).

In addition to Model 1, we define Model 1* by replacing L^{wait} with L^{turn}. Theorem 7 and Corollaries 10 and 11 show that with an optimal solution $(f_e)_{e\in E}$ to Model 1*, it is possible to obtain a cost-minimal public transport system under two different assumptions: Either $L^{\text{turn}} = L^{\text{wait}}$ needs to hold, i.e., the minimum time a vehicle has to wait after serving a line equals the time a vehicle has to wait at a station while serving a line, or the requirement of lines being simple paths and operated in both directions is dropped.

Model 2: Integrating Load Generation and Line Planning

Without the assumption $L^{\text{turn}} = L^{\text{wait}}$ and with the requirement that lines need to be simple paths operated in both directions, a more detailed model is necessary, which captures the line planning stage correctly. Hence, Model 2 is proposed as a linearized version of the minimization problem

$$\begin{aligned} \min_{f,\mathcal{L}} \quad & \sum_{l\in\mathcal{L}} 2c_{\text{time}}\text{dur}_l + 2c_{\text{length}}\text{length}_l && \text{(Model 2)} \\ \text{s.t.} \quad & f \text{ feasible w.r.t. Model 1,} \\ & l \text{ simple path in PTN} && \forall l \in \mathcal{L}, \\ & f_e \leq |\{l \in \mathcal{L} : e \in l\}| && \forall e \in E. \end{aligned}$$

In the publication, it is shown in Theorem 13 that Model 2 yields a stronger relaxation of (cost-opt) than Model 1. After defining Model 2* by replacing 2dur_l with $\lceil 2\text{dur}_l \rceil_T$, Theorem 16 then shows that an optimal line concept $\mathcal{L}$ of Model 2* can be extended to an optimal solution to (cost-opt) if only line-pure vehicle schedules are allowed.

Model 3: Integrating Vehicle Scheduling as well

In order to achieve a model that finds a cost-minimal public transport system without any of the above-mentioned assumptions, the vehicle scheduling stage needs to be included into the model. To this end, every trip $l' \in \mathcal{L}'$, which is created by converting every $l \in \mathcal{L}$ into two trips (corresponding to its forward and backward direction), needs to be assigned to exactly one ride $r \in \mathcal{R}$. Hence, Model 3 is a linearized version of

$$\begin{aligned} \min_{f,\mathcal{L},\mathcal{R}} \quad & \sum_{r\in\mathcal{R}} c_{\text{time}}\text{dur}_r + c_{\text{length}}\text{length}_r && \text{(Model 3)} \\ \text{s.t.} \quad & f \text{ feasible w.r.t. Model 1/2,} \\ & \mathcal{L} \text{ feasible w.r.t. Model 2,} \\ & |\{r \in \mathcal{R} : l' \in r\}| = 1 && \forall l' \in \mathcal{L}'. \end{aligned}$$

In Theorem 23 we show that Model 3 solves (cost-opt) to optimality without any further assumptions.

Computational Results

We implemented all three models and tested them on several instances from LinTim (cf. Schiewe et al., 2018). Next to *grid* and *bahn*, which are already introduced in

Section 3.1, we run the implementations on *linear*, a linear network with 4 stops, and *toy*, a small network with 8 vertices and 8 edges between them.

Instance	Model 1		Model 2		Model 3	
	Model 1	Model 1*	Model 2	Model 2*	lb	ub
linear	80	130	130	130	130	130
toy	1424	1474	1424	1696	1288°	1539°
grid	1034	1134	1030°	1140	–	–
bahn	74462°	85612°	54148°	–	–	–

Table 2.1.: Objective values for instances with $L^{\text{turn}} > L^{\text{wait}}$. ∘ means time out after three computing hours with no optimal solution

For each model the left column constitutes a lower bound on (cost-opt) and the right column constitutes an upper bound on (cost-opt). From the number of solved instances we infer that the models increase in intricacy. Models 1 and 1* are clearly the fastest (since smallest) models since they provide solutions for all instances. While Model 1 provides lower bounds, Model 1* computes the best solutions for all instances. Finally, for all instances up to *grid* (if $L^{\text{turn}} = L^{\text{wait}}$) Model 1* even computes cost-optimal solutions, which outperforms previous approaches to tackle this problem (see DFG Research Unit FOR 2083, 2019). Model 2 still gives solutions to all instances up to grid, but for the large *bahn* instance it is only able to provide a lower bound. Unfortunately, Model 3 is only able to provide bounds for *toy*. Nevertheless, for *linear* it can be seen that the objective values of the three models can coincide and that Model 2 provides a stronger lower bound than Model 1.

Overall, the proposed models are able to provide optimal solutions to (cost-opt) under different but arguably slight assumptions, while, at the same time, being computationally tractable even for large-scale instances.

3.3. The Trickle-in Effect: Modeling Passenger Behavior in Delay Management

This publication considers the public transport planning problem of delay management and introduces a new effect worth respecting when modeling delay management, the *trickle-in effect*. The trickle-in effect describes an observation of passenger behavior at train stations, namely that the transfer time from one train to another is not equal for all passengers but rather lies in some interval. In other words, passengers do not switch trains instantaneously but instead trickle in to board the train. Within this trickling interval, i.e., while passengers are boarding, a train is not able to depart from a station which may lead to unexpected delays.

Classical Delay Management

Traditionally (see, e.g., Schöbel, 2007) the delay management problem is defined on an aperiodic EAN $(\mathcal{E}, \mathcal{A})$ together with a timetable $\pi \in \mathbb{N}^{|\mathcal{E}|}$ and some source delays $s \in \mathbb{R}_{\geq 0}^{|\mathcal{E}|+|\mathcal{A}|}$. Delay management provides a disposition timetable $x \in \mathbb{N}^{|\mathcal{E}|}$ that aims at reducing the overall passenger delay. A mathematical model for obtaining a disposition timetable while minimizing (approximated) passenger delays, as introduced in Schöbel (2007), then reads as

$$\min \quad \sum_{i \in \mathcal{E}^{\text{arr}}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} y_a w_a T \tag{DM}$$

$$\text{s.t.} \quad x_i \geq \pi_i + s_i \qquad \forall i \in \mathcal{E}, \tag{3.5}$$

$$x_j - x_i \geq L_a + s_a \qquad \forall a = (i,j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}, \tag{3.6}$$

$$M y_a + x_j - x_i \geq L_a \qquad \forall a = (i,j) \in \mathcal{A}_{\text{change}}, \tag{3.7}$$

$$x_i \in \mathbb{R} \qquad \forall i \in \mathcal{E},$$

$$y_a \in \{0,1\} \qquad \forall a \in \mathcal{A}_{\text{change}},$$

with w_i being the number of passengers unboarding at event i and some big $M \in \mathbb{R}$. Furthermore, the binary variables y_a decide if a connection $a = (i,j) \in \mathcal{A}_{\text{change}}$ is maintained or if the connecting train departs (event j) before the passengers from the feeder train have arrived (event i).

Modeling the Trickle-in Effect

For a disposition timetable that is found by solving (DM), it can be the case, however, that some train B is scheduled to depart soon after train A has arrived such that train B is not supposed to wait for passengers from train A. An example of such

scenario is the following: train B is scheduled to depart at 10:02, train A arrives at 10:00 and it is assumed that passengers need at least 3 minutes to transfer from train A to train B. Now, if a fast passenger from train A reaches train B, then there exists the possibility that slower passengers from train A might also reach train B since the doors are prevented to close due to the boarding of the (fast) passenger. This effect might result in a stream of passengers boarding train B and preventing it from departing at 10:02, such that train B is only able to depart at 10:05. This effect is called the trickle-in effect and the time interval $(L_a^{\min}, L_a^{\max})$, denoting the changing duration for all passengers (here, e.g., $(1, 5)$ minutes) is called *trickling interval*.

In order to model the trickle-in effect we require that the train departure cannot lie in the trickling interval for all departure events. Hence,

$$x_j \notin (x_i + L_a^{\min}, x_i + L_a^{\max}) \quad \forall a \in \mathcal{A}_{\text{change}}. \tag{3.8}$$

To achieve this, we replace constraint (3.6) in (DM) by two new constraints, i.e.,

$$My_a + x_j - x_i \geq L_a^{\max}, \tag{3.9}$$

$$M(y_a - 1) + x_j - x_i \leq L_a^{\min}. \tag{3.10}$$

The new model is then called (DM-trick). In our contribution we show that (DM-trick) in fact satisfies the requirement (3.8) in Lemma 1. Furthermore, in Lemma 2 and 3 we give bounds on the choice of big M for (DM-trick) for different structures of the source delays.

Finally, it is shown in Lemma 4 that for two trickling intervals $I_1 \subseteq I_2 \subset \mathbb{R}$, the for the objective values z_1 and z_2 of (DM-trick) with I_1 and (DM-trick) with I_2 it holds that $z_1 \leq z_2$. Due to this fact, (DM) is a relaxation of (DM-trick) and Lemma 5 shows that a choice of $L_a = L_a^{\max}$ yields the strongest relaxation (DM) of (DM-trick).

Computational Results

We investigate the dataset *bahn* (modeling the German ICE network) for different trickling intervals. In Figure 5 one can observe that the objective value, which is given in the overall passenger delay in seconds, of (DM-trick) increases with an increasing size of the trickling interval. Furthermore an increase in $L_a^{\max}$ results in a distinct increase in the objective value of (DM-trick), whereas a decrease in $L_a^{\min}$ only slightly increases the objective value.

In Figure 6 (DM)'s performance as a relaxation for (DM-trick) (with a trickling interval of $[60, 300]$ seconds) is shown. The result underlines Lemma 5, i.e., that (DM) with $L_a = L_a^{\max}$ is the best approximation of (DM-trick), and especially the computation times (1 second for DM vs. 72 seconds for DM-trick) underline that

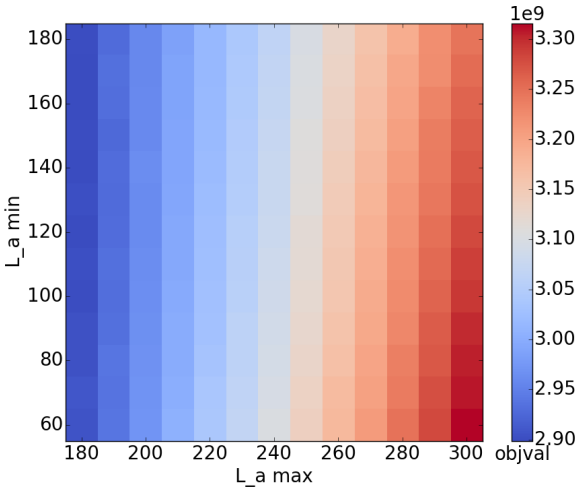


Figure 5: objective values for (DM-trick) for different trickling intervals (in seconds)

for obtaining a fast and still reasonably good approximation it can make sense to solve (DM) instead of (DM-trick).

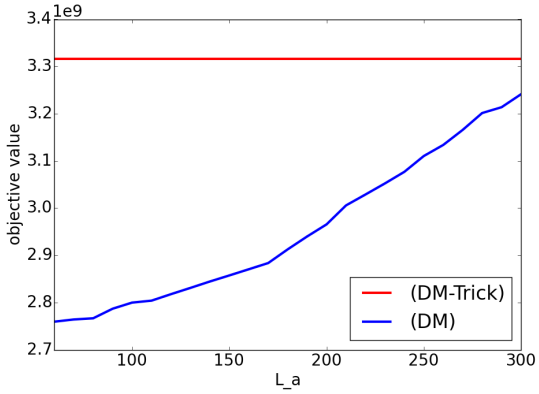


Figure 6: objective values for (DM) for different L_a

Finally, further experiments in the publication show that even without any source delays the objective of (DM-trick) is not zero. This can be explained by the fact that the duration of some changing activities lies in the trickling interval, which, as a result, have to be rescheduled. Thus, the trickle-in effect implies constraints that already need to be respected in the timetabling stage. It thus becomes evident that modeling the trickle-in effect is a highly relevant aspect for delay management models.

3.4. Approximate Cutting Plane Approaches for Exact Solutions to Robust Optimization Problems

In this publication we consider the concept of strict robustness and present speed-up techniques for the cutting plane method described in Section 2.2. We give correctness and convergence results for the proposed solution algorithm. In the computational experiments, we apply the speed-up techniques for the class of mixed-integer optimization problems with uncertain objective function. The results show that the use of approximate cutting plane techniques can be very beneficial.

Cutting Planes for Strict Robustness

We define the strictly robust counterpart of a minimization problem, which suffers from uncertainty only in the objective function g, as

$$\min_{x \in X} \sup_{u \in \mathcal{U}} g(x, u), \tag{RC($\mathcal{U}$)}$$

for some uncertainty set $\mathcal{U} \subseteq R^n$ and a set of feasible solutions $X \subseteq \mathbb{R}^n$. For a given solution $x^* \in X$ a worst-case scenario with respect to uncertainty set $\mathcal{U}$ is found by solving

$$g_{\mathcal{U}}(x^*) \coloneqq \sup_{u \in \mathcal{U}} g(x^*, u). \tag{Pes(x^*)}$$

With this notation the basic scheme of a cutting plane approach can be formulated as depicted in Figure 7. We call the process of finding a new solution x_k robustification step and the process of finding a new scenario to u_k some solution pessimization step.

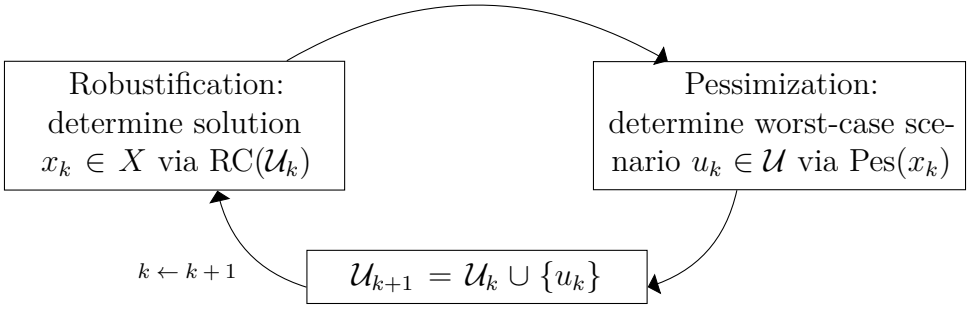


Figure 7: Basic Scheme of a cutting plane approach

Approximate Cutting Planes

The contribution of this paper is the proposal of not solving the subproblems RC and Pes in each iteration exactly, i.e., to optimality, but only approximatively. To this end, define for a given problem RC($\mathcal{U}'$) and some threshold $t \in \mathbb{R}$ the approximated version of RC($\mathcal{U}'$) to return any solution $x \in X$ with $g_{\mathcal{U}'}(x) \leq t$. If no such solution exists, it returns an optimal solution to RC($\mathcal{U}'$). Analogously, define for a given problem Pes(x) and some threshold $t \in \mathbb{R}$ the approximated version of Pes(x) to return any scenario $u \in \mathcal{U}$ with $g(x, u) \geq t$. If no such scenario exists, a worst-case scenario to x, i.e., an optimal solution to Pes(x) is returned.

A-RC($\mathcal{U}', t$)	Return some $x \in X$ with $g_{\mathcal{U}'}(x) \leq t$. If such an x does not exist, return $x \in X$ optimal to RC($\mathcal{U}'$)
A-Pes(x, t)	Return some $u \in \mathcal{U}$ with $g(x, u) \geq t$. If such a u does not exist, return $u \in \mathcal{U}$ optimal to Pes(x).

With this notion of an approximated version we extend the basic scheme of Figure 7 to a modified version of the cutting plane approach, depicted in Algorithm 1.

The algorithm works similar to a standard cutting plane algorithm. Instead of iterating between RC and Pes, it iterates between A-RC($\mathcal{U}_k, (t_{opt})_k$) and A-Pes($x_k, (t_{pes})_k$) and adds a new scenario u_k to $\mathcal{U}_k$ in each iteration. If A-RC (or A-Pes) is solved to optimality, the lower bound lb_k (or upper bound ub_k) can get updated, because then an optimal solution x_k yields an actual lower bound $g_{\mathcal{U}_k}(x_k)$ to RC($\mathcal{U}$). Vice versa, a worst-case scenario u_k to some solution x_k yields an actual upper bound $g(x_k, u_k)$. The basic idea behind this proposal is that the process of obtaining new solutions and scenarios via A-Pes and A-RC is faster than for RC and Pes. Hence, the goal of Algorithm 1 is to find the optimal solution to RC($\mathcal{U}$) in more iterations with "good enough" cuts and solutions, but in less time since cuts and solutions can be obtained faster.

The correctness of Algorithm 1 is shown in Lemma 2 of the publication. For the convergence results it is assumed that A-RC($\mathcal{U}_k, t$) with finite $|\mathcal{U}_k|$ and A-Pes(x_k, t) are solvable in a finite number of steps and their solutions are always finite. Under this assumption we give convergence results for finite $|X|$ or finite $|\mathcal{U}|$ in Lemmas 3 to 5. Furthermore, Theorems 8 to 10 provide convergence results for infinite $|X|$ and $|\mathcal{U}|$, differing in the assumptions made on the structures of X and $\mathcal{U}$ and the choice of $(t_{opt})_k$ and $(t_{pes})_k$.

Additionally, we introduce three improvements of Algorithm 1 for the case that the underlying nominal optimization problem is of a mixed-integer type. First, the subproblems do not have to be solved from scratch: When providing the IP solver with the current best solution, or worst scenario, respectively, we obtain a warm start. Second, the bounds lb_k and ub_k can be strengthened via incorporating

Algorithm 1: Approximate Cutting Plane Approach

Input: problem (RC), nominal scenario $u_{\text{nom}} \in \mathcal{U}$, stopping criterion $\epsilon > 0$
Output: solution $x \in X$ to RC($\mathcal{U}$), with an absolute deviation from the optimal solution of at most ϵ

$\mathcal{U}_0 \leftarrow \{u_{\text{nom}}\}$, $k \leftarrow 0$, $lb_0 \leftarrow -\infty$, $ub_0 \leftarrow \infty$
while $ub_k - lb_k > \epsilon$ **do**
  determine $(t_{opt})_k$
  $x_k \leftarrow$ solution to A-RC$(k, (t_{opt})_k)$
  if *A-RC*$(\mathcal{U}_k, (t_{opt})_k)$ *solved optimally and* $lb_k < g_{\mathcal{U}_k}(x_k)$ **then**
    $lb_{k+1} \leftarrow g_{\mathcal{U}_k}(x_k)$
  else
    $lb_{k+1} \leftarrow lb_k$
  determine $(t_{pes})_k$
  $u_k \leftarrow$ solution to A-Pes$(x_k, (t_{pes})_k)$
  if *A-Pes*$(x_k, (t_{pes})_k)$ *solved optimally and* $ub_k > g(x_k, u_k)$ **then**
    $ub_{k+1} \leftarrow g(x_k, u_k)$
    $x_{ret} \leftarrow x_k$
  else
    $ub_{k+1} \leftarrow ub_k$
  $\mathcal{U}_{k+1} \leftarrow \mathcal{U}_k \cup \{u_k\}$
  $k \leftarrow k + 1$
end
return x_{ret}

the bounds found by the IP solver. Finally, Algorithm 1 can be transformed to a branch-and-cut scheme by using callback functions. In the publication additional computational experiments are introduced where the improvements are successfully tested.

Computational Results

Algorithm 1 is tested with respect to the class of uncertain mixed-integer optimization problems. Given $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ of the form

$$\begin{aligned} \min \quad & \tau \\ \text{s.t.} \quad & Ax \geq b, \\ & u^t x \leq \tau \qquad \forall u \in \mathcal{U}, \\ & x \in \mathbb{R}^{n-p} \times \mathbb{Z}^p, \\ & \tau \in \mathbb{R}, \end{aligned}$$

with a polyhedral uncertainty set $\mathcal{U} \coloneqq \{u \in \mathbb{R}^{n-q} \times \mathbb{Z}^q | \; Cu \leq d\}$ for $C \in \mathbb{R}^{m \times n}, d \in \mathbb{R}^m$. We distinguish between easy and hard instances of X and $\mathcal{U}$ with respect to the number of integer variables. A choice of $p = 1$ implies that all (except one) of the x-components are continuous variables and hence A-RC (the robustification) is easy to solve – compared to a choice of $p = n$. Similarly, a choice of $q = 1$ makes A-Pes (the pessimization) easy compared to $q = n$. This definition leads to four different instance cases: Robustification easy or hard crossed with pessimization easy or hard.

In the computational results we compared four different versions of Algorithm 1: Solving robustification via RC or A-RC combined with solving pessimization via Pes or A-Pes. Computational results are given in Table 4.2.

	Both easy	Rob hard	Pes hard	Both hard
A-RC + A-Pes	0.482	44.224	379.172	14.922
A-RC + Pes	0.48	45.01	928.298	17.278
RC + A-Pes	0.474	100.574	379.122	21.09
RC + Pes	0.474	98.102	929.43	21.2

Table 4.2.: Average runtime in seconds for solving five instances with $n = 20$ variables to optimality.

It can be seen that A-RC works faster than RC if only robustification is difficult and A-Pes works faster than Pes if only pessimization is difficult. When both problems are easy all instances are solved with similar speed, whereas for instances with both problems being difficult, A-RC + A-Pes yields the best results. The publication consists of further computational experiments, including the effect of the above-mentioned improvements and the effect of different choices on $(t_{opt})_k$ and $(t_{pes})_k$.

In conclusion, this publication introduces an improved scheme of the cutting plane methods used in robust optimization and gives several convergence results. In addition the computational results illustrate that the use of its scheme can significantly decrease computation times of strictly robust mixed-integer problems.

3.5. Finding Robust Periodic Timetables by Integrating Delay Management

This publication investigates the problem of finding robust periodic timetables. To this end, a model for periodic delay management (P-DM) is introduced that enables evaluating periodic timetables with respect to their delay resistance. By combining (P-DM) and (PESP), an adjustable robust version of (PESP) is introduced, called (RPT), that is used for finding robust periodic timetables. Two different solution approaches to (RPT) are presented and computationally investigated.

Evaluation of a Periodic Timetable

As described in Section 2.1 and defined in Section 3.1, (PESP) is a commonly used model for periodic timetable optimization. Its solutions are optimized with respect to the nominal travel times of passengers, i.e., in the absence of any delays. The performance of a timetable in practice, however, is highly influenced by regularly occuring delays. Hence, instead of finding a timetable with respect to the objective of (PESP), we follow the scheme of Figure 8 in order to find and evaluate robust timetables.

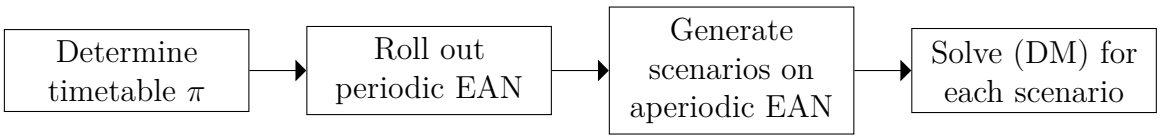


Figure 8: Workflow for finding and evaluating timetables

In order to find a periodic timetable with good performance with respect to the above evaluation, periodic delay management is defined. For source delays $s \in \mathbb{R}_{\geq 0}^{|\mathcal{E}|+|\mathcal{A}|}$ and a given timetable π, define

$$\begin{aligned}
\min \quad & \sum_{a\in\mathcal{A}} w_a d_a + \sum_{i\in\mathcal{E}^{\text{dep}}} w_i d_i && && \text{(P-DM)}\\
\text{s.t.} \quad & d_i \geq s_i && \forall i \in \mathcal{E}, && (3.11)\\
& d_a = d_j - d_i && \forall a = (i,j) \in \mathcal{A}\backslash A_{\text{change}}, && (3.12)\\
& d_a = d_j - d_i + z_a T && \forall a = (i,j) \in \mathcal{A}_{\text{change}}, && (3.13)\\
& \pi_a + d_a \geq L_a + s_a && \forall a = (i,j) \in \mathcal{A}, && (3.14)\\
& d_i \in \mathbb{R} && \forall i \in \mathcal{E} \cup \mathcal{A},\\
& z_a \in \mathbb{Z} && \forall a \in \mathcal{A}_{\text{change}},
\end{aligned}$$

with the set of all feasible propagated delays d being

$$\mathcal{D} = \mathcal{D}(\pi, s) \coloneqq \{d \in \mathbb{R}^{|\mathcal{E}|+|\mathcal{A}|} \mid \exists\, z \in \mathbb{Z}^{|\mathcal{A}_{\text{change}}|} \text{ s.th. } (d, z) \text{ is feasible for (P-DM)}\}.$$

(P-DM) can be viewed as a modification of (DM), which is defined in Section 3.3. The disposition timetable x from (DM) is given here as $(\pi_i + d_i)_{i\in\mathcal{E}}$. Furthermore, the d_i for $i \in \mathcal{E}$ are bounded from below by s_i and the d_a are bounded from below by $L_a + s_a - \pi_a$, where $\pi_a \coloneqq (\pi_j - \pi_i - L_a) \bmod T + L_a$ is the duration of activity $a = (i, j) \in \mathcal{A}$. One difference of (P-DM) in comparison to (DM) is that the passenger delays are counted as the delay of their departure event $\mathcal{E}^{\text{dep}}$ plus the sum of each activity delay they are passing, and not as the delay of the last arrival event of the passenger path (as done in DM). Furthermore, since the delays occur on a periodic EAN, also the delay management decisions work periodically which differs to the model (DM). Note that the delays d_a in (P-DM) are propagated along driving and waiting activities, but for all changing activities a modulo T operation is executed via constraint (3.13). In the following (P-DM) is used to estimate the behavior of a periodic timetable π when rolled out and used for aperiodic delay management as in Figure 8. Hence, (P-DM) can be utilized to find robust periodic timetables.

Robust Periodic Timetabling

By using (P-DM), the roll-out step in Figure 8 can be neglected and thus the problem of finding a delay resistant timetable reduces to the three steps: timetabling, scenario generation, and periodic delay management. Consequently, the concept of adjustable robustness (cf. Adjustable RC) can be used to define a robust timetabling problem. To this end, let Π be the set of all feasible timetables with respect to (PESP) for a given EAN. Then, define

$$\min_{\pi\in\Pi} \sup_{s\in\mathcal{S}} \min_{d\in\mathcal{D}(\pi,s)} \tau(\pi, d) \tag{RPT}$$

with

$$\tau(\pi, d) \coloneqq \sum_{a\in\mathcal{A}} w_a(\pi_a + d_a) + \sum_{i\in\mathcal{E}^{\text{dep}}} w_i d_i,$$

and some uncertainty set $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{E}|+|\mathcal{A}|}_{\geq 0}$. Hence, the timetable $\pi \in \Pi$ denotes the here-and-now variables and the disposition timetable corresponds to the wait-and-see variables of the adjustable robust counterpart. Due to the immense difficulty of solving (RPT) directly, we propose two different simplifications of the model.

Simplification A: Fixed Delay Management Strategy (F-RPT)

One simplification is to require that the delay management strategy is determined by timetable π and scenario s. For this purpose, a strict no-wait policy for trains is assumed, but also any other strategy can be used, as long as it can be expressed in terms of π and s. With this fixed delay management strategy the inner max-min problem of finding the worst-case scenario for a given timetable is transformed into a mixed-integer maximization problem. Hence, (RPT) reduces to a min-max problem that is solvable by the cutting-planes approach for robust optimization problems, as introduced in Section 3.4. For this first simplification we show convergence of the proposed solution algorithm to the optimal solution of the simplified model, which is called F-RPT.

Simplification B: Finding Bad-Case Scenarios (RPT($\mathcal{S}'$))

A different simplification is to refrain from searching for a worst-case scenario s for a given timetable π, and instead focus on finding a bad scenario heuristically. This is done by randomly sampling scenarios from $\mathcal{S}$ and taking the worst-case scenario of this smaller subset with respect to (P-DM). This bad case scenario is added to the set of considered scenarios $\mathcal{S}'$ and an improved timetable can then be found by solving (RPT) on $\mathcal{S}'$. Thus, it can again be iterated between timetabling and scenario generation until some stopping criterion is met. The second simplification serves as a heuristic algorithm, called RPT($\mathcal{S}'$).

Computational Results

In the computational experiments the two proposed algorithms F-RPT and RPT($\mathcal{S}'$) are compared against MATCH (cf. Pätzold and Schöbel, 2016), an algorithm for solving (PESP). The algorithms are tested on the three instances *toy*, *grid* and *bahn*. Their nominal travel times are compared against delayed travel times from the disposition timetable obtained after delay management, i.e., evaluation by following Figure 8. The delayed times are averaged for 10 different randomly generated delay scenarios.

	MATCH	F-RPT	RPT($\mathcal{S}'$)
toy	6.0	9.1	7.1
grid	20.4	24.6	24.1
bahn	166.4	177.6	185.8

Table 5.3.: Nominal travel times in minutes

	MATCH	F-RPT	RPT($\mathcal{S}'$)
toy	13.4	11.3	11.0
grid	29.9	27.3	28.3
bahn	205.9	204.6	202.1

Table 5.4.: Average delayed travel times in minutes

It is observable that MATCH performs best for nominal travel times, whereas both of the proposed algorithms yield better results than MATCH for the average delay objective. Importantly, the source delays in the scenarios were chosen to be quite small for every activity and hence impose rather small perturbations of the public transport system. Nevertheless, these small delays induce high delays in passenger travel time for MATCH (up to 40 minutes on average for *bahn*), whereas the timetables found by F-RPT and RPT($\mathcal{S}'$) appear to be more robust against the source delays: here, the average delay for *bahn* is 16 minutes, or 27 minutes, respectively. Hence, with RPT we propose a new model for robust timetabling and show with F-RPT and RPT($\mathcal{S}'$) that there exist solution algorithms to simplified variants of (RPT) yielding periodic timetables that are robust – even when evaluated in an aperiodic setting.

4. Discussion

After having summarized the main results of the five publications that make up this dissertation in the previous chapter, this section discusses the publications' individual novelties and then identifies similarities as well as connecting factors between them.

The novelty of Section 3.1 is the integrated problem formulation of finding a public transport system together with all its intermediate steps, i.e., creation of EAN and rollout. Consequently, the introduced look-ahead heuristics are among the first approaches of solving this integrated version of public transport problems with respect to cost-minimality. The major and novel contribution of Section 3.2 is the provision of optimal solutions for the integrated problem of finding cost-minimal public transport systems. Especially the tractability of the presented models might be able to open up new research avenues for integrated public transport planning with different objectives. Section 3.3 incorporates the trickle-in effect, a relevant real-world observation, into delay management models, which is novel since this effect has not yet been considered before in delay management models. The approximate cutting plane methods of Section 3.4 represent a first systematic investigation on this class of algorithms for robust optimization problems. Section 3.5 provides a novel approach for evaluating periodic timetables with respect to their delay-resistance. Next to the new methods for obtaining robust periodic timetables that result from this evaluation, another important contribution is the resulting integration of periodic timetabling and delay management.

The first property that is shared between all of the five papers is that they investigate multi-stage problems, whose difficulty arises from the interdependencies between their respective problem stages: Section 3.1 and 3.2 streamline the three stages of line planning, timetabling and vehicle scheduling in order to minimize the final objective of the multi-stage problem, i.e., vehicle schedule costs. Section 3.3 shows how modeling passenger behavior takes influence on the delay management problem and it is shown that this new model actually already interferes at the timetabling stage. Section 3.4 investigates how to iterate between robustification and pessimization stages in order to achieve performance increases for solving the overall problem. Lastly, Section 3.5 shows the potential of connecting delay management and timetabling stages for obtaining an improved robustness measure of the underlying periodic timetable.

A further immediate connection exists between Section3.1 and Section 3.2, since

the latter publication constitutes a natural extension of the former. This can be illustrated by means of the following considerations: First, the objective function of (cost-opt), i.e., g, is designed as a streamlined version of g^{cost} from Section 3.1. Second, the improvement `vs-first`, introduced in Section 3.1, is developed further in Section 3.2, culminating in the eventual inclusion of periodic vehicle schedules. Third, by arguing that there always exists a feasible timetable for the given problem structure, the objects EAN and timetable, which are included in Section 3.1, are neglected in Section 3.2. Due to this adaption, excessive problem complexity is reduced without loosing track of the overall objective of finding cost-minimal vehicle schedules. Thus, Section 3.2 constitutes an improvement of several aspects from Section 3.1 and thus extends it in a meaningful way.

Still considering Sections 3.1 and 3.2, an important lesson that can be learned from their investigation of cost-minimal public transport systems is that it is necessary to distinguish between exogenously given data and auxiliary structures inherited from models in the literature. In this case, PTN and OD-data can be viewed as exogenously given data, whereas line pool, minimum frequencies, and cost parameters cost_l of lines (from Section 3.1) are auxiliary structures that have simply been adopted from models from the literature. The convincing performance of the models from Section 3.2, which work without these auxiliary structures, leads to the takeaway that it is important to keep in mind the bigger picture by questioning existing models and considering the possibility of changes in model structures.

In addition to solving certain well-defined problems for which streamlining and rationalization might be a good approach, this thesis is furthermore concerned with improving mathematical models for real-world decision processes. According to this partition, Sections 3.1, 3.2 and 3.4 fall into the first category since they solve well-defined problems, whereas Sections 3.3 and 3.5 fall into the second category as their objective is to build better mathematical models for real-world problems. The challenge for the second category, which has been less-discussed so far, is to find a balance between the importance of adding a certain detail from the real world and the decrease of computational tractability when adding this detail. For Section 3.3 it has been shown that including the trickle-in effect can significantly change the objective of the delay management model. On the other hand, computing solutions on large scale datasets is computational tractable, which simplifies the choice of including the trickle-in effect into the model. For robust timetabling from Section 3.5, things are more complicated, for two reasons: First, it is of utmost importance to consider realistic delayed passenger travel times instead of nominal travel times. Second, however, considering delayed travel times comes at a high price in terms of computational effort. As a result, the aim of Section 3.5 is to make a reasonable choice between model detail and solvability. From the computational experiments in Section 3.5 it becomes evident that the choice made has been successful.

Choosing a well-balanced simplification, which reduces problem intricacy while maintaining essential model details, is a key aspect of many of the presented publications and thus makes up another connecting factor between the publications of this dissertation. In Section 3.1 the problem is, for example, simplified via the choice of a sequential approach to the solution process and additionally shrinking the set of possible lines and loads by working with line pool and minimal frequencies. In Section 3.2, by contrast, it is chosen to simplify the set of possible timetables and vehicle schedules. Furthermore, Section 3.3 shows that solving (DM-trick) can be straightforwardly simplified to solving (DM), which yields comparable results by using less computation time. In Section 3.4 the subproblems of the cutting plane algorithm are simplified by approximating the optimal solutions to individual subproblems. It is shown that this simplification does not compromise the overall objective of solving the general min-max problem. Finally, Section 3.5 proposes two different simplifications of the main problem (RPT): the first simplification is to fix the delay management strategy (hence the $\mathcal{D}$-space) and the second one reduces the size of considered scenarios ($\mathcal{S}$-space) in order to maintain the whole solution space Π for the timetables. With both simplifications it is possible to increase the robustness of a timetable with a reasonable amount of computational effort.

The last aspect unifying the discussed publications centers around the challenge of designing a robust public transport system. This aspect at the same time opens up a direction for further research. In this spirit, the iterative framework introduced in Section 3.4, which is successfully applied in Section 3.5, could be further extended to iteratively determine a whole public transport system. Depending on the considered objective function, the public transport stages could then be streamlined in a similar fashion to Section 3.2. This process would obtain a solution algorithm that iterates between robustification, i.e. creating a public transport system, and determining a new scenario for it by testing its robustness. Such an approach would be highly desirable as the robustness of a public transport systems is in the interest of both customers and operating companies.

5. Conclusion and Outlook

This thesis connects the two research areas of public transport optimization and robust optimization via first contributing research for each field individually and then combining this research in the investigation of robust public transportation. It thus not only provides concrete solutions to well-defined problems, such as Sections 3.1, 3.2 and 3.4, but also derives mathematical models for real-world problems by considering Sections 3.3 and 3.5.

Even though the individual publications already discuss possible directions of further research for their respective topic, this dissertation concludes by providing some additional ideas for future research endeavors. First, the models for cost-minimal public transport systems from Section 3.2 can be modified to include the requirement of satisfying a certain amount of passenger convenience, which could lead to more sophisticated line concepts and hence shorter travel times. Building a model that not only generates lines, but also creates the resulting change and go graph (defined in Schöbel and Scholl, 2006), might be an interesting extension of the models from Section 3.2. In general, short travel times (improving passenger convenience) combined with short vehicle driving times (decreasing vehicle costs) might not impose a big contradiction after all. An argument for this statement is that a passenger having long travel times might also occupy vehicles for a long time, which results in more costs for the operating company. Second, the linear objective for timetabling as well as for delay management, i.e., (delayed) duration multiplied with the number of passengers, does not seem to reflect experiences in practice: Delaying 100 passengers for one minute might not result in severe dissatisfaction, whereas one passenger with 100 minutes of delay might have more reason to file a complaint. This dissatisfaction might even be amplified if the passenger's nominal travel time is comparably short. Thus, a quadratic objective function, which could naturally penalize large delays, might be better suited to address this issue in a mathematical model.

On a more general level, it seems worth to emphasize that research in public transport stems from a concrete real-world application and is hence not reserved for mathematical optimization and mixed integer programming. Thus, in the future it could be fruitful to consider more than only one particular method for solving public transport optimization problems, which has been mostly mixed integer (linear) programming so far. Related methods, such as constraint programming, have already been investigated (e.g., Großmann et al., 2012; Gattermann, Großmann, Nachtigall,

and Schöbel, 2016a), but especially the combination with upcoming machine learning methods could turn out to be a promising research direction, not only for public transport optimization, but also for mathematical optimization in general (cf., e.g., Khalil, Dai, Zhang, Dilkina, and Song, 2017).

All in all, with this thesis it is hoped to show that public transport optimization as well as robust optimization are highly relevant, mathematically intriguing, and, on a personal level, fascinating and fun topics to work on.

6. Summary of Contributions

The author's contributions to the presented publications are as follows.

- "Look-Ahead Approaches for Integrated Planning in Public Transport Optimization": Parts of the look-ahead improvement ideas and parts of the implementation as well as large parts of the computational experiments and a considerable part of the write-up are the author's work, judging his contribution to be 40%.
- "Cost-Minimal Public Transport Planning": For the first two models, ideas, proofs of correctness, cutting plane improvements, implementation and computational experiments as well as considerable parts of the paper's write-up are the author's work, judging his contribution to be 55%.
- "The Trickle-in Effect: Modeling Passenger Behavior in Delay Management": Implementation, computational results and considerable parts of the write-up are the author's work, judging his contribution to be 40%.
- "Approximate Cutting Planes for Exact Solutions to Robust Optimization Problems": Large parts of the ideas, proofs, write-up as well as all implementations and computational experiments are the author's work, judging his contribution to be 80%.
- "Finding Robust Periodic Timetables by Integrating Delay Management": Ideas, proofs, implementation, computational experiments and write-up are the author's work, judging his contribution to be 100%.

Bibliography

Aissi, H., Bazgan, C., & Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, *197*(2), 427–438.

Albert, S., Kraus, P., Müller, J., & Schöbel, A. (2018). Passenger-induced delay propagation: Agent-based simulation of passengers in rail networks. In *Clausthal-göttingen international workshop on simulation science 2017*. Communications in Computer and Information Science (CCIS). to appear. Springer.

Amberg, B., Amberg, B., & Kliewer, N. (2018). Robust efficiency in urban public transportation: Minimizing delay propagation in cost-efficient bus and driver schedules. *Transportation Science*, *53*(1), 89–112.

Assavapokee, T., Realff, M. J., Ammons, J. C., & Hong, I.-H. (2008). Scenario relaxation algorithm for finite scenario-based min–max regret and min–max relative regret robust optimization. *Computers & operations research*, *35*(6), 2093–2102.

Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. (Eds.). (2006). *Special issue on robust optimization*. Mathematical Programming B. Springer.

Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. (2009). *Robust optimization*. Princeton: Princeton University Press.

Ben-Tal, A. & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, *23*(4), 769–805.

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

Ben-Tal, A., Goryashko, A., Guslitzer, E., & Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, *99*(2), 351–376.

Bertsimas, D., Dunning, I., & Lubin, M. (2016). Reformulation versus cutting-planes for robust optimization. *Computational Management Science*, *13*(2), 195–217.

Bertsimas, D. & Sim, M. (2004). The price of robustness. *Operations Research*, *52*(1), 35–53.

Bienstock, D. (2007). Histogram models for robust portfolio optimization. *Journal of computational finance*, *11*(1), 1.

Borndörfer, R., Grötschel, M., & Pfetsch, M. (2007). A Column-Generation Approach to Line Planning in Public Transport. *Transportation Science*, *41*, 123–132.

Borndörfer, R., Karbstein, M., Liebchen, C., & Lindner, N. (2018). A simple way to compute the number of vehicles that are required to operate a periodic timetable. In R. Borndörfer & S. Storandt (Eds.), *18th workshop on algorithmic approaches for transportation modelling, optimization, and systems (atmos 2018)* (Vol. 65, 16:1–16:15). OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.ATMOS.2018.16

Borndörfer, R., Hoppmann, H., & Karbstein, M. (2017). Passenger routing for periodic timetable optimization. *Public Transport*, *9*(1-2), 115–135.

Borndörfer, R., Hoppmann, H., Karbstein, M., & Lindner, N. (2018). *Separation of cycle inequalities in periodic timetabling* (tech. rep. No. 18-16). ZIB. Takustr. 7, 14195 Berlin.

Borndörfer, R. & Karbstein, M. (2012). A direct connection approach to integrated line planning and passenger routing. In *12th workshop on algorithmic approaches for transportation modelling, optimization, and systems*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Borndörfer, R., Lindner, N., & Roth, S. (2019). *A concurrent approach to the periodic event scheduling problem* (tech. rep. No. 19-07). ZIB. Takustr. 7, 14195 Berlin.

Borndörfer, R., Reuther, M., Schlechte, T., Waas, K., & Weider, S. (2015). Integrated optimization of rolling stock rotations for intercity railways. *Transportation Science*, *50*(3), 863–877.

Borndörfer, R., Reuther, M., Schlechte, T., & Weider, S. (2011). A hypergraph model for railway vehicle rotation planning. In *11th workshop on algorithmic approaches for transportation modelling, optimization, and systems*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Bunte, S. & Kliewer, N. (2009). An overview on vehicle scheduling models. *Public Transport*, *1*(4), 299–317.

Bürger, M., Notarstefano, G., & Allgöwer, F. (2014). A polyhedral approximation framework for convex and robust distributed optimization. *IEEE Transactions on Automatic Control*, *59*(2), 384–395.

Burggraeve, S., Bull, S., Vansteenwegen, P., & Lusby, R. (2017). Integrating robust timetabling in line plan optimization for railway systems. *Transportation Research Part C: Emerging Technologies*, *77*, 134–160.

Bussieck, M. R., Winter, T., & Zimmermann, U. T. (1997). Discrete optimization in public rail transport. *Mathematical programming*, *79*(1-3), 415–444.

Bussieck, M. (1998). *Optimal lines in public transport* (Doctoral dissertation, Technische UniversitÃ€t Braunschweig).

Cacchiani, V., Caprara, A., Galli, L., Kroon, L., Maróti, G., & Toth, P. (2012). Railway rolling stock planning: Robustness against large disruptions. *Transportation Science*, *46*(2), 217–232.

Cacchiani, V. & Toth, P. (2012). Nominal and robust train timetabling problems. *European Journal of Operational Research*, *219*(3), 727–737.

Cadarso, L. & Marín, Á. (2014). Improving robustness of rolling stock circulations in rapid transit networks. *Computers & Operations Research*, *51*, 146–159.

Caprara, A., Fischetti, M., & Toth, P. (2002). Modeling and solving the train timetabling problem. *Operations research*, *50*(5), 851–861.

Ceder, A. & Wilson, N. H. (1986). Bus network design. *Transportation Research Part B: Methodological*, *20*(4), 331–344.

Cicerone, S., D'Angelo, G., Stefano, G. D., Frigioni, D., Navarra, A., Schachtebeck, M., & Schöbel, A. (2009). Recoverable robustness in shunting and timetabling. In R. K. Ahuja, R. Möhring, & C. Zaroliagis (Eds.), *Robust and online large-scale optimization* (Vol. 5868, pp. 28–60). Lecture Notes in Computer Science. Springer.

Cicerone, S., D'Angelo, G., Di Stefano, G., Frigioni, D., & Navarra, A. (2007). On the interaction between robust timetable planning and delay management. In *2nd annual international conference on combinatorial optimization and applications (cocoa'08)* (Vol. 4598).

Claessens, M., van Dijk, N., & Zwaneveld, P. (1998). Cost optimal allocation of rail passenger lines. *European Journal on Operational Research*, *110*, 474–489.

De Giovanni, L., Heilporn, G., & Labbé, M. (2008). Optimization models for the single delay management problem in public transportation. *European Journal of Operational Research*, *189*(3), 762–774.

Desaulniers, G. & Hickman, M. D. (2007). Public transit. *Handbooks in operations research and management science*, *14*, 69–127.

DFG Research Unit FOR 2083. (2019). Public tranport networks. Retrieved May 3, 2019, from https://github.com/FOR2083/PublicTransportNetworks

Dollevoet, T., Huisman, D., Kroon, L., Schmidt, M., & Schöbel, A. (2015). Delay management including capacities of stations. *Transportation Science*, *49*(2), 185–203.

Dollevoet, T., Huisman, D., Kroon, L., Veelenturf, L., & J.C.Wagenaar. (2017). Application of an iterative framework for real-time railway scheduling. *Computers and Operations Research*, *78*, 203–217.

Dollevoet, T., Huisman, D., Schmidt, M., & Schöbel, A. (2012). Delay management with rerouting of passengers. *Transportation Science*, *46*(1), 74–89.

Dollevoet, T., Huisman, D., Schmidt, M., & Schöbel, A. (2018). Delay propagation and delay management in transportation networks. In R. B. et al. (Ed.), *Handbook of optimization in the railway industry*. Springer.

Fischetti, M. & Monaci, M. (2009). Light robustness. In R. K. Ahuja, R. Möhring, & C. Zaroliagis (Eds.), *Robust and online large-scale optimization* (Vol. 5868, pp. 61–84). Lecture Note on Computer Science. Springer.

Fischetti, M. & Monaci, M. (2012). Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation*, *4*(3), 239–273.

Friedrich, M., Hartl, M., Schiewe, A., & Schöbel, A. (2017). Integrating Passengers' Assignment in Cost-Optimal Line Planning. In G. D'Angelo & T. Dollevoet (Eds.), *17th workshop on algorithmic approaches for transportation modelling, optimization, and systems (atmos 2017)* (Vol. 59, pp. 1–16). OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany.

Gattermann, P., Großmann, P., Nachtigall, K., & Schöbel, A. (2016a). Integrating Passengers' Routes in Periodic Timetabling: A SAT approach. In M. Goerigk & R. Werneck (Eds.), *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)* (Vol. 54, pp. 1–15). OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:http://dx.doi.org/10.4230/OASIcs.ATMOS.2016.3

Gattermann, P., Harbering, J., & Schöbel, A. (2017). Line pool generation. *Public Transport*, *9*(1), 7–32.

Gattermann, P., Großmann, P., Nachtigall, K., & Schöbel, A. (2016b). Integrating passengers' routes in periodic timetabling: A sat approach. In *16th workshop on algorithmic approaches for transportation modelling, optimization, and systems (atmos 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Ghaoui, L. E. & Lebret, H. (1997). Robust solutions to least-squares problems with uncertain data. *SIAM Journal of Matrix Anal. Appl. 18*, 1034–1064.

Giacco, G. L., D'Ariano, A., & Pacciarelli, D. (2014). Rolling stock rostering optimization under maintenance constraints. *Journal of Intelligent Transportation Systems*, *18*(1), 95–105.

Goerigk, M. (2012). *Algorithms and concepts for robust optimization* (Doctoral dissertation, Universität Göttingen).

Goerigk, M. (2015). Exact and heuristic approaches to the robust periodic event scheduling problem. *Public Transport*, *7*(1), 101–119.

Goerigk, M., Knoth, M., Müller-Hannemann, M., Schmidt, M., & Schöbel, A. (2014). The Price of Strict and Light Robustness in Timetable Information. *Transportation Science*, *48*, 225–242.

Goerigk, M. & Schmidt, M. (2017). Line planning with user-optimal route choice. *European Journal of Operational Research*, *259*(2), 424–436.

Goerigk, M. & Schöbel, A. (2013). Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, *40*(5), 1363–1370.

Goerigk, M. & Schöbel, A. (2016). Algorithm engineering in robust optimization. In L. Kliemann & P. Sanders (Eds.), *Algorithm engineering: Selected results*

and surveys (Vol. 9220, pp. 245–279). LNCS State of the Art. Retrieved from http://arxiv.org/abs/1505.04901

Goerigk, M. (2018). PESPlib - A benchmark library for periodic event scheduling. Retrieved May 3, 2019, from http://num.math.uni-goettingen.de/~m.goerigk/pesplib/

Goerigk, M. & Liebchen, C. (2017). An improved algorithm for the periodic timetabling problem. In *17th workshop on algorithmic approaches for transportation modelling, optimization, and systems (atmos 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Goerigk, M., Schachtebeck, M., & Schöbel, A. (2013). Evaluating line concepts using travel times and robustness. *Public Transport*, *5*(3), 267–284.

Goerigk, M. & Schöbel, A. (2014). Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. *Computers & Operations Research*, *52*, 1–15.

Goossens, J., van Hoesel, C., & Kroon, L. (2006). On solving multi-type railway line planning problems. *European Journal of Operational Research*, *168*(2), 403–424.

Goossens, J.-W., Van Hoesel, S., & Kroon, L. (2004). A branch-and-cut approach for solving railway line-planning problems. *Transportation Science*, *38*(3), 379–393.

Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., & Steinke, P. (2012). Solving periodic event scheduling problems with sat. In *International conference on industrial, engineering and other applications of applied intelligent systems* (pp. 166–175). Springer.

Harbering, J. (2016). *Planning a public transportation system with a view towards passengers' convenience* (Doctoral dissertation, Universität Göttingen).

Huisman, D., Freling, R., & Wagelmans, A. P. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, *39*(4), 491–502.

Huisman, D., Kroon, L. G., Lentink, R. M., & Vromans, M. J. (2005). Operations research in passenger railway transportation. *Statistica Neerlandica*, *59*(4), 467–497.

Kelley, J. E., Jr. (1960). The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, *8*(4), 703–712.

Kepaptsoglou, K. & Karlaftis, M. (2009). Transit route network design problem. *Journal of transportation engineering*, *135*(8), 491–505.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in neural information processing systems* (pp. 6348–6358).

Kouvelis, P. & Yu, G. (1997). *Robust discrete optimization and its applications*. Kluwer Academic Publishers.

Kroon, L., Maróti, G., Helmrich, M. R., Vromans, M., & Dekker, R. (2008). Stochastic improvement of cyclic railway timetables. *Transportation Research Part B: Methodological*, *42*(6), 553–570.

Laporte, G., Mesa, J., Ortega, F., & Perea, F. (2011). Planning rapid transit networks. *Socio-Economic Planning Sciences*, *45*(3), 95–104.

Laporte, G., Mesa, J. A., & Ortega, F. A. (2000). Optimization methods for the planning of rapid transit systems. *European Journal of Operational Research*, *122*(1), 1–10.

Laporte, G., Mesa, J. A., & Perea, F. (2010). A game theoretic framework for the robust railway transit network design problem. *Transportation Research Part B: Methodological*, *44*(4), 447–459.

Liebchen, C., Schachtebeck, M., Schöbel, A., Stiller, S., & Prigge, A. (2010). Computing delay-resistant railway timetables. *Computers and Operations Research*, *37*, 857–868.

Liebchen, C. (2007). Periodic timetable optimization in public transport. In *Operations research proceedings 2006* (pp. 29–36). Springer.

Liebchen, C. (2008). The first optimized railway timetable in practice. *Transportation Science*, *42*(4), 420–435.

Liebchen, C., Lübbecke, M., Möhring, R., & Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization* (pp. 1–27). Springer.

Lusby, R., Larsen, J., Ehrgott, M., & Ryan, D. (2011). Railway track allocation: Models and methods. *OR spectrum*, *33*(4), 843–883.

Lusby, R. M., Larsen, J., & Bull, S. (2018). A survey on robustness in railway planning. *European Journal of Operational Research*, *266*(1), 1–15.

Marín, Á., Mesa, J. A., & Perea, F. (2009). Integrating robust railway network design and line planning under failures. In *Robust and online large-scale optimization* (pp. 273–292). Springer.

Montemanni, R. (2006). A benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, *174*(3), 1479–1490.

Mutapcic, A. & Boyd, S. (2009). Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods & Software*, *24*(3), 381–406.

Nachtigall, K. (1998). Periodic network optimization and fixed interval timetables. *Deutsches Zentrum für Luft–und Raumfahrt, Institut für Flugführung, Braunschweig*.

Nachtigall, K. & Opitz, J. (2008). Solving periodic timetable optimisation problems by modulo simplex calculations. In *Oasics-openaccess series in informatics* (Vol. 9). Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Odijk, M. A. (1996). A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, *30*(6), 455–464.

Patriksson, M. (2015). *The traffic assignment problem: Models and methods*. Courier Dover Publications.

Pätzold, J., Schiewe, A., Schiewe, P., & Schöbel, A. (2017). Look-Ahead Approaches for Integrated Planning in Public Transportation. In G. D'Angelo & T. Dollevoet (Eds.), *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)* (Vol. 59, pp. 1–16). OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany.

Pätzold, J. & Schöbel, A. (2016). A Matching Approach for Periodic Timetabling. In M. Goerigk & R. Werneck (Eds.), *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)* (Vol. 54, pp. 1–15). OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany.

Pätzold, J., Schiewe, A., & Schöbel, A. (2018). Cost-Minimal Public Transport Planning. In R. Borndörfer & S. Storandt (Eds.), *18th workshop on algorithmic approaches for transportation modelling, optimization, and systems (atmos 2018)* (Vol. 65, 8:1–8:22). OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.ATMOS.2018.8

Peeters, L. (2003). *Cyclic Railway Timetabling Optimization* (Doctoral dissertation, ERIM, Rotterdam School of Management).

Peeters, L. & Kroon, L. (2001). A cycle based optimization model for the cyclic railway timetabling problem. In *Computer-aided scheduling of public transport* (pp. 275–296). Springer.

Polinder, G.-J., Breugem, T., Dollevoet, T., & Maróti, G. (2019). *An adjustable robust optimization approach for periodic timetabling*. Retrieved from http://hdl.handle.net/1765/113303

Reemtsen, R. (1994). Some outer approximation methods for semi-infinite optimization problems. *Journal of Computational and Applied Mathematics*, *53*(1), 87–108.

Rückert, R., Lemnian, M., Blendinger, C., Rechner, S., & Müller-Hannemann, M. (2017). Panda: A software tool for improved train dispatching with focus on passenger flows. *Public Transport*, *9*, 307–324.

Schachtebeck, M. (2010). *Delay management in public transportation: Capacities, robustness, and integration* (Doctoral dissertation, Universität Göttingen).

Schachtebeck, M. & Schöbel, A. (2010). To wait or not to wait and who goes first? Delay management with priority decisions. *Transportation Science*, *44*(3), 307–321. doi:10.1287/trsc.1100.0318

Schiewe, A. (2019). *Integrated algorithms for cost-optimal public transport planning* (Doctoral dissertation, Universität Göttingen).

Schiewe, A., Albert, S., Pätzold, J., Schiewe, P., Schöbel, A., & Schulz, J. (2018). *LinTim: An integrated environment for mathematical public transport optimization. Documentation* (tech. rep. No. 2018-08). Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen.

Schiewe, P. (2018). *Integrated Optimization in Public Transport Planning* (Doctoral dissertation, Universität Göttingen).

Schmidt, M. (2014). *Integrating Routing Decisions in Public Transportation Problems*. Optimization and Its Applications. Springer.

Schmidt, M. & Schöbel, A. (2015a). The complexity of integrating routing decisions in public transportation models. *Networks*, *65*(3), 228–243.

Schmidt, M. & Schöbel, A. (2015b). Timetabling with passenger routing. *OR Spectrum*, *37*, 75–97.

Schöbel, A. (2001). A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, *50*(1).

Schöbel, A. (2007). Integer programming approaches for solving the delay management problem. In *Algorithmic methods for railway optimization* (4359, pp. 145–170). Lecture Notes in Computer Science. Springer.

Schöbel, A. (2012). Line planning in public transportation: Models and methods. *OR Spectrum*, *34*(3), 491–510.

Schöbel, A. (2014). Generalized light robustness and the trade-off between robustness and nominal quality. *MMOR*, *80*(2), 161–191.

Schöbel, A. (2017). An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research C*, *74*, 348–365.

Schöbel, A. & Scholl, S. (2006). Line planning with minimal travel time. In *5th workshop on algorithmic methods and models for optimization of railways* (06901). Dagstuhl Seminar Proceedings.

Serafini, P. & Ukovich, W. (1989). A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, *2*(4), 550–581.

Siddiqui, S., Azarm, S., & Gabriel, S. (2011). A modified benders decomposition method for efficient robust optimization under interval uncertainty. *Structural and Multidisciplinary Optimization*, *44*(2), 259–275.

Siebert, M. & Goerigk, M. (2013). An experimental comparison of periodic timetabling models. *Computers & Operations Research*, *40*(10), 2251–2259.

Soyster, A. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, *21*, 1154–1157.

Steinzen, I., Gintner, V., Suhl, L., & Kliewer, N. (2010). A time-space network approach for the integrated vehicle-and crew-scheduling problem with multiple depots. *Transportation Science*, *44*(3), 367–382.

Suhl, L., Biederbick, C., & Kliewer, N. (2001). Design of customer-oriented dispatching support for railways. In S. Voß & J. Daduna (Eds.), *Computer-aided transit scheduling* (Vol. 505, pp. 365–386). Lecture Notes in Economics and Mathematical systems. Springer.

Terry, T. L. (2009). Robust linear optimization with recourse: Solution methods and other properties.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. *European journal of operational research*, *226*(3), 367–385.

Zwaneveld, P. (1997). *Railway planning — routing of trains and allocation of passenger lines* (Doctoral dissertation, School of Management, Rotterdam).

Appendix

A. Look-Ahead Approaches for Integrated Planning in Public Transportation

J. Pätzold, A. Schiewe, P. Schiewe, A. Schöbel
Look-Ahead Approaches for Integrated Planning in Public Transportation
Published as Pätzold, Schiewe, Schiewe, and Schöbel (2017) (Proceedings of *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, 2017).
Awarded with ATMOS 2017 Best Paper Award.

Look-Ahead Approaches for Integrated Planning in Public Transportation*

Julius Pätzold[1], Alexander Schiewe[2], Philine Schiewe[3], and Anita Schöbel[4]

1 **Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Göttingen, Germany**
`j.paetzold@math.uni-goettingen.de`
2 **Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Göttingen, Germany**
`a.schiewe@math.uni-goettingen.de`
3 **Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Göttingen, Germany**
`p.schiewe@math.uni-goettingen.de`
4 **Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Göttingen, Germany**
`schoebel@math.uni-goettingen.de`

Abstract

In this paper we deal with three consecutive planning stages in public transportation: Line planning (including line pool generation), timetabling, and vehicle scheduling. These three steps are traditionally performed one after another in a sequential way often leading to high costs in the (last) vehicle scheduling stage. In this paper we propose three different ways to "look ahead", i.e., to include aspects of vehicle scheduling already earlier in the sequential process: an adapted line pool generation algorithm, a new cost structure for line planning, and a reordering of the sequential planning stages. We analyze these enhancements experimentally and show that they can be used to decrease the costs significantly.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Theory, G.2.3 Applications

Keywords and phrases line pool generation, line planning, vehicle scheduling, integrated planning, public transport

Digital Object Identifier 10.4230/OASIcs.ATMOS.2017.17

1 Sequential versus integrated planning

Planning a public transport supply can have many goals. Two major goals are usually minimizing the perceived travel times of passengers as well as the costs that incur to the public transportation company. Motivated by this we consider a bi-objective model for railway or bus planning with these two objectives.

Traditionally, public transportation planning is done in sequential stages. The first stage after the design of a network, that is spanned by stops (or stations) and their direct connections (edges or tracks), is *line planning*. In this stage, first a set of possible lines, the line pool, has to be generated on the network. Research towards the effect of line pool

* This work was partially supported by DFG under SCHO 1140/8-1 and by the Simulation Science Center Clausthal/Göttingen.

17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017).
Editors: Gianlorenzo D'Angelo and Twan Dollevoet; Article No. 17; pp. 17:1–17:16

Open Access Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

generation, and an algorithm to find suitable line pools is presented in [7]. In the line planning problem one then chooses a feasible subset of lines from the line pool, i.e., a set of lines such that all passengers can be transported. See [21] for an overview. With a given line plan one can create an event-activity network which constitutes the input for the *timetabling stage*. Periodic timetabling consists of deciding when and how fast vehicles (trains or buses) should drive along the edges and how long they should wait at stops (or stations). The problem is modeled as a periodic event scheduling problem (PESP), see [23]. Other timetabling models can be found in [10]. After a timetable is chosen, vehicle schedules are planned, determining which vehicle should drive which route such that all lines are operated according to their timetables. A survey on *vehicle scheduling* is given in [4]. Finally, crew scheduling and rostering are planning stages to be performed after the vehicle schedules are found.

Obviously, proceeding sequentially does not need to lead to an optimal solution as there are dependencies between the different subproblems. It would hence be beneficial to solve the entire problem in an integrated system. Since this is computationally too complex, heuristic approaches have been proposed as in [22].

Our contribution. We consider line planning, timetabling and vehicle scheduling in conjunction with each other. To this end we formally define what an *integrated* transport supply (LTS-plan), consisting of a line plan, a timetable, and a vehicle schedule, is and how it can be evaluated. We propose three enhancements of the traditional approach which consider the vehicle scheduling costs already in the line planning stage. Finally, we evaluate them experimentally and show that our proposed enhancements lead to LTS-plans with significantly smaller costs than the traditional sequential approach.

2 A bi-objective model for integrated planning in public transportation

In this section we formally describe what a feasible transport supply (*LTS-plan*), consisting of a line plan (L), a timetable (T), and a vehicle schedule (S), is and how its quality can be evaluated. Note that for the single stages, i.e., for a line plan, for a timetable, and for a vehicle schedule, this has been extensively discussed in the literature. However, it is in the literature usually assumed that an event-activity network is already known for timetabling and a set of trips is already given for vehicle scheduling. Since we plan from scratch, we also have to describe the intermediate steps, i.e., how to build the event-activity network and how to build the set of trips. In order to keep the timetabling step tractable, we restrict ourselves in this paper to periodic LTS-plans for which all lines are operated with the same frequency.

As input for the bi-objective model we are given:

- A public transport network PTN= (V, E) consisting of a set of stops V and direct connections E between them.
- For every node $v \in V$:
  - lower and upper bounds $L_v^{wait} \leq U_v^{wait}$ for the time vehicles wait at stop v,
  - lower and upper bounds $L_v^{trans} \leq U_v^{trans}$ for the time passengers need to transfer between two vehicles at the same stop v.
  - We furthermore need for every pair $v, u \in V$ the time(v, u) a vehicle needs if it drives directly from stop v to stop u.
- For every edge $e = (v_1, v_2) \in E$:
  - a length (in kilometers) length$_e$,
  - lower and upper edge frequency bounds $f_e^{\min} \leq f_e^{\max}$,
  - lower and upper bounds on the travel times along the edge, i.e., $L_e^{drive} \leq U_e^{drive}$.

- An OD-matrix W with entries W_{uv} for each pair of stops $u, v \in V$. The OD-matrix is assumed to be consistent with the lower edge frequencies, i.e., there exist paths P_{uv} for every OD-pair (u, v) through the PTN such that for every edge e we have:

$$\sum_{u,v \in V:\ e \in P_{uv}} W_{uv} \leq \text{Cap} \cdot f_e^{\min}$$

  for Cap being the capacity of the (identical) vehicles, i.e., each passenger can be transported,
- a period length T, and the number of periods p to be considered for planning
- a penalty pen for transfers,
- a minimal turnaround time for vehicles $L_{\min}$,
- cost parameters
  - c_1 costs per minute for a vehicle driving with passengers,
  - c_2 costs per kilometer for a vehicle driving with passengers,
  - c_3 costs per vehicle for the whole planning horizon (p periods),
  - c_4 costs per minute for a vehicle driving empty (i.e., without passengers),
  - c_5 costs per kilometer for a vehicle driving empty (i.e., without passengers).

We then look for an LTS-plan, which consists of a *line plan* (L), a periodic *timetable* (T) and a *vehicle schedule* (S) which are together feasible. These objects are defined as follows:

Line plan L

A *line* is a path through the PTN. A *line plan* is a set of lines $\mathcal{L}$, which is feasible if

$$f_e^{\min} \leq |\{l \in \mathcal{L} : e \in l\}| \leq f_e^{\max}, \tag{1}$$

i.e., if each edge of the PTN is covered by the required number of lines. We assume that lines are symmetric, i.e., they are operated in both directions. In our setting all lines are operated with a frequency of 1.

Timetable T

Given a set of lines, a timetable assigns a time to every departure and arrival of every line at its stops. These times are then repeated periodically. In order to model a timetable usually event-activity networks $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ are used (see, e.g., [11, 12, 14, 17, 18]). The set of events $\mathcal{E}$ consists of all departures and all arrivals of all lines at all stops, and the set $\mathcal{A}$ connects these events by driving, waiting and transfer activities. For each activity, the number of passengers using this activity is usually given as input for timetabling. (It is subject of ongoing research how this can be relaxed, see [3, 6, 19, 20]). The lower and upper bounds L_a and U_a are set as

- L_e^{drive} and U_e^{drive} if a is a driving activity on edge $e \in E$,
- L_v^{wait} and U_v^{wait} if a is a waiting activity in stop $v \in V$, and as
- L_v^{trans} and U_v^{trans} if a is a transfer activity in stop $v \in V$.

A timetable π is an assignment of times $\pi_j \in \mathbb{Z}$ to every event $j \in \mathcal{E}$. It is feasible if it respects the lower and upper bounds for all its activities, i.e., if

$$(\pi_j - \pi_i - L_a) \mod T \in [0, U_a - L_a] \text{ for all } a = (i, j) \in \mathcal{A}. \tag{2}$$

The objective function in timetabling minimizes the total slack times. If all passengers use the paths they have been assigned to in the event-activity network this is equivalent to minimizing the sum of passengers' travel times.

Vehicle schedule S

Given a set of lines and a timetable, a *vehicle schedule* determines the number of vehicles and the exact routes of the vehicles for operating the timetable. To this end, we use the line plan and the timetable to construct a set of *trips* $\mathcal{T}$ where each trip

$$t = (l_t, v_t^{start}, v_t^{end}, \tilde{\pi}_t^{start}, \tilde{\pi}_t^{end}) \in \mathcal{T}$$

is specified by a line l_t together with its first and last stop v_t^{start} and v_t^{end} and its corresponding *start time* $\tilde{\pi}_t^{start}$ and *end time* $\tilde{\pi}_t^{end}$. These times can be taken from the periodic timetable, but we have to consider the real time (e.g. in minutes after midnight) by adding the correct multiple of the period length. The end time $\tilde{\pi}_t^{end}$ of a line at its final stop is the arrival time at this stop plus some minutes allowing passengers to deboard. Analogously, the start time $\tilde{\pi}_t^{start}$ of a line at a stop is the time when it arrives at this stop, i.e., a bit earlier than its departure time there. For every line l we receive two trips starting per period, namely one forward and one backward trip. A route of a vehicle is given by its sequence of trips $r = (t_1, \ldots, t_k)$ such that

$$(\tilde{\pi}_{t_{i+1}}^{start} - \tilde{\pi}_{t_i}^{end}) \geq \text{time}(v_{t_i}^{end}, v_{t_{i+1}}^{start}) \text{ for all } i = 1, \ldots, k-1.$$

A set of vehicle routes $\mathcal{R}$ is feasible if all its routes are feasible and if each trip is contained in exactly one route.

Evaluating an LTS-plan

An LTS-plan is specified by a line plan, a corresponding timetable and a corresponding vehicle schedule, i.e., it is specified by the tuple $(\mathcal{L}, \pi, \mathcal{R})$. Given a feasible LTS-plan we use the two most common evaluation criteria: the sum of passengers' travel times (including a penalty for every transfer) and the costs. These objectives are formally defined below:

Costs. The costs of an LTS-plan depend mainly on the costs of the corresponding vehicle schedule and thus on the distance which is driven, the total duration of driving and the number of required vehicles. For the distance and the duration of the trips we distinguish if the vehicle drives on a trip which can be used by passengers (here called *full ride*) or if the vehicle drives empty between two consecutive trips t_i, t_{i+1} in the same vehicle route (here called an *empty ride*) as the costs can be different for full and empty rides.

As the vehicle schedule in general is aperiodic, we consider the costs for a whole planning horizon (e.g. a day) instead of a planning period by rolling out the periodic line plan and timetable for a fixed time span which is given by the number of periods p it covers. Note that we have to take special care at the beginning and the end of the roll-out period, regarding lines traversing the period boundaries. For simplicity reasons we do not go into detail here how this is handled explicitly.

Before defining the costs, we introduce the duration and the length of a line and an empty ride. Let a line be defined as a sequence of nodes and edges.

The duration of a line can be determined after the timetable is known. We get

$$\begin{aligned} \text{dur}_l = & \sum_{\substack{a=(i,j)\in\mathcal{A}_{drive}:\\ a \text{ belongs to } e\in l}} (L_e^{drive} + (\pi_j - \pi_i - L_e^{drive} \mod T)) \\ & + \sum_{\substack{a=(i,j)\in\mathcal{A}_{wait}:\\ a \text{ belongs to } v\in l}} (L_v^{wait} + (\pi_j - \pi_i - L_v^{wait} \mod T)), \end{aligned}$$

i.e., all driving times along edges and waiting times at stops are added. When a heuristic approach to timetabling is used where the duration of all driving and waiting activities is set to their respective lower bounds, as done here, the duration of a line simplifies to

$$\text{dur}_l = \sum_{e \in l} L_e^{drive} + \sum_{v \in l} L_v^{wait}. \tag{3}$$

The length of a line is computed as sum over all edge lengths

$$\text{length}_l = \sum_{e \in l} \text{length}_e$$

and is independent from the timetable. The duration of an empty ride between two trips $t_1 = (l_{t_1}, v_{t_1}^{start}, v_{t_1}^{end}, \tilde{\pi}_{t_1}^{start}, \tilde{\pi}_{t_1}^{end})$ and $t_2 = (l_{t_2}, v_{t_2}^{start}, v_{t_2}^{end}, \tilde{\pi}_{t_2}^{start}, \tilde{\pi}_{t_2}^{end})$ can be computed as

$$\text{dur}_{t_1,t_2} = \tilde{\pi}_{t_2}^{start} - \tilde{\pi}_{t_1}^{end},$$

i.e., the time between the end of t_1 and the start of t_2.

The length of the empty ride is defined as

$$\text{length}_{t_1,t_2} = SP(v_{t_1}^{end}, v_{t_2}^{start}),$$

i.e., we assume that a vehicle takes the shortest path from the last station $v_{t_1}^{end}$ of trip t_1 to the first station $v_{t_2}^{start}$ of trip t_2.

Now we can define the following cost components. Note that we have to count the full duration and length of each line twice as two trips belong to every line (one in forward and one in backward direction).

- *full duration*, i.e., time it takes to cover all trips (full rides):

$$\text{dur}_{\text{full}} = \sum_{l \in \mathcal{L}} 2 \cdot \text{dur}_l \cdot \text{p}\,,$$

- *full distance*, i.e., distance driven along lines:

$$\text{length}_{\text{full}} = \sum_{l \in \mathcal{L}} 2 \cdot \text{length}_l \cdot \text{p}\,,$$

- *number of vehicles*: $\text{veh} = |\mathcal{R}|\,,$
- *empty duration*, i.e., time of empty rides between trips:

$$\text{dur}_{\text{empty}} = \sum_{r=(t_1,\ldots,t_{k_r}) \in \mathcal{R}} \sum_{i=1}^{k_r-1} \text{dur}_{t_i,t_{i+1}}\,,$$

- empty distance, i.e., distance of empty rides between trips:

$$\text{length}_{\text{empty}} = \sum_{r=(t_1,\ldots,t_{k_r}) \in \mathcal{R}} \sum_{i=1}^{k_r-1} \text{length}_{t_i,t_{i+1}}\,.$$

In total we get

$$g^{\text{cost}}(\mathcal{L}, \pi, \mathcal{R}) := c_1 \cdot \text{dur}_{\text{full}} + c_2 \cdot \text{length}_{\text{full}} + c_3 \cdot \text{veh} + c_4 \cdot \text{dur}_{\text{empty}} + c_5 \cdot \text{length}_{\text{empty}}. \tag{4}$$

Travel times. For determining the travel time we follow the traditional approach of fixing the passengers' routes when constructing the event-activity network, assuming that the passengers use these assigned paths. In the event-activity network, passengers are routed on a shortest path according to the lower bounds on the activities and assigned as weights c_a to the activities $a \in \mathcal{A}$. Additionally to the travel time, we consider a penalty pen for every transfer. The total *perceived* travel time on these fixed paths can then be determined as

$$g^{\text{time}}(\mathcal{L}, \pi, \mathcal{R}) = \sum_{a=(i,j)\in\mathcal{A}} c_a \cdot (L_a + (\pi_j - \pi_i - L_a \mod T)) + \sum_{a\in\mathcal{A}_{trans}} c_a \cdot \text{pen}. \tag{5}$$

Note that the travel time does not depend on the vehicle schedule.

The two objective functions we have sketched here are common in the literature when broken down to one single planning stage:

Nearly all papers dealing with vehicle scheduling minimize a combination of empty kilometers and number of vehicles needed, i.e., $\text{veh} + a \cdot \text{length}_{\text{empty}}$. This is equivalent to g^{cost} if the duration of full and empty rides are weighted equally and a is chosen as $a = \frac{c_5}{c_3}$ since the duration and the length of the lines are all known due to the timetable being fixed.

In timetabling, the goal is usually to minimize the sum of (perceived) travel times for the passengers. Since it is computationally very difficult, most papers make the simplifying assumption that the number of travelers on every activity in the event-activity network is known and fixed, as it is done here.

Pareto optimal LTS-plans. We call a feasible LTS-plan $(\mathcal{L}, \pi, \mathcal{R})$ *Pareto optimal* if there does not exist another LTS-plan $(\mathcal{L}', \pi', \mathcal{R}')$ which satisfies

$$g^{\text{cost}}(\mathcal{L}', \pi', \mathcal{R}') \le g^{\text{cost}}(\mathcal{L}, \pi, \mathcal{R}), \quad g^{\text{time}}(\mathcal{L}', \pi', \mathcal{R}') \le g^{\text{time}}(\mathcal{L}, \pi, \mathcal{R})$$

with one of the two inequalities being strict.

3 Traditional sequential approach

The traditional approach is a combination of algorithms which have been described in the literature. It goes through line planning, timetabling, and vehicle scheduling sequentially and finds (close to) optimal solutions in each of the steps.

Step L: Line planning. There exists a variety of algorithms for line planning, see [21]. Some of them assume a line pool to be given, others determine the lines during their execution ([2]). If a line pool is required, a line pool generation procedure can be used (see [7] and references therein).
In our experiments: We use the cost model for a fixed line pool which is either given (dataset Bahn) or generated by [7] (dataset Grid).

Step T: Timetabling. Solving the integer programming formulations is too time-consuming for most instances, hence often heuristics ([9, 15, 16]) are used.
In our experiments: We use the fast MATCH heuristic [16].

Step S: Vehicle scheduling. There exists a variety of algorithms, see [4].
In our experiments: We use the flow-based model of [4].

We remark that even if all three steps are solved optimally, the resulting LTS-plan need not be Pareto optimal. This is due to the sequential approach: the line plan is the basis for the timetable and the vehicle schedule, but optimal lines cannot be determined without knowing the optimal timetable and the optimal vehicle schedule.

4 Look-ahead enhancements

As already mentioned, the vehicle schedules have a large impact on the costs of an LTS-plan. Since the vehicle schedules are determined only in the last of the three considered planning stages, the costs of an LTS-plan determined by the sequential approach are usually not minimal. We propose three enhancements in order to receive LTS-plans with better costs than in the sequential approach. We nevertheless also evaluate the perceived travel times for the passengers.

4.1 Using new costs in the line planning step

When evaluating the costs of an LTS-plan, (4) shows that the costs are determined to a large amount by the number of vehicles needed. Even if as few lines as possible are established it is not clear how many vehicles are needed in the end and how many empty kilometers are necessary.

In the traditional approach the costs of a line are usually assumed to be proportional to its length with some fixed costs to be added, i.e.,

$$\overline{\text{cost}}_l = \text{cost}_{\text{fix}} + c \cdot \text{length}_l \tag{6}$$

where $\text{cost}_{\text{fix}} \in \mathbb{R}_+$ and $c \in \mathbb{R}_+$ is a scaling factor.

Here, we now try to compute the costs of a line as closely as possible to the costs it may have later in the evaluation of the LTS-plan. The idea is to approximate the costs per line by distributing the costs specified in (4) to the lines and computing the costs per period, i.e., we want to get

$$g^{\text{cost}} \approx \sum_{l \in \mathcal{L}} \text{cost}_l \cdot \text{p}.$$

For full duration and distance this can be done straightforwardly, as we only need to know the number of planning periods which are considered in total as the length and duration of a line does not change between periods. Under our assumptions, we know the duration of a line beforehand by (3). The number of vehicles needed, the empty distance and the empty duration are in general more difficult to approximate as they can differ between the planning periods due to an aperiodic vehicle schedule. As upper bound we use a very simple vehicle schedule where all vehicles periodically cover only one line and its backwards direction. This gives us that the empty distance is always zero and can be neglected. The empty duration of a line can be computed as

$$\text{empty duration after driving on line } l = \frac{T}{2} - (\text{dur}_l \mod \frac{T}{2}),$$

and for a given minimal turnaround time $L_{\min}$ of a vehicle, the number of vehicles needed to serve a line and its backwards direction can be approximated by

$$\#\text{vehicles needed for line } l \text{ and backwards direction} = \lceil 2 \cdot (\text{dur}_l + L_{\min})/T \rceil .$$

Summarizing, we can approximate the line costs as:

$$\text{cost}_l = 2 \cdot c_1 \cdot \text{dur}_l + 2 \cdot c_2 \cdot \text{length}_l + \frac{c_3}{\text{p}} \cdot \left\lceil 2 \cdot \frac{\text{dur}_l + L_{\min}}{T} \right\rceil + 2 \cdot c_4 \cdot \left(\frac{T}{2} - \text{dur}_l \mod \frac{T}{2} \right). \tag{7}$$

4.2 Line pool generation with look-ahead

The next idea is to take account of good vehicle schedules already in the very first step: we construct the lines in the line pool in a way such that no empty kilometers are needed and that the resulting lines are likely to be operated with a small number of vehicles.

To create a line pool which already considers the vehicle routing aspect, we modified the line pool generation algorithm described in [7]. For a given minimal turnaround time $L_{\min}$ of a vehicle and a maximal allowed buffer time α we ensure that the duration dur_l as defined in (3) of a line l satisfies

$$\frac{T}{2} - L_{\min} - \alpha \leq \mathrm{dur}_l \mod \frac{T}{2} \leq \frac{T}{2} - L_{\min}. \tag{8}$$

Here, the duration of a line is computed according to the minimal driving time on edges and the minimal waiting time in stops. Equation (8) ensures that at the end of a trip, i.e., the driving of a line, the vehicle has enough time to start the trip belonging to the backwards direction of the same line and has to wait no more than α minutes to do so. Thus, we get that the round-trip of forward and backward direction together differs from an integer multiple of the period length by at most $2 \cdot \alpha$.

4.3 Vehicle scheduling first

In our last suggestion we propose to switch Step T and Step S in the sequential approach, i.e., to find (preliminary) vehicle schedules directly after the line planning phase. This is particularly interesting if the line plan contains lines which can be operated efficiently by one vehicle, i.e., lines with small α, since it ensures that the timetable will not destroy this property. This is done as follows:

Step L: This step is done as in the traditional approach.

S-first: For every line l we introduce *turnaround* activities in the periodic event-activity network between the last arrival event of the line in forward direction and the first departure event of the line in backward direction, and vice versa. The lower bound for these activities is set to $L_{\min}$ and the upper bound to $L_{\min} + 2 \cdot \alpha$. These activities ensure that the timetable to be constructed in the next step allows the vehicle schedule we want, namely that only one vehicle operates the line.

Step T: We then proceed with timetabling as in the traditional approach but respecting the turnaround activities such that the resulting timetable does not destroy the desired vehicle schedule.

Step S: After timetabling we perform an additional vehicle scheduling step as in the classic approach: We delete the turnaround activities and proceed with vehicle scheduling as usual. Nevertheless, it is likely, that many of the vehicle routes already determined in S-first will be found again.

Note that S-first can be performed very efficiently in the number of lines in the line concept. We furthermore remark that for a line plan in which all lines have a buffer time $\alpha = 0$, the Step S can be omitted since having line-pure vehicle schedules is an optimal solution in such a case. Even if not all lines have zero buffer times, fixing a timetable in Step T with respecting the turnaround activities often already determines the optimal vehicle schedule. This means that vehicle scheduling in Step S is often redundant, which was not only observable in most cases of our experiments, but is also illustrated more precisely in Example 1 of the appendix.

5 Experiments

We compared the traditional approach for finding an LTS-plan against the enhancements proposed using LinTim, a software framework for public transport optimization [1, 8]. We use the following parameters to describe the different combinations of our enhancements.

1. Using the new costs (7) in line planning (Step L) as proposed in Section 4.1 is denoted by `new cost`, whereas traditional costs are denoted as `normal cost`.
2. The second option, described in Section 4.2 is to construct a new pool (`new pool`), whereas `normal pool` uses some given (standard) pool for line planning (Step L). Combining both pools has been done in a third option (`combined pool`).
3. The decision of computing the timetable or the vehicle schedules first (so using Step S-first from Section 4.3), is denoted by `TT first` and `VS first` respectively.

As test instances we used two significantly different datasets.

Dataset Grid: A grid graph of 5 by 5 nodes and 40 edges, which is a model for a bus network constructed in [5]. In this example, we have $T = 20$ and we used $p = 24$ periods. The `normal pool` for this instance has been calculated with the tree based heuristic from [7].

Dataset Bahn: This is a close-to-real world instance which consists of 250 stations and 326 edges describing the German ICE network. The period length is $T = 60$, we computed for $p = 32$ periods in order to achieve a reasonable time horizon for vehicle scheduling. Note that p is even larger in practical railway applications. As `normal pool` we used a pool of Deutsche Bahn. For the computations we used a standard notebook with i3-2350M processor and 4 GB of RAM. The computation time for one data point of the Grid dataset did not exceed 3 min, while computing a solution for the Bahn dataset took up to 30 minutes.

5.1 Dataset Grid

Figure 1 shows 12 solutions, one for every combination of our parameters. These are graphed according to travel times (x-axis) and their costs (y-axis). We computed the costs and the travel times of the LTS-plans as described in (4) and in (5). We observe the following:

- The solution of the traditional approach (circle with grey marker, left side filled) is dominated by the solution obtained when replacing `normal pool` by `combined pool`.
- Using `new cost` (black markers) instead of `normal cost` (grey markers) always decreases the costs.
- Using `combined pool` always has better costs than using `new pool` or `normal pool`. The travel times sometimes decrease and sometimes increase.
- The option `TT first` yields better travel times compared to `VS first` while `VS first` always has lower costs than `TT first`.
- There are five non-dominated solutions, four of them computed by using `new cost`. Whenever `new pool` or `combined pool` was used together with `new cost` the resulting solution was non-dominated.

The new pool to be generated depends on the parameter α. In Figure 1, $\alpha = 3$ was used. We also tested the parameters $\alpha = 2, 3, \ldots, 10$ for all combinations. The result is depicted in Figure 2. Note that $\alpha \geq 10$ implies no restrictions on the line lengths.

The basic findings described for $\alpha = 3$ remain valid also for other line pools generated: Solutions generated with `new cost` have lower costs while solutions generated with `normal cost` have smaller travel times. The leftmost solutions correspond to `TT first` and bottom-most solutions correspond to `VS first`. In fact, for every single LTS-plan that has been

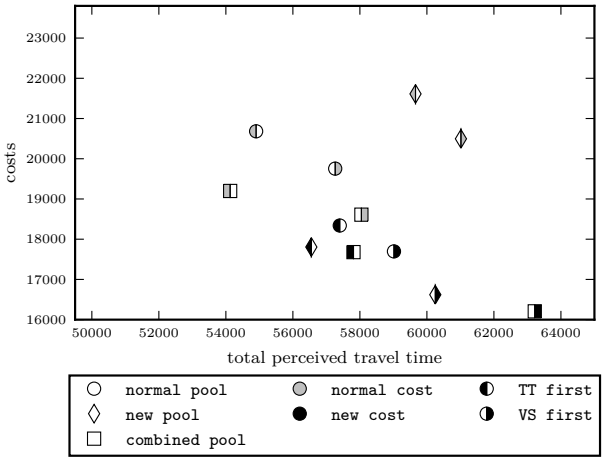


Figure 1 Different combinations of look-ahead steps.

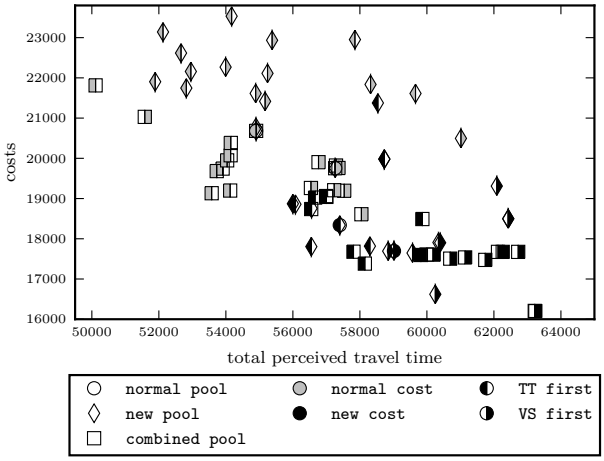


Figure 2 Different combinations of look-ahead steps and different choices for α.

computed, `VS first` yielded a cheaper solution than `TT first` while the latter resulted in a solution with smaller travel time than `VS first`. Finally, none of the solutions computed by using `normal pool` is non-dominated; the Pareto front (i.e., the non-dominated solutions) consists mostly of squares, i.e., solutions generated with `combined pool`. Nevertheless, we see that the quality of the solution obtained depends significantly on the choice of the parameter α. This is investigated in Figure 3.

First of all, we again see that for every fixed α `new cost` yields better solutions than `normal cost` and that the `combined pool` always yields lower costs than `new pool`. If all three look-ahead enhancements `new cost`, `combined pool` and `VS first` are applied, there is a trend of increasing costs once α increases, corresponding to the conjecture that cheap LTS-plans can be found by a small choice of α. For $\alpha = 0$ and $\alpha = 1$ the restrictions on the line length implied by equation 8 is in this example of a grid graph so strict that no feasible solution is possible.

5.2 Dataset Bahn

Applying the implemented enhancements to Bahn with the parameter choice $\alpha = 10$ (Note that $\alpha = 3$ for $T = 20$ in dataset Grid is similar to $\alpha = 10$ for $T = 60$ in dataset Bahn.) yields the results depicted in Figure 4.

The remarkable thing observable in this scenario is that `new` and `combined pool` lead to drastically vehicle cost reductions of more than 40%, whereas the travel time increases by up to 20%. Next to the fact of `combined pool` leading to better costs also the behaviour of `TT first` against `VS first` remains similar to the Grid instance. One can see that `VS first` saves costs between 1 and 5% and `TT first` decreases the travel time by 1 to 3 %. Since the size of the generated line pool had to be chosen small in comparison to the instance size (because of runtime and memory limitations), also the number of feasible line concepts is comparable small. Therefore, this example did not show any impact of using `normal` or `new cost` to the vehicle scheduling costs.

6 Relation to the Eigenmodel

In [22], it is proposed to use different paths through the Eigenmodel (depicted in Figure 5 in the appendix) when optimizing an LTS-plan. In this model, the traditional approach (`normal cost`, `normal pool`, `TT first`) has been depicted as the blue path starting with line planning, then finding a timetable and finally a vehicle schedule. In this paper we compared this traditional approach to two other paths:

- The approach (`normal cost`, `normal pool`, `VS first`) corresponds to the red path in which first a line planning step is performed, then vehicle schedules are determined and finally a timetable. We have seen that this approach leads to significantly better costs but to a higher travel time.
- The approach (`new cost`, `new pool`, `VS first`) can be interpreted as the green path in which we start with vehicle scheduling (by generating a line pool with small α only containing lines with low vehicle scheduling costs), choose a line plan out of this pool and finally determine a timetable which respects the preferred vehicle schedules. In Figure 1 we see that this approach generated the solution with lowest costs. Neglegting the tiny difference between `normal` and `new cost` this also holds for the Bahn instance.

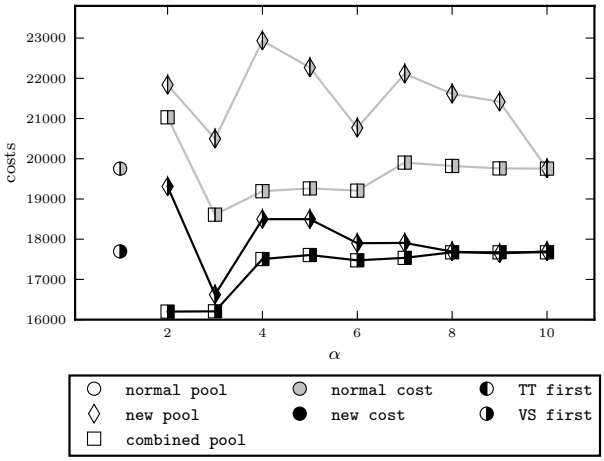


Figure 3 Impact of choice for α.

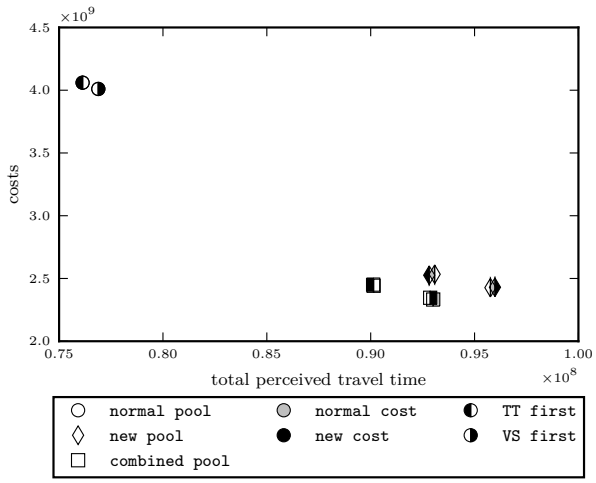


Figure 4 Different combinations of look-ahead steps.

7 Outlook and further research

Summarizing our experiments, all three look-ahead enhancements lead in the majority of cases to a cheaper LTS-plan. Even choosing only one of the approaches will most likely lead to this goal. It is remarkable that the implementation of the proposed algorithmic ideas even performs very well on the Bahn dataset, that has the size and structure of a real world instance. Since exact approaches are far away from solving data sets of this size, the look-ahead heuristic proves itself useful for revealing the strength of considering integrated public transportation optimization.

The presented look-ahead approaches are designed to find a cost-optimized LTS-plan. One could also try to find heuristic approaches focussing on finding a passenger-convenient LTS-plan. A possible step towards this direction would be to choose a different line planning procedure, in order to optimize not with respect to the costs, but for example with respect to the number of direct travelers in the network.

Further research could also be carried out regarding exact approaches of integrated public transportation planning. It would be interesting to investigate different ways of decomposing the integrated problem, in particular, if also routing decisions are included. First results are under research, see [13].

References

1 S. Albert, J. Pätzold, A. Schiewe, P. Schiewe, and A. Schöbel. LinTim – Integrated Optimization in Public Transportation. Homepage. see `http://lintim.math.uni-goettingen.de/`.

2 R. Borndörfer, M. Grötschel, and M. Pfetsch. A Column-Generation Approach to Line Planning in Public Transport. *Transportation Science*, 41:123–132, 2007.

3 R. Borndörfer, H. Hoppmann, and M. Karbstein. Passenger routing for periodic timetable optimization. *Public Transport*, 2016. `doi:10.1007/s12469-016-0132-0`.

4 S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317, 2009.

5 M. Friedrich, M. Hartl, A. Schiewe, and A. Schöbel. Angebotsplanung im öffentlichen Verkehr – planerische und algorithmische Lösungen. In *Heureka'17*, 2017.

6 P. Gattermann, P. Großmann, K. Nachtigall, and A. Schöbel. Integrating Passengers' Routes in Periodic Timetabling: A SAT approach. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/OASIcs.ATMOS.2016.3`.

7 P. Gattermann, J. Harbering, and A. Schöbel. Line pool generation. *Public Transport*, 9(1):7–32, 2017. `doi:10.1007/s12469-016-0127-x`.

8 M. Goerigk, M. Schachtebeck, and A. Schöbel. Evaluating Line Concepts using Travel Times and Robustness: Simulations with the LinTim toolbox. *Public Transport*, 5(3), 2013.

9 M. Goerigk and A. Schöbel. Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, 40(5):1363–1370, 2013.

10 S. Harrod. A tutorial on fundamental model structures for railway timetable optimization. *Surveys in Operations Research and Management Science*, 17:85–96, 2012.

11 C. Liebchen. *Periodic Timetable Optimization in Public Transport*. dissertation.de – Verlag im Internet, Berlin, 2006.

12 C. Liebchen and R. Möhring. The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables – and Beyond. In *Proceedings of 9th meeting on Computer-Aided Scheduling of Public Transport (CASPT 2004)*. Springer, 2004.

13 M. Lübbecke, C. Puchert, P. Schiewe, and A. Schöbel. Detecting structures in network models of integrated traffic planning. Presentation at the Clausthal-Göttingen International Workshop on Simulation Science.

14 K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. PhD thesis, University of Hildesheim, 1998.

15 K. Nachtigall and J. Opitz. Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations. In *Proc. ATMOS*, 2008.

16 J. Pätzold and A. Schöbel. A Matching Approach for Periodic Timetabling. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2016.1.

17 L. Peeters. *Cyclic Railway Timetabling Optimization*. PhD thesis, ERIM, Rotterdam School of Management, 2003.

18 L. Peeters and L. Kroon. A Cycle Based Optimization Model for the Cyclic Railway Timetabling Problem. In S. Voß and J. Daduna, editors, *Computer-Aided Transit Scheduling*, volume 505 of *Lecture Notes in Economics and Mathematical systems*, pages 275–296. Springer, 2001.

19 M. Schmidt. *Integrating Routing Decisions in Public Transportation Problems*, volume 89 of *Optimization and Its Applications*. Springer, 2014.

20 M. Schmidt and A. Schöbel. Timetabling with passenger routing. *OR Spectrum*, 37:75–97, 2015.

21 A. Schöbel. Line planning in public transportation: models and methods. *OR Spectrum*, 34(3):491–510, 2012.

22 A. Schöbel. An Eigenmodel for Iterative Line Planning, Timetabling and Vehicle Scheduling in Public Transportation. *Transportation Research C*, 74:348–365, 2017. doi:10.1016/j.trc.2016.11.018.

23 P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematic*, 2:550–581, 1989.

A Appendix

The following example shows that it is unlikely to find a better vehicle schedule in Step S.

▶ **Example 1.** Consider two lines l_1 and l_2 such that line l_1 ends at the station that l_2 starts at as shown in Figure 6.

Let the duration of the lines be $\text{dur}_{l_1} = \frac{T}{2} + \epsilon$ and $\text{dur}_{l_2} = \frac{T}{2} - \epsilon$ such that $\text{dur}_{l_1} + \text{dur}_{l_2} = T$. Then using S-first with $L_{\min} = 0$ we will need two vehicles vehicles to serve line l_1 and an additional vehicle to serve line l_2, as the following computation shows. The corresponding vehicle schedule can be seen in Figure 7.

$$\left\lceil \frac{2 \cdot (\frac{T}{2} + \epsilon)}{T} \right\rceil = \left\lceil \frac{T + 2 \cdot \epsilon}{T} \right\rceil = 2$$

$$\left\lceil \frac{2 \cdot (\frac{T}{2} - \epsilon)}{T} \right\rceil = \left\lceil \frac{T - 2 \cdot \epsilon}{T} \right\rceil = 1$$

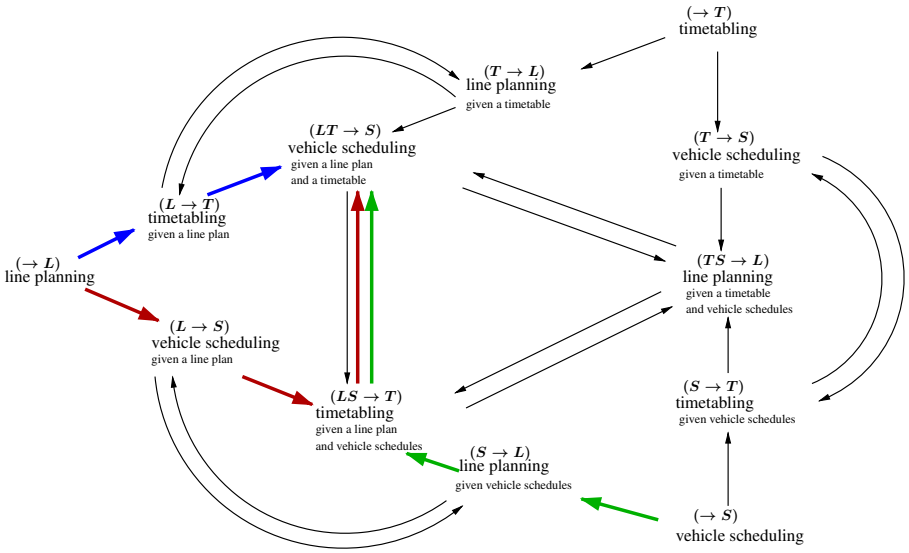


Figure 5 The paths investigated in the Eigenmodel.

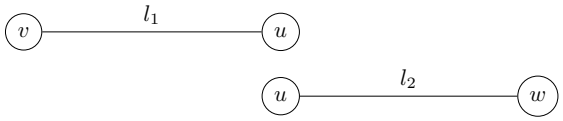


Figure 6 Lines overlapping at station u.

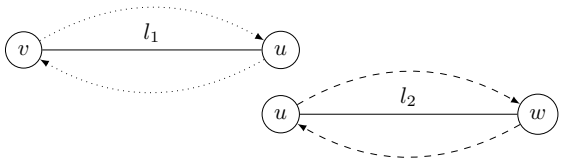


Figure 7 Vehicle schedule derived by S-first.

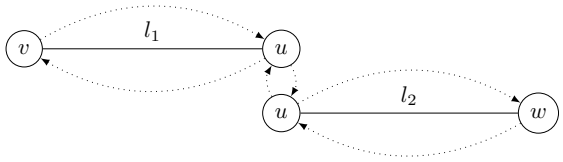


Figure 8 Optimal vehicle schedule.

However, both lines could also be served consecutively by the same vehicle, leading to a total of two instead of three vehicles as can be seen in Figure 8.

$$\left\lceil \frac{2 \cdot (\frac{T}{2} + \epsilon + \frac{T}{2} - \epsilon)}{T} \right\rceil = \left\lceil \frac{2 \cdot T}{T} \right\rceil = 2\,.$$

Nevertheless, it is very unlikely that this vehicle schedule is possible after the timetabling stage T. Consider an OD-pair from v to w. These passengers have to transfer at station u with a minimal transfer time of $\epsilon' > 0$. Then, during the timetabling stage (Step T), the lines will be synchronized such that the passengers can transfer at station u. Therefore, the vehicle schedule shown in Figure 8 will also need three vehicles:

$$\left\lceil \frac{2 \cdot (\frac{T}{2} + \epsilon + \frac{T}{2} - \epsilon + \epsilon')}{T} \right\rceil = \left\lceil \frac{2 \cdot T + 2 \cdot \epsilon'}{T} \right\rceil = 3\,.$$

This shows that the vehicle schedule computed in Step S-first is already optimal as the vehicle schedule shown in Figure 7 is still feasible.

B. Cost-Minimal Public Transport Planning

J. Pätzold, A. Schiewe, A. Schöbel
Cost-Minimal Public Transport Planning
Working paper, 2019.
Extension of Pätzold, Schiewe, and Schöbel (2018) (Proceedings of *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, 2018).

Cost-Minimal Public Transport Planning*

Julius Pätzold[1], Alexander Schiewe[2], and Anita Schöbel[3]

[1] *University of Göttingen, Lotzestr. 16-18, 37083 Göttingen, Germany, j.paetzold@math.uni-goettingen.de*
[2] *Technical University of Kaiserslautern, Gottlieb-Daimler-Straße, 67663 Kaiserslautern, Germany, a.schiewe@mathematik.uni-kl.de*
[3] *Technical University of Kaiserslautern, Gottlieb-Daimler-Straße, 67663 Kaiserslautern, Germany, schoebel@mathematik.uni-kl.de*

Abstract

In this paper we investigate cost-optimal public transport plans, i.e., a line plan, a timetable and a vehicle schedule which can be operated with minimal costs while, at the same time, allowing all passengers to travel between their origins and destinations. We are hereby interested in an exact solution of the *integrated* problem. In contrast to a passenger-optimal public transport plan, in which there is a direct connection for every origin-destination pair, the structure or mathematical model for determining a cost-optimal public transport plan is not obvious and has not been researched so far.

We present three models which differ with respect to the structures we are looking for. If lines are directed and may contain circles, we prove that a cost-optimal schedule can (under weak assumptions) already be obtained by first distributing the passengers in a cost-optimal way. We are able to streamline the resulting integer program such that it can be applied to real-world instances. Additionally, solutions to this first model give bounds for the general case. In the second model we look for lines operated in both directions by integrating the line planning stage. We show that this model yields a stronger lower bound than the first one. Our third and most realistic model looks for lines operated in both directions, and allows all structures for the vehicle schedules. This model, although theoretically being capable of determining general cost-optimal public transport plans, is only computable for small instances.

After introducing these three models and proving the mentioned bounds we compare their computational results and solution quality experimentally.

1 Introduction

Public transport planning is a challenging task since it consists of several stages including network design, line planning, timetabling, vehicle- and crew scheduling. In this paper we look for a line plan in combination with a timetable and a

*This work was partially supported by DFG under SCHO 1140/8-1.

vehicle schedule, i.e., a *public transport plan*. Apart from the different subproblems that need to be solved in an integrated way, there are also different objectives to be considered. A public transport plan should be passenger-friendly (mostly reflected by a short traveling time for the passengers) but also have low operating costs. For individual planning stages such as line planning or vehicle scheduling there exist models and algorithms but finding an integrated solution to this multi-stage problem is more challenging.

The goal of integrated planning is to find the set of pareto solutions with respect to costs and traveling time and then to choose a solution from this set that is affordable and good for the passengers. From an academic point of view it is interesting to find theoretical bounds on the two objective function values of the pareto solutions, i.e. finding the best achievable traveling time for the passengers, and finding the minimal costs (under the condition that all passengers can be transported). The former problem can be solved by a *taxi-solution*, providing a direct and fast connection for each origin-destination pair. Nevertheless, what a cost-optimal transportation plan would look like has not been studied so far and does not seem to be obvious.

Our contribution: In this paper we propose models for finding cost-optimal public transport plans. More precisely, for a given public transport network, passengers' demand and a homogeneous fleet with a given vehicle capacity we design a line plan, a timetable, and a vehicle schedule under the constraint that all passengers can be transported, i.e., for each passenger there exists a possible (maybe non-desirable) connection from their origin to their destination such that none of the vehicles is overloaded. The three models presented are increasing in detail and complexity, allowing for quickly solvable approximations as well as a more detailed exact formulation, depending on the need of the planner. For the models computing approximations we prove bounds on their solution quality for the overall problem.

2 Literature Review

Traditionally, computing a public transport plan consists of solving a series of problems in a sequential order, as can be seen in [CW86, DH07, LM04]. A sequential approach, however, is unsatisfactory since the quality of the overall solution is dependent on all stages and can therefore often not be sufficiently approximated in early planning stages. Therefore integrated planning is an ongoing topic in mathematical public transport planning, see for example the recent special issue [MCZT18] and beyond, e.g., [TK00, DC18, KDC18].
Surprisingly, only a few papers *evaluate* both cost and traveling time for integrated public transport plans. A first approach in which line plans, timetables and vehicle schedules have been evaluated together under different criteria has been given in [GSS13]. More recently, [FHSS17] propose to measure costs and traveling time and evaluate public transport plans under these criteria (cf. Figure 7). Given a line pool, [BNP09] determine a line plan such that all origin-destination pairs can travel. The costs for the lines, however, are only approximated and not determined by the vehicle schedule. Furthermore, capacities are neglected. Other approaches often only integrate timetabling and vehicle scheduling while optimizing costs, see [vdHvdAvK08] or [DRB$^+$17].

In contrast to these works, we take an integrated point of view and propose models for finding cost-optimal public transport plans including lines, timetables, and vehicle schedules. Additionally, we aim at solving the integrated system exactly, meaning that we do not provide iterative heuristics as in [BBLV17, Sch17, VKM17] or a sequential approach as in [PSSS17].
For the single planning stages line planning, timetabling, and vehicle scheduling models and algorithms are well-researched. For line planning cost-oriented models (e.g., [Zwa97, CvDZ98, GvHK06]) and passenger-oriented models (e.g., [Bus98, SS06, BGP07]) are known, see [Sch12] for a survey. (Periodic) timetabling focuses on the passengers and is the hardest of the three problems. Exact approaches to this problem can be found in [SU89, Nac98, PK03, Lie06] and heuristics in [NO08, GS13, PS16] and references therein. See [LLER11] for a survey. Integrating the passengers' routes in timetabling is an ongoing problem, see [SS15, GGNS16, BHK17, Sch18]. For vehicle scheduling we refer to the survey in [BK09]. In this paper we consider periodic vehicle scheduling, which is equivalent to aperiodic planning under some assumptions as shown in [BKLL18].

3 A Cost-Optimal Public Transport Plan

In this section we formally describe what a feasible *public transport plan*, consisting of a *line plan*, a *timetable*, and a *vehicle schedule*, is and how its quality can be evaluated. We restrict ourselves to periodic public transport plans (including periodic vehicle scheduling) in this paper.

Notation 1. The following input data is required:

- a public transport network PTN $= (V, E)$ with a set of stops V and direct connections E between them,
- for every edge $e \in E$:
  - a length (in kilometers) length_e,
  - a lower bound on the traveling time along the edge L_e^{drive},
- a lower bound L^{wait} for the time vehicles have to wait at every stop,
- a minimal turnaround time for vehicles L^{turn}, denoting the minimal time a vehicle has to wait at the end of a line. We assume that $L^{\text{wait}} \leq L^{\text{turn}}$.
- an OD-matrix W with entries W_{uv} for each pair of stops $u, v \in V$, denoting how many passengers want to travel from an origin u to the destination v in a representative time period. A pair of stations $u, v \in V$ with $W_{uv} > 0$ is called an OD-pair.
- a capacity Cap being the maximal number of passengers each vehicle can transport,
- cost parameters
  - c_{time} costs per time period for a vehicle,
  - c_{length} costs per kilometer driven by a vehicle per time period.

We assume that the fixed costs (cost of a vehicle, administration, etc.) are included in the costs per time period and costs per kilometer as is often done in practice.

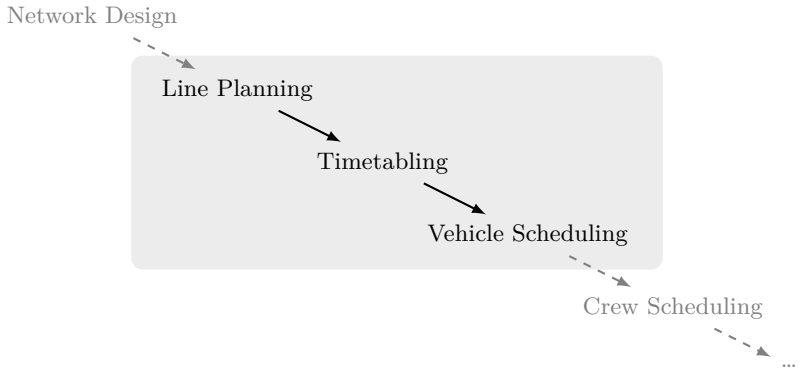


Figure 1: Overview of the sequential planning procedure. The stages integrated here are highlighted with a grey box.

With this input data we then look for a public transport plan whose objects are described next. An overview on the sequential planning approach and the stages integrated here can be found in Figure 1.

Line plan

A *line* is a path in the PTN. A *line plan* is a set of lines $\mathcal{L}$, each of them operated once in the planning period (often an hour). A line plan is *feasible* if every passenger can be transported, i.e., if for every OD-pair (u, v) there exist

- a set of directed paths P_{uv} from u to v, $P_{\text{all}} = \bigcup_{u,v \in V} P_{uv}$
- weights w_p for each path $p \in P_{uv}$

such that $\sum_{p \in P_{uv}} w_p = W_{uv}$ and such that for every edge e it holds that

$$\sum_{p \in P_{\text{all}} : e \in p} w_p \leq \text{Cap} \cdot |\{l \in \mathcal{L} : e \in l\}|. \tag{1}$$

Note, that this notion of feasibility does not require the paths P_{uv} to be good paths for the passengers, but only that all passengers can be transported, not necessarily on their shortest path in the network. See Section 7 for the effects on the computed solutions.
We furthermore have the following requirements on lines.

Requirement 2.

- Lines are simple paths
- Lines are operated in both directions

Note that we do not forbid identical lines, i.e., there may be multiple lines with the same path. In our setting we allow any such path to be a possible line (as also done in [BGP07]) in contrast to many papers which require a line pool of limited size.

Timetable

Given a set of lines $\mathcal{L}$, a timetable assigns a time to every departure and arrival of each line at each of its stops. Determining a (periodic) timetable is the hardest of the three problems line planning, timetabling, and vehicle scheduling, and even finding a feasible timetable that respects the upper and lower bounds on driving, waiting, transfer and turnaround activities is intractable. Since we neglect the passengers, no upper bounds on transfer activities are required and hence a feasible timetable exists for every possible line plan $\mathcal{L}$ (since the timetable for each line can then be determined separately.). Since we are only interested in minimizing the costs we furthermore need not care about optimizing the traveling time of the passengers, meaning that any feasible timetable is sufficient. More precisely, we can neglect the timetabling as a separate planning stage in cost-optimal planning by setting the duration of all drive and wait activities to their lower bounds and simply using the arrival and departure times which are determined by the vehicle schedule.

Vehicle schedule

Given a line plan a *vehicle schedule* determines the number of vehicles and the exact routes of the vehicles for operating the lines. We construct a set of *trips* $\mathcal{L}'$ which contains two directed lines for every (undirected) line $l \in \mathcal{L}$, one in forward and the other one in backward direction.
A route of a vehicle is given by the sequence of (directed) lines it passes,

$$r = (l'_1, \ldots, l'_k),\ l'_i \in \mathcal{L}',$$

whereby requiring all l'_i, $i = 1, \ldots, k$ to be pairwise distinct. We assume that the vehicle, after having taken the last trip l'_k in a route, starts again with l'_1.

This sequence r is interpreted as follows: A vehicle starts with operating line l'_1 at some point in time x. At the end of line l'_1 it drives to the beginning stop of line l'_2, operates this line, and so on. At the end of line l'_k the vehicle returns to the beginning stop of l'_1 and starts again at time y. In order to ensure the required periodicity of the schedule the vehicle needs to start after an integer multiple of the period T, i.e., $y = x + d_r \cdot T$ with d_r being the number of periods needed for a complete operation of the route r.

A vehicle schedule thus consists of a set of routes $\mathcal{R}$. It is *feasible* if each directed line in $\mathcal{L}'$ is contained in exactly one route, i.e., if

$$|\,\{r \in \mathcal{R} : l' \in r\}\,| = 1 \quad \forall l' \in \mathcal{L}'. \tag{2}$$

With these assumptions in place we can now define a *public transport plan.*

Definition 3. A feasible public transport plan is a tuple $(\mathcal{L}, \mathcal{R})$, such that

- $\mathcal{L}$ is a feasible line plan, i.e., it satisfies (1) and the lines conform to Requirement 2,
- $\mathcal{R}$ is a feasible vehicle schedule for the directed lines $\mathcal{L}'$ constructed by the line plan $\mathcal{L}$, i.e., $\mathcal{R}$ satisfies (2).

Costs of a public transport plan

The costs of a public transport plan are given by the distance driven by all vehicles and its total duration. Since we compute a periodic schedule, we consider the costs per planning period T.

A vehicle route r consists of (directed) lines $l' \in \mathcal{L}'$. Hence, we first determine time and duration of a line l', that is

$$\text{length}_{l'} = \sum_{e \in l'} \text{length}_e \tag{3}$$

$$\text{dur}_{l'} = (|l'| - 1)L^{\text{wait}} + \sum_{e \in l'} L_e^{\text{drive}}, \tag{4}$$

where $|l'| := \{e \in E | e \in l'\}$ and (4) uses the fact that it is always cheaper to operate a line as fast as possible. For the empty rides between a pair of lines l'_1 and l'_2 we can use the PTN to determine the parameters

$$\text{length}_{l'_1,l'_2} = \text{length when driving from last station of } l'_1 \text{ to first station of } l'_2$$
$$\text{time}_{l'_1,l'_2} = \text{time for driving from last station of line } l'_1 \text{ to first station of } l'_2$$

The minimum turnaround time (usually accounting for a driver's break) has to be added to the duration of an empty ride. This yields

$$\text{dur}_{l'_1,l'_2} = L^{\text{turn}} + \text{time}_{l'_1,l'_2}. \tag{5}$$

The number of kilometers covered by a given public transport plan is determined by summing up the kilometers of each single route, i.e.,

$$\begin{aligned} \text{length}(\mathcal{L}, \mathcal{R}) &= \sum_{l' \in \mathcal{L}'} \text{length}_{l'} + \sum_{r=(l'_1,\ldots,l'_{k_r}) \in \mathcal{R}} \sum_{i=1}^{k_r} \text{length}_{l'_i,l'_{i+1}} \\ &= \sum_{l \in \mathcal{L}} 2 \cdot \text{length}_l + \sum_{r=(l'_1,\ldots,l'_{k_r}) \in \mathcal{R}} \sum_{i=1}^{k_r} \text{length}_{l'_i,l'_{i+1}} \end{aligned} \tag{6}$$

with $l'_{k_r+1} := l'_1$. The duration of a route $r = (l'_1, \ldots, l'_{k_r}) \in \mathcal{R}$ is measured by the number of time periods dur_r needs. Formally, this can be computed by

$$\text{dur}_r = \left\lceil \sum_{i=1}^{k_r} \text{dur}_{l'_i} + \text{dur}_{l'_i,l'_{i+1}} \right\rceil_T \tag{7}$$

with $\lceil a \rceil_T := \min\{n \in \mathbb{N} | n \cdot T \geq a\}$ for any $a \in \mathbb{R}$ and $l'_{k_r+1} := l'_1$. The overall duration is hence given as

$$\text{dur}(\mathcal{L}, \mathcal{R}) = \sum_{r \in \mathcal{R}} \text{dur}_r. \tag{8}$$

Finally, the cost function is defined as

$$g(\mathcal{L}, \mathcal{R}) := c_{\text{time}} \cdot \text{dur}(\mathcal{L}, \mathcal{R}) + c_{\text{length}} \cdot \text{length}(\mathcal{L}, \mathcal{R}). \tag{9}$$

77

The number of required vehicles is determined by the number of time periods used in $(\mathcal{L}, \mathcal{R})$, i.e., by $\mathrm{dur}(\mathcal{L}, \mathcal{R})$. Once again, any fixed costs per vehicle can be included by being added to c_{time}. Since this does not change the structure of the cost function we assume vehicle costs to already be included in c_{time}.

The cost function defined above allows us to define the optimization problem we are concerned with in this paper.

Problem (cost-opt): Given the input data from Notation 1, find a feasible public transport plan $(\mathcal{L}, \mathcal{R})$, i.e., satisfying Requirement 2, (1) and (2), with minimal costs $g(\mathcal{L}, \mathcal{R})$. We denote the optimal objective value with z^{opt}.

The rest of this paper is structured as follows: In order to find the exact cost minimum of the integrated problem (cost-opt) we present three different models (see Figure 2). The first model, presented in Section 4, aims at distributing the OD-pairs in a cost-optimal way (called *load generation*). Although the first model considers only this very first step, we can show that under certain conditions it already determines the minimal costs of an integrated public transport plan. Section 5 presents the second model that integrates load generation and line planning while minimizing a cost function that approximates (now in greater detail) the costs of a resulting public transport plan. Finally, Section 6 presents a third model, an exact IP formulation for integrating load generation, line planning, timetabling, and vehicle scheduling; it hence provides an exact model for (cost-opt).

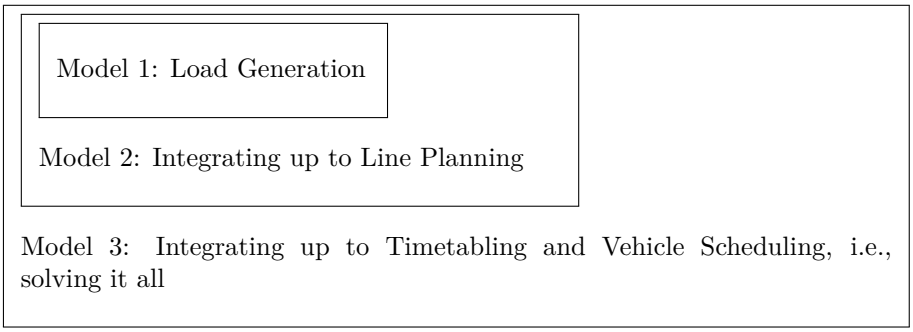


Figure 2: Three proposed models for solving (cost-opt)

4 Model 1: Creating a Cost-Efficient Load

Line planning is often decomposed into two steps. In the first step, a *load* is generated. This is done by routing all OD-pairs (u, v) through the PTN which results in paths P_{uv} with $P_{\text{all}} = \bigcup_{u,v \in V} P_{uv}$ and weights w_p for every path $p \in P_{uv}$ (with $\sum_{p \in P_{uv}} w_p = W_{uv}$). This data is then used to define the load, also known as minimal frequencies:

$$f_e^{\min} = \left\lceil \sum_{p \in P_{\text{all}} : e \in p} w_p \cdot \frac{1}{\text{Cap}} \right\rceil,$$

78

specifying how often an edge $e \in E$ in the PTN has at least to be served by some vehicle. In the second step, the line planning problem, i.e., finding a line plan $\mathcal{L}$ satisfying $f_e^{\min} \leq |\{l \in \mathcal{L} : e \in \mathcal{L}\}|$, is solved using these minimal frequencies.

For our first model we only consider the first one of these two steps: calculating a load. Normally the load $f_e^{\min}$ is calculated assuming that all passengers are able to travel on their shortest path in the PTN to their destination. Since we are interested in finding a cost-minimal public transport plan, we do not want to work with such a fixed assumption. Instead, in our system we want to admit just enough capacities to ensure that every passenger has some possibility to travel to their destination. We use this insight to find a load that eventually even leads to a cost-minimal public transport plan.
Of course, in this early planning stage we do not yet have all information to exactly determine the costs of the resulting public transport plan since they depend on the line plan and the vehicle schedule. Nevertheless, we can already approximate the costs with the following model.

Model 1. Given the input data from Notation 1, calculate a load (i.e., $f_e^{\min}$ for all $e \in E$) that aims at minimizing the cost of a public transport plan.

$$\min \quad c_{\text{time}} \cdot \text{dur} + c_{\text{length}} \sum_{e \in E} 2 \cdot \text{length}_e \cdot f_e^{\min} \tag{10}$$

$$\text{s.t.} \quad \sum_{e \in E} 2 f_e^{\min} (L_e^{\text{drive}} + L^{\text{wait}}) \leq T \cdot \text{dur} \tag{11}$$

$$\sum_{u \in V} f_{(i,j),u} \leq f_e^{\min} \cdot \text{Cap} \qquad \forall i, j \in V \text{ with } \{i,j\} \in E \tag{12}$$

$$\sum_{i \in V : \{i,v\} \in E} f_{(i,v),u} = W_{uv} + \sum_{i \in V : \{v,i\} \in E} f_{(v,i),u} \qquad \forall u \in V \; \forall v \in V \backslash \{u\} \tag{13}$$

$$\sum_{i \in V : \{u,i\} \in E} f_{(u,i),u} = \sum_{v \in V} W_{uv} \qquad \forall u \in V \tag{14}$$

$$\begin{aligned} & f_{(i,j),v} \in \mathbb{N} && \forall \{i,j\} \in E, i, j, v \in V \\ & f_e^{\min} \in \mathbb{N} && \forall e \in E \\ & \text{dur} \in \mathbb{N} \end{aligned} \tag{15}$$

Variables:

- $f_{(i,j),u}$ – number of passengers starting from stop $u \in V$ traveling on arc (i,j) for some $i, j \in V$ with $\{i,j\} \in E$ (non-negative, continuous)
- $f_e^{\min}$ – load for edge e, i.e., how often e has to be covered (integer)
- dur – total duration (counted in periods) (integer)

In this model we define some passenger flow from every stop $u \in V$ in the PTN going to all destinations $v \in V$. In order to not mix up passengers starting from different stops we have to define $|V|$ different flows. The constraints (13)

and (14) describe the flow conservation constraints. In order to restrict the number of passengers traveling on a certain edge in the network we define capacity constraints in (12). Note, that the flow variables $f_{(i,j),u}$ for $u \in V$ are defined on directed edges (i,j) whereas the minimal frequencies $f_e^{\min}$ are defined on undirected edges $\{i,j\} = e \in E$. Finally, constraint (11) rounds up the minimal duration to the next multiple of time period T and the objective function amounts the costs required in the best case, that is, for a vehicle schedule without any empty ride and as less time loss (by the periodicity rounding) as possible. We will call the optimal objective value to this model z_1^{opt}

The following theorem shows that Model 1 is indeed an approximation of (cost-opt)as its optimal solution yields a lower bound.

Theorem 4. *Model 1 is a relaxation of (cost-opt), in particular it holds that*

$$z_1^{\text{opt}} \leq z^{\text{opt}}.$$

Proof. Let $(\mathcal{L}, \mathcal{R})$ be some feasible solution to (cost-opt). Since the line plan $\mathcal{L}$ is feasible, we can construct some feasible flow from it by setting $f_e^{\min} = |\{l \in \mathcal{L} | e \in l\}|$ and $f_{e,u} = \sum_{p \in P_{\text{all}}: e \in p} w_p$ with P_{all} and w_p obtained from (1).
Now we get for all $i, j \in V$ with $\{i,j\} \in E$

$$\sum_{u \in V} f_{(i,j),u} = \sum_{p \in P_{\text{all}}:(i,j) \in p} w_p \underbrace{\leq}_{\text{by (1)}} f_e^{\min} \cdot \text{Cap}$$

by definition of feasibility of a line plan, i.e., constraint (12) is satisfied. Since the w_p correspond to paths in the PTN the flow conservation constraints (13) and (14) are also satisfied. By setting

$$\text{dur} = \left\lceil \frac{\sum_{e \in E} 2 f_e^{\min} (L_e^{\text{drive}} + L^{\text{wait}})}{T} \right\rceil$$

we have constructed a feasible solution to Model 1.
We now show that the objective function value of the constructed solution is better than $g(\mathcal{L}, \mathcal{R}) = c_{\text{time}} \cdot \text{dur}(\mathcal{L}, \mathcal{R}) + c_{\text{length}} \cdot \text{length}(\mathcal{L}, \mathcal{R})$.
We first consider $\text{length}(\mathcal{L}, \mathcal{R})$: We know that for the constructed solution it holds that $f_e^{\min} = |\{l \in \mathcal{L} | e \in l\}|$, hence

$$\text{length}(\mathcal{L}, \mathcal{R}) \underbrace{\geq}_{\text{by (6)}} \sum_{l' \in \mathcal{L}'} \text{length}_{l'} = \sum_{l \in \mathcal{L}} \sum_{e \in l} 2\text{length}_e \geq \sum_{e \in E} 2\text{length}_e f_e^{\min}.$$

For $\text{dur}(\mathcal{L}, \mathcal{R})$ we calculate

$$
\begin{aligned}
\mathrm{dur}(\mathcal{L},\mathcal{R}) \quad &= \quad \sum_{r\in\mathcal{R}} \mathrm{dur}_r = \sum_{r\in\mathcal{R}} \left\lceil \sum_{l'\in r} (\mathrm{dur}_{l'} + L^{\mathrm{turn}}) \right\rceil_T \\
&\geq \quad \left\lceil \sum_{r\in\mathcal{R}} \sum_{l'\in r} (\mathrm{dur}_{l'} + L^{\mathrm{turn}}) \right\rceil_T \\
&\underbrace{=}_{(4)} \quad \left\lceil \sum_{r\in\mathcal{R}} \sum_{l'\in r} \left((|l|-1)L^{\mathrm{wait}} + L^{\mathrm{turn}} + \sum_{e\in l'} L_e^{\mathrm{drive}} \right) \right\rceil_T \\
&= \quad \left\lceil \sum_{l'\in\mathcal{L}} \left(L^{\mathrm{turn}} - L^{\mathrm{wait}} + \sum_{e\in l'} (L_e^{\mathrm{drive}} + L^{\mathrm{wait}}) \right) \right\rceil_T \\
&\geq \quad \left\lceil \sum_{l\in\mathcal{L}} 2 \left((\underbrace{L^{\mathrm{turn}} - L^{\mathrm{wait}}}_{\geq 0}) + \sum_{e\in l} (L_e^{\mathrm{drive}} + L^{\mathrm{wait}}) \right) \right\rceil_T \\
&\underbrace{\geq}_{f_e^{\min} = |\{l\in\mathcal{L} | e\in l\}|} \left\lceil \sum_{e\in E} 2 f_e^{\min} (L_e^{\mathrm{drive}} + L^{\mathrm{wait}}) \right\rceil_T = \mathrm{dur}.
\end{aligned}
$$

Overall it holds that

$$
\begin{aligned}
g(\mathcal{L},\mathcal{R}) &= c_{\mathrm{time}} \mathrm{dur}(\mathcal{L},\mathcal{R}) + c_{\mathrm{length}} \mathrm{length}(\mathcal{L},\mathcal{R}) \\
&\geq c_{\mathrm{time}} \mathrm{dur} + c_{\mathrm{length}} \sum_{e\in E} 2\mathrm{length}_e \cdot f_e^{\min}.
\end{aligned}
$$

Thus every feasible solution to (cost-opt) can be transformed to a solution for Model 1 whose objective is smaller than $g(\mathcal{L},\mathcal{R})$. Hence, Model 1 is a relaxation of (cost-opt). □

For large problem instances a speed-up of the solution process is possible by adding the following valid inequalities to Model 1.

Lemma 5. *Let (X, Y) be some cut, i.e., some disjoint node partition in the PTN with $E_{cut} = \{\{i,j\} = e \in E | i \in X \text{ and } j \in Y\}$ being all cut edges. Then it holds that*

$$
\sum_{u\in X} \sum_{v\in Y} W_{uv} \leq Cap \cdot \sum_{e\in E_{cut}} f_e^{\min}.
$$

Proof. We start with constraint (13), i.e.,

$$
\sum_{i\in V:\{i,v\}\in E} f_{(i,v),u} = W_{uv} + \sum_{i\in V:\{v,i\}\in E} f_{(v,i),u} \quad \forall u \in V \forall v \in V\backslash\{u\}
$$

and argue that for any $u \in X$ it holds that

$$\sum_{v\in Y}\sum_{i\in V:\{i,v\}\in E} f_{(i,v),u} = \sum_{v\in Y}\left(W_{uv} + \sum_{i\in V:\{v,i\}\in E} f_{(v,i),u}\right)$$

$$\underbrace{\Leftrightarrow}_{V=X\cup Y} \quad \sum_{v\in Y}\left(\sum_{i\in X:\{i,v\}\in E} f_{(i,v),u} + \sum_{i\in Y:\{i,v\}\in E} \underbrace{f_{(i,v),u}}_{=(*)}\right)$$

$$= \sum_{v\in Y}\left(W_{uv} + \sum_{i\in X:\{v,i\}\in E} f_{(v,i),u} + \sum_{i\in Y:\{v,i\}\in E} \underbrace{f_{(v,i),u}}_{=(*)}\right)$$

$$\underbrace{\Leftrightarrow}_{(*)\text{ cancel out}} \sum_{v\in Y}\sum_{i\in X:\{i,v\}\in E} f_{(i,v),u} = \sum_{v\in Y}\left(W_{uv} + \sum_{i\in X:\{v,i\}\in E} f_{(v,i),u}\right)$$

$$\Leftrightarrow \quad \sum_{\substack{v\in Y, i\in X:\\ \{v,i\}\in E_{cut}}} f_{(i,v),u} = \sum_{v\in Y} W_{uv} + \sum_{\substack{v\in Y, i\in X:\\ \{v,i\}\in E_{cut}}} f_{(v,i),u}$$

Hence we can conclude

$$\sum_{i\in X, v\in Y:\{v,i\}\in E_{cut}} f_{(i,v),u} \geq \sum_{v\in Y} W_{uv} \quad \forall u \in X. \tag{16}$$

Thus we get that

$$\text{Cap}\cdot \sum_{e\in E_{cut}} f_e^{\min} \underbrace{\geq}_{(12)} \sum_{\substack{i\in X, v\in Y:\\ \{i,v\}\in E_{cut}}} \sum_{u\in V} f_{(i,v),u}$$

$$\underbrace{\geq}_{X\subseteq V} \sum_{u\in X}\sum_{\substack{i\in X, v\in Y:\\ \{i,v\}\in E_{cut}}} f_{(i,v),u} \underbrace{\geq}_{(16)} \sum_{u\in X}\sum_{v\in Y} W_{uv}.$$

□

In the computational experiments, see Section 7, we investigated adding these valid inequalities, which resulted in an improvement of the runtime of up to 50%.
In order to find an upper bound for (cost-opt) instead of a lower bound, we slightly modify Model 1.

Definition 6. We define an adjusted version of Model 1, where L^{wait} is replaced by L^{turn} in constraint (11), to be Model 1*. We call the optimal objective value of this model $z_{1^*}^{\text{opt}}$.

Using this new model, we are able to compute an upper bound to (cost-opt). Note, that in the following of this chapter we always assume the graph $G := (V, \bar{E})$ with $\bar{E} = \{e \in E\colon f_e^{\min} > 0\}$ to be connected for an optimal solution to Model 1*. This is the case, for example, when the graph (V, W') with $W' = \{\{u, v\} \subseteq V : W_{uv} > 0\}$ of the OD pairs is connected.

Theorem 7. *If G is connected, then for every feasible solution to Model 1* there exists a feasible solution to (cost-opt) with the same objective value, in particular it holds that*

$$z^{\text{opt}} \leq z_{1^*}^{\text{opt}}$$

Proof. For every solution to Model 1*, i.e., for some feasible $(f^{\min}, f)$, we can construct some feasible solution $(\mathcal{L}, \mathcal{R})$ to (cost-opt) as follows: We define the line plan $\mathcal{L}$ that contains for each edge $e \in E$ exactly $f_e^{\min}$ lines containing exactly this one edge e, i.e., $\mathcal{L} := \{e^1, \dots, e^{f_e^{\min}} : e \in E\}$. Since $f_e^{\min} = |\{l \in \mathcal{L} | e \in l\}|$ and $f_e^{\min}$ admits a feasible load, e.g., corresponding to f, the line plan $\mathcal{L}$ is feasible.
For this line plan we now generate a vehicle schedule $\mathcal{R}$ that consists of only one large route. To this end, we consider the resulting set of directed lines $\mathcal{L}'$

$$\mathcal{L}' = \left\{(i,j)^1, \dots, (i,j)^{f_e^{\min}}, (j,i)^1, \dots, (j,i)^{f_e^{\min}} : e = \{i,j\} \in E\right\}$$

which contains $f_e^{\min}$ copies of both directions of every edge $e \in E$. This is a set of directed edges which creates a directed multigraph $(V, \mathcal{L}')$. Due to the assumption that $G = (V, \bar{E})$ with $\bar{E} = \{e \in E \colon f_e^{\min} > 0\}$ is connected, this graph is strongly connected and every node in $(V, \mathcal{L}')$ has the same indegree as outdegree. Hence we can find an Eulerian Cycle on it (see, e.g., [Fle91]). This means that we can form a route containing all directed lines $r = (l'_1, \dots, l'_k)$ (with $|r| = |\mathcal{L}'|$) such that $\text{length}_{l'_i, l'_{i+1}} = 0$ and $\text{time}_{l'_i, l'_{i+1}} = 0$. We set the vehicle schedule $\mathcal{R} = \{r\}$ to contain exactly this route r.
We hence have constructed some solution $(\mathcal{L}, \mathcal{R})$ to (cost-opt) with

$$\begin{aligned}\text{length}(\mathcal{L}, \mathcal{R}) &= \sum_{l \in \mathcal{L}'} \text{length}_l + \sum_{r=(l'_1, \dots, l'_{k_r}) \in \mathcal{R}} \sum_{i=1}^{k_r} \underbrace{\text{length}_{l'_i, l'_{i+1}}}_{=0} \\ &= \sum_{l \in \mathcal{L}} 2 \cdot \text{length}_l \underbrace{=}_{f_e^{\min} = \{e \in \mathcal{L} | e \in l\}} \sum_{e \in E} 2\text{length}_e f_e^{\min}\end{aligned}$$

and

$$\begin{aligned}\text{dur}(\mathcal{L}, \mathcal{R}) &= \sum_{r \in R} \text{dur}_r \underbrace{=}_{|\mathcal{R}|=1} \left\lceil \sum_{l \in \mathcal{L}'} (\text{dur}_l + L^{\text{turn}}) \right\rceil_T \\ &\underbrace{=}_{f_e^{\min} = \{e \in \mathcal{L} | e \in l\}} \left\lceil \sum_{e \in E} 2 f_e^{\min} (L_e^{\text{drive}} + L^{\text{turn}}) \right\rceil_T = \text{dur}.\end{aligned}$$

Hence, for every solution to Model 1 we can construct a solution $(\mathcal{L}, \mathcal{R})$ to (cost-opt) such that $g(\mathcal{L}, \mathcal{R}) = c_{\text{time}} \text{dur} + c_{\text{length}} \sum_{e \in E} 2\text{length}_e \cdot f_e^{\min}$. Together with Theorem 4 the solution $(\mathcal{L}, \mathcal{R})$ is optimal for (cost-opt) and hence Model 1 has the same objective value as (cost-opt). □

We can now compute a gap between Model 1 and Model 1*. This allows us to estimate the objective value to (cost-opt) by only computing a solution to Model 1.

Theorem 8. *Let* $(\mathrm{dur}, f, f^{\min})$ *be an optimal solution to Model 1. Then the gap between the optimal objective values to Model 1 and Model 1* is bounded, i.e.,*

$$z_{1^*}^{\mathrm{opt}} - z_1^{\mathrm{opt}} \leq \left\lceil 2 \cdot \frac{L^{\mathrm{turn}} - L^{\mathrm{wait}}}{T} \cdot \sum_{e \in E} f_e^{\min} \right\rceil \cdot c_{\mathrm{time}}.$$

Proof. Let $(\mathrm{dur}, f, f^{\min})$ be an optimal solution to Model 1. For every optimal solution to Model 1, it holds that

$$\mathrm{dur} = \left\lceil \frac{\sum_{e \in E} 2 \cdot f_e^{\min} \cdot (L_e^{\mathrm{drive}} + L^{\mathrm{wait}})}{T} \right\rceil.$$

By setting

$$\mathrm{dur}^* := \left\lceil \frac{\sum_{e \in E} 2 \cdot f_e^{\min} \cdot (L_e^{\mathrm{drive}} + L^{\mathrm{turn}})}{T} \right\rceil,$$

$(\mathrm{dur}, f, f^{\min})$ can be transformed to a feasible solution $(\mathrm{dur}^*, f, f^{\min})$ for Model 1*. With this and the fact that $\lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil$ for all $x, y \in \mathbb{R}$ it holds that

$$\begin{aligned} z_{1^*}^{\mathrm{opt}} - z_1^{\mathrm{opt}} &= c_{\mathrm{time}} \cdot \Bigg(\left\lceil \frac{\sum_{e \in E} 2 \cdot f_e^{\min} \cdot (L_e^{\mathrm{drive}} + L^{\mathrm{turn}})}{T} \right\rceil \\ &\qquad - \left\lceil \frac{\sum_{e \in E} 2 \cdot f_e^{\min} \cdot (L_e^{\mathrm{drive}} + L^{\mathrm{wait}})}{T} \right\rceil \Bigg) \\ &\leq c_{\mathrm{time}} \cdot \left\lceil \frac{\sum_{e \in E} 2 \cdot f_e^{\min} \cdot (L^{\mathrm{turn}} - L^{\mathrm{wait}})}{T} \right\rceil. \end{aligned}$$

□

This bound can be extended to a gap to (cost-opt).

Corollary 9. *The absolute error of solving Model 1 or Model 1* is bounded by*

$$z_{1^*}^{\mathrm{opt}} - z^{\mathrm{opt}} \leq \left\lceil 2 \cdot \frac{L^{\mathrm{turn}} - L^{\mathrm{wait}}}{T} \cdot \sum_{e \in E} f_e^{\min} \right\rceil \cdot c_{\mathrm{time}}$$

$$z^{\mathrm{opt}} - z_1^{\mathrm{opt}} \leq \left\lceil 2 \cdot \frac{L^{\mathrm{turn}} - L^{\mathrm{wait}}}{T} \cdot \sum_{e \in E} f_e^{\min} \right\rceil \cdot c_{\mathrm{time}}.$$

Additionally, this bound allows an optimality condition, where the optimal objective value of Model 1 and Model 1* is the optimal objective value of (cost-opt).

Corollary 10. *Let* $L^{\mathrm{wait}} = L^{\mathrm{turn}}$*. Then the optimal objective of Model 1 and Model 1* is equal to the optimal objective of (cost-opt).*

If we allow that lines do not have to be bidirectional and simple paths in the PTN, i.e., dropping Requirement 2, we are able to show even more: We will always obtain an optimal solution to (cost-opt) by just solving Model 1. This can be done by converting the Eulerian Cycle constructed in the proof of Theorem 7 into one big line.

84

Corollary 11. *Let $L^{\text{wait}} \leq L^{\text{turn}}$. Then the optimal objective value of Model 1 is equal to the optimal objective of (cost-opt) if we drop Requirement 2.*

This, of course, may lead to non-practical lines, as can be seen in the following example.

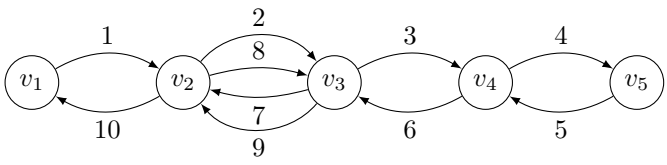


Figure 3: Solution of Model 1 for Example 12

Example 12. We examine the solution provided by Corollary 11 on a small example. Consider the PTN given in Figure 3, with Cap passengers traveling from v_1 to v_5 and 1 passenger traveling from v_2 to v_3. Then the solution provided by Model 1 is given by lower bounds of $[1, 2, 1, 1]$ and the vehicle schedule of Corollary 11 is depicted in Figure 3, where the edges are numbered in order of their usage. As can be seen here, the resulting line structure, that is, if the whole vehicle schedule is transformed into a single line, is not suitable for a practical public transport system, since it contains a cycle.

5 Model 2: Integrating Load Generation and Line Planning

Although we can already find a solution to (cost-opt) using Model 1, it is only cost-minimal in the case of $L^{\text{wait}} = L^{\text{turn}}$. For $L^{\text{wait}} < L^{\text{turn}}$, however, we have seen that if we want to obtain a cost-minimal solution, the resulting line plan may consist of directed lines (without their symmetric counterparts) and the lines may contain circles. We hence want to incporate the next steps of public transport planning to resolve this issue and ensure that the lines satisfy the usual requirements. To this end, we combine the load generation of Model 1 with line planning to improve the approximation of the cost objective of the overall plan. This idea is approached by the following model.

Model 2. Given the input data from Notation 1, calculate a load $f_e^{\min}$ and a line plan $\mathcal{L}$ that aim at minimizing the costs of a public transport plan.

$$\min \quad c_{\text{time}} \cdot \text{dur} + c_{\text{length}} \sum_{l \in [L]} \sum_{e \in E} 2 x_{e,l} \text{length}_e \tag{17}$$

s.t. (12) - (14)

$$\sum_{l \in [L]} \left(2 z_l (L^{\text{turn}} - L^{\text{wait}}) + \sum_{e \in E} 2 (L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l} \right) \leq \text{dur} \cdot T \tag{18}$$

$$\sum_{l \in [L]} x_{e,l} \geq f_e^{\min} \quad \forall e \in E \tag{19}$$

$$x_{e,l} \leq z_l \quad \forall e \in E \, \forall l \in [L] \tag{20}$$

$$\sum_{e \in E} x_{e,l} \geq z_l \quad \forall l \in [L] \tag{21}$$

$$\sum_{e \in E : s \in e} x_{e,l} \leq 2 \quad \forall s \in V \, \forall l \in [L] \tag{22}$$

$$2 x_{e,l} \leq y_{i,l} + y_{j,l} \quad \forall l \in [L] \, \forall (i,j) = e \in E \tag{23}$$

$$\sum_{s \in V} y_{s,l} = \sum_{e \in E} x_{e,l} + z_l \quad \forall l \in [L] \tag{24}$$

$$\sum_{(i,j) = e \in E : i \in C \text{ and } j \in C} x_{e,l} \leq |C| - 1 \qquad \forall \text{ circles } C \subseteq E \, \forall l \in [L] \tag{25}$$

$f_{(i,j),v} \in \mathbb{N} \quad \forall \{i,j\} \in E, i, j, v \in V$

$f_e^{\min} \in \mathbb{N} \quad \forall e \in E$

$\text{dur} \in \mathbb{N}$

$z_l \in \{0,1\} \quad \forall \in [L]$

$y_{s,l} \quad \forall s \in V, l \in [L]$

$x_{e,l} \quad \forall e \in E, l \in [L]$

Coefficients:

- L – maximal possible number of lines (integer) and $[L] := \{1, ..., L\}$.

Variables:

- z_l – is 1 iff line l is non-empty. (binary)
- $y_{s,l}$ – is 1 iff stop s is contained in line l. (binary)
- $x_{e,l}$ – is 1 iff edge e is contained in line l. (binary)
- dur – total duration of all lines (counted in periods) (integer)
- $f_e^{\min}$ – as in Model 1, including the variables $f_{(i,j),u}$ and constraints (12) - (14) from Model 1.

This model finds some feasible line plan. First the z_l-variables determine if line number l is a line or empty. Constraint (20) and (21) ensure this. Now we need for every index l that for every stop of some line there are at most two incident edges (constraint (22)). This ensures that the $x_{e,l}$ variables form circles

86

or paths. To ensure that they form only one connected path we could consider them as flow variables. Here, we decided to add y-variables for every visited stop and count the number of stops that a line visits. The y-variables are set to one for the incident nodes of all edges the line visits in (23). We then can ensure that there is some connected path by requiring that there exists exactly one more stop than edges in a line in constraint (24). Finally we need to rule out subtours which is done by constraint (25) (As usual they are added by constraint generation procedures). The variables $f_e^{\min}$ taken from Model 1 help us to determine feasibility of the line plan, which is done by constraint (19). Finally we round up the duration to the next multiple of a time period, which is done by (18). We call an optimal objective value to this model z_2^{opt}.

The objective function is again a lower bound on the exact costs of a public transport plan which is shown in the next theorem. Note, that the choice of the size of L is crucial for the quality of the model and will be discussed later.

Theorem 13. *For sufficiently large L, the optimal objective value of Model 2 is a lower bound on the optimal objective value of (cost-opt) and an upper bound to the optimal objective value of Model 1, in particular it holds that*

$$z_1^{\text{opt}} \leq z_2^{\text{opt}} \leq z^{\text{opt}}.$$

Proof. Let $(\mathcal{L}, \mathcal{R})$ be some feasible solution to (cost-opt). Then we know that we can set $f_e^{\min} = |\{l \in \mathcal{L} | e \in l\}|$ (and $f_{e,u}$ accordingly) as in the proof of Theorem 4 to some feasible flow which satisfies (19). Furthermore we can enumerate all lines with some bijective mapping $\varphi : \mathcal{L} \to [|\mathcal{L}|]$ such that $x_{e,\varphi(l)} = 1$ iff $e \in l$ for all $l \in \mathcal{L}$ and also $y_{s,\varphi(l)} = 1$ iff $s \in e$ for some $e \in l$. Finally, we have to set $z_i = 1$ for all $i \in [|\mathcal{L}|]$ and 0 for all $i \in [L] \backslash [|\mathcal{L}|]$. Since $\mathcal{L}$ was some feasible line plan, all lines are simple paths and hence also constraints (20) to (25) are satisfied. Now for the objective function it holds that

$$\begin{aligned}
\text{length}(\mathcal{L}, \mathcal{R}) &= \sum_{l' \in \mathcal{L}'} \text{length}_{l'} + \sum_{r=(l'_1,\dots,l'_{k_r}) \in \mathcal{R}} \sum_{i=1}^{k_r - 1} \text{length}_{l'_i, l'_{i+1}} \\
&\geq \sum_{l \in \mathcal{L}} \sum_{e \in l} 2\text{length}_e = \sum_{l \in \mathcal{L}} \sum_{e \in E} 2x_{e,\varphi(l)} \text{length}_e = \sum_{l \in [L]} \sum_{e \in E} 2x_{e,l} \text{length}_e.
\end{aligned}$$

For the duration we get

$$\begin{aligned}
\text{dur}(\mathcal{L}, \mathcal{R}) &= \sum_{r=(l'_1,\dots,l'_{k_r}) \in \mathcal{R}} \left\lceil \sum_{i=1}^{k} \text{dur}_{l'_i} + \text{dur}_{l'_i, l'_{i+1}} \right\rceil_T \geq \left\lceil \sum_{r \in \mathcal{R}} \sum_{l' \in r} (\text{dur}_{l'} + L^{\text{turn}}) \right\rceil_T \\
&\underbrace{=}_{(4)} \left\lceil \sum_{r \in \mathcal{R}} \sum_{l' \in r} \left((|l| - 1) L^{\text{wait}} + L^{\text{turn}} + \sum_{e \in l'} L_e^{\text{drive}} \right) \right\rceil_T \\
&= \left\lceil \sum_{l' \in \mathcal{L}} \left(L^{\text{turn}} - L^{\text{wait}} + \sum_{e \in l'} (L_e^{\text{drive}} + L^{\text{wait}}) \right) \right\rceil_T \\
&= \left\lceil \sum_{l \in [L]} \left(2z_l (L^{\text{turn}} - L^{\text{wait}}) + \sum_{e \in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l} \right) \right\rceil_T \geq \text{dur}
\end{aligned}$$

87

Hence, by finally setting

$$\text{dur} = \left\lceil \frac{\sum_{l\in[L]} \left(2z_l(L^{\text{turn}} - L^{\text{wait}}) + \sum_{e\in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l}\right)}{T} \right\rceil$$

we conclude that from any feasible solution $(\mathcal{L}, \mathcal{R})$ to (cost-opt) we can construct some feasible solution to Model 2 such that

$$g(\mathcal{L}, \mathcal{R}) \geq c_{\text{time}}\text{dur} + c_{\text{length}} \sum_{l\in[L]} \sum_{e\in E} 2x_{e,l}\text{length}_e,$$

which means that the objective function value of Model 2 is a lower bound to (cost-opt).
On the other hand every feasible solution to Model 2 is a feasible solution to Model 1. This can be seen by setting the three types of variables, $f_e^{\min}$, $f_{e,u}$ and dur, that are contained in both models, to be the same. Hence constraints (12) - (14) are satisfied, and also (11) is satisfied since

$$\begin{aligned}\text{dur} \cdot T &\geq \sum_{l\in[L]} \left(2z_l \underbrace{(L^{\text{turn}} - L^{\text{wait}})}_{\geq 0} + \sum_{e\in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l} \right) \\ &\geq \sum_{e\in E} 2f_e^{\min}(L_e^{\text{drive}} + L^{\text{wait}}).\end{aligned}$$

For the objective functions it additionally holds that

$$\sum_{l\in[L]} \sum_{e\in E} 2x_{e,l}\text{length}_e = \sum_{e\in E} 2f_e^{\min}\text{length}_e.$$

This means that every solution to Model 2 can be projected to a solution of Model 1 with smaller objective value in Model 1, meaning that Model 2 is an upper bound to Model 1. □

We can again construct a feasible solution for (cost-opt) from the solution of Model 2 in the case that we are only interested in line-pure vehicle schedules. In such schedules, every vehicle serves the same line, alternating between its forward and its backward direction. More formally:

Definition 14. A solution to (cost-opt) is called line-pure if $\mathcal{R} = \{r_l : l \in \mathcal{L}\}$, with $r_l = (l^+, l^-)$ being the route that contains only the forward and backward direction of line $l \in \mathcal{L}$.

Again, we do not only want to find a lower, but also an upper bound to (cost-opt). To this end we slightly modify Model 2. Instead of measuring the overall duration of all lines in constraint (18), we track each line individually by using the constraints

$$2z_l(L^{\text{turn}} - L^{\text{wait}}) + \sum_{e\in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l} \leq d_l \cdot T \quad \forall l \in [L] \tag{26}$$

$$\sum_{l\in[L]} d_l = \text{dur} \tag{27}$$

$$d_l \in \mathbb{N}. \tag{28}$$

By doing so, we implicitly evaluate our lines using a line-pure vehicle schedule.

88

Definition 15. Consider Model 2 and replace constraint (18) by constraints (26)-(28). We call this modified version Model 2* and its optimal objective value $z^{\text{opt}}_{2^*}$.

Restricting ourselves to the special structure of line-pure vehicle schedules, we are still able to obtain the optimal solution to (cost-opt) by simply considering loads and lines. This is the main result of this section.

Theorem 16. *An optimal solution to Model 2* solves (cost-opt) under the restriction that only line-pure vehicle schedules are allowed.*

Proof. Let $\mathcal{L}, \mathcal{R}$ be some line-pure feasible solution to (cost-opt). For the objective value of $(\mathcal{L}, \mathcal{R})$ we know that

$$\text{length}(\mathcal{L},\mathcal{R}) = \sum_{r=(l'_1,\dots,l'_{k_r})\in\mathcal{R}} \sum_{i=1}^{k_r} \text{length}_{l'_i} + \underbrace{\text{length}_{l'_i,l'_{i+1}}}_{=0} = \sum_{l\in\mathcal{L}} 2\text{length}_l = \sum_{l\in\mathcal{L}}\sum_{e\in l} 2\text{length}_e,$$

and that

$$\begin{aligned}\text{dur}(\mathcal{L},\mathcal{R}) &= \sum_{r\in R}\left\lceil \sum_{l'\in r}(\text{dur}_{l'} + L^{\text{turn}})\right\rceil_T = \sum_{l\in\mathcal{L}}\left\lceil 2(\text{dur}_l + L^{\text{turn}})\right\rceil_T \\ &= \sum_{l\in\mathcal{L}}\left\lceil 2(L^{\text{turn}} - L^{\text{wait}}) + \sum_{e\in E: e\in l} 2(L^{\text{drive}}_e + L^{\text{wait}})\right\rceil_T.\end{aligned}$$

We can extend the line plan $\mathcal{L}$ to some feasible solution to Model 2* by again defining a bijective mapping $\varphi : \mathcal{L} \to [|\mathcal{L}|]$ such that $x_{e,\varphi(l)} = 1$ iff $e \in l$ for $l \in \mathcal{L}$ for all $e \in E$. Analogously a solution $x_{e,l}$ can be transformed into some feasible line plan $\mathcal{L}$ by defining a line l to contain exactly all edges $e \in E$ if $x_{e,l} = 1$. Thus there exists a bijection between the set of feasible solutions between (cost-opt) and Model 2* as well as the same objective function for both problems since

$$\sum_{l\in\mathcal{L}}\sum_{e\in l} 2\text{length}_e = \sum_{l\in\mathcal{L}}\sum_{e\in E} 2x_{e,\varphi(l)}\text{length}_e = \sum_{l\in[L]}\sum_{e\in E} 2x_{e,l}\text{length}_e$$

and

$$\begin{aligned}&\sum_{l\in\mathcal{L}}\left\lceil 2(L_{\text{turn}} - L_{\text{wait}}) + \sum_{e\in E: e\in l} 2(L^{\text{drive}}_e + L^{\text{wait}})\right\rceil_T \\ &= \sum_{l\in[L]}\left\lceil 2z_l(L^{\text{turn}} - L^{\text{wait}}) + \sum_{e\in E} 2x_{e,l}\text{length}_e(L^{\text{drive}}_e + L^{\text{wait}})\right\rceil_T = \sum_{l\in[L]} d_l.\end{aligned}$$

Hence their optimal objective values coincide. □

For the general case of (cost-opt), i.e., without the restriction of line-pure vehicle schedules, Model 2* still finds a feasible solution and therefore provides an upper bound to (cost-opt).

Corollary 17. *The optimal objective value to Model 2* imposes an upper bound on the optimal objective value of (cost-opt), i.e.,*

$$z^{\text{opt}} \le z^{\text{opt}}_{2^*}.$$

Additionally, for an L, that is known to be sufficiently large, we can provide an a priori bound between z_2^{opt} and $z_{2^*}^{\text{opt}}$.

Theorem 18. *The gap between the optimal objective values of Model 2 and Model 2* is bounded by $c_{\text{time}} \cdot (L-1)$, i.e.,*

$$z_{2^*}^{\text{opt}} - z_2^{\text{opt}} \leq c_{\text{time}} \cdot (L-1).$$

Proof. Let $(\text{dur}, f^{\min}, f, x, z)$ be an optimal solution to Model 2. Since the solution is optimal, we know that

$$\text{dur} = \left\lceil \frac{\sum_{l\in[L]} \left(2z_l(L^{\text{turn}} - L^{\text{wait}}) + \sum_{e\in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l}\right)}{T} \right\rceil.$$

Denote

$$a_l := \frac{2z_l(L^{\text{turn}} - L^{\text{wait}}) + \sum_{e\in E} 2(L_e^{\text{drive}} + L^{\text{wait}}) \cdot x_{e,l}}{T},$$

i.e.,

$$\text{dur} = \left\lceil \sum_{l\in[L]} a_l \right\rceil.$$

Since the only difference between Model 2 and Model 2* is the replacement of constraint (18) by constraints (26) and (27), $(\text{dur}^*, f^{\min}, f, x, z)$ with

$$d_l = \lceil a_l \rceil$$
$$\text{dur}^* = \sum_{l\in[L]} d_l$$

is a feasible solution for Model 2*. Therefore

$$z_{2^*}^{\text{opt}} \leq c_{\text{time}} \cdot \text{dur}^* + c_{\text{length}} \cdot \sum_{l\in[L]} \sum_{e\in E} 2x_{e,l}\text{length}_e$$

holds. Let

$$\bar{a_l} = a_l - \lfloor a_l \rfloor.$$

be the non-integer part of a_l. Without loss of generality there exists an $l \in \{1, \ldots, L\}$ with $\bar{a_l} > 0$, because otherwise the gap would be 0. Then it holds that

$$\begin{aligned}
z_{2^*}^{\text{opt}} - z_2^{\text{opt}} &\leq c_{\text{time}} \cdot (\text{dur}^* - \text{dur}) \\
&= c_{\text{time}} \cdot \left(\sum_{l\in[L]} \lceil a_l \rceil - \left\lceil \sum_{l\in[L]} a_l \right\rceil \right) \\
&= c_{\text{time}} \cdot \Big(\underbrace{\sum_{l\in[L]} \lceil \bar{a_l} \rceil}_{\leq L} - \underbrace{\left\lceil \sum_{l\in[L]} \bar{a_l} \right\rceil}_{\geq 1} \Big) \\
&\leq c_{\text{time}} \cdot (L-1)
\end{aligned}$$

□

Using this gap, Model 2 can provide an a priori bound on the objective value of (cost-opt).

Corollary 19. *The absolute error of solving Model 2 or Model 2* is at most* $c_{\text{time}} \cdot (L-1)$*, i.e.,*

$$z_{2^*}^{\text{opt}} - z^{\text{opt}} \leq c_{\text{time}} \cdot (L-1)$$
$$z^{\text{opt}} - z_2^{\text{opt}} \leq c_{\text{time}} \cdot (L-1).$$

The following example shows that the bound provided in Corollary 19 can be attained.

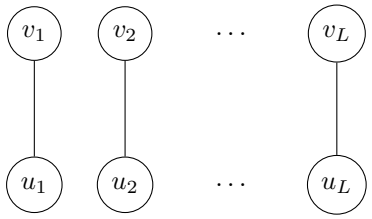


Figure 4: Infrastructure network for Example 20

Example 20. Let L be known. Consider the PTN depicted in Figure 4 with L single edges, connecting two nodes each. Each edge has a length of $L_e^{\text{drive}} = \epsilon$, one passenger travelling and let $L^{\text{turn}} = \epsilon$. Then

$$z_{2^*}^{\text{opt}} - z^{\text{opt}} = c_{\text{time}} \cdot L - c_{\text{time}} \cdot \left\lceil \frac{4n\epsilon}{T} \right\rceil \overset{\epsilon \to 0}{\to} c_{\text{time}} \cdot (L-1).$$

As we have already mentioned, the presented theoretical results of this section only hold true if L is chosen sufficiently large. One possible upper bound for L is $\sum_{u,v \in OD} \lceil \frac{W_{uv}}{\text{Cap}} \rceil$, giving every od pair the possibility to build its own lines. In practice, however, much smaller values for L are already feasible. Smaller values of L, that are still large enough, can be computed with the following insight.

Theorem 21. *Let* ψ *be a feasible solution to Model 2* with objective value* obj *with an arbitrary* L*. Define*

$$L^{\text{ub}} := \frac{\text{obj} - c_{\text{length}} \cdot \frac{\sum_{u,v \in V} \text{SP}(u,v) \cdot W_{uv}}{\text{Cap}}}{c_{\text{time}}}, \tag{29}$$

where $\text{SP}(u,v)$ *for* $u, v \in V$ *is the shortest path from* u *to* v *with respect to the edge lengths* length_e*,* $e \in E$*. Then the number of lines of the optimal solution to Model 2* is bounded by* L^{ub}*.*

Proof. Assume ψ' to be an optimal solution to Model 2* with objective value

$z_{2^*}^{\text{opt}}$ that uses $L' > L^{\text{ub}}$ lines. Then

$$
\begin{aligned}
z_{2^*}^{\text{opt}} &= c_{\text{time}} \underbrace{\text{dur}}_{\geq L' > L^{\text{ub}}} + c_{\text{length}} \sum_{l \in |L|} \sum_{e \in E} 2 \cdot x_{el} \cdot \text{length}_e \\
&\underbrace{>}_{(27)} c_{\text{time}} \cdot L^{\text{ub}} + c_{\text{length}} \sum_{l \in |L|} \sum_{e \in E} 2 \cdot x_{el} \cdot \text{length}_e \\
&\underbrace{=}_{(29)} \text{obj} + c_{\text{length}} \left(\sum_{l \in |L|} \sum_{e \in E} 2 \cdot x_{el} \cdot \text{length}_e - \frac{\sum_{u,v \in V} \text{SP}(u,v) \cdot W_{uv}}{\text{Cap}} \right) \\
&\underbrace{\geq}_{(19)} \text{obj} + c_{\text{length}} \left(\sum_{l \in |L|} \sum_{e \in E} 2 \cdot {f_e}^{\min} \cdot \text{length}_e - \frac{\sum_{u,v \in V} \text{SP}(u,v) \cdot W_{uv}}{\text{Cap}} \right) \\
&\underbrace{\geq}_{(*)} \text{obj}
\end{aligned}
$$

which is a contradiction to ψ' being optimal. Here, $(*)$ holds due to $\sum_{l \in |L|} \sum_{e \in E} 2 \cdot {f_e}^{\min} \cdot \text{length}_e$ being the (passenger-weighted) length of a feasible flow and $\frac{\sum_{u,v \in V} \text{SP}(u,v) \cdot W_{uv}}{\text{Cap}}$ being the length of the corresponding shortest flow. □

Using Theorem 21 we can now obtain a sufficiently large, but still reasonably low, choice of L by solving Model 2* only twice: For obtaining a first solution an arbitrarily chosen L is sufficient. With the objective value of this first solution we then can calculate L^{ub} by using (29). Now, if we solve Model 2* again with L^{ub}, we can be sure that an optimal solution will be found.

Continuing our process of finding public transport plans of good quality, we investigate how Model 2* behaves when confronted with Example 12. It illustrates that the solutions of Model 2* are more usable than the solutions of Model 1*, i.e., the practical problems demonstrated at the end of Section 4 are solved by Model 2*.

Example 22. We continue Example 12 and now consider the solution constructed in Theorem 13. These now provide simple lines, resulting in the line-pure vehicle schedule depicted in Figure 5, improving on the line structure of Example 12. The first line is depicted in red, the second is dashed in green. The lines here look much more reasonable for practical implementation than the solution which was obtained by Model 1*.

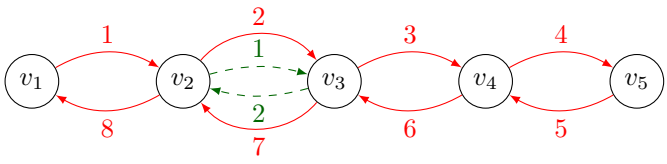


Figure 5: Solution of Model 2

6 Model 3: Integrating Timetabling and Vehicle Scheduling

In Model 1 and Model 2 we did not consider all arising subproblems of (cost-opt) so far. Especially, we did not include a proper vehicle scheduling into the mathematical models. With the following model we want to overcome this issue and formulate the whole problem in an integrated way.

To formulate the integrated model, we need a notation for the event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ (see, e.g., [Lie06, LM04, Nac98, Pee03, PK01]). The set of events $\mathcal{E}$ consists of all departures and all arrivals of all lines at all stops and two additional OD-events $((u, \text{dep}), (u, \text{arr}))$ per stop u for passengers to enter and leave the network, denoted as $\mathcal{E}_{OD}$. The set $\mathcal{A}$ connects the events by driving, waiting and transfer activities. The OD-events are connected to each departure event of the corresponding stop using OD-activities ($\mathcal{A}_{OD}$). Using this, we can now formulate the integrated model. Let further denote with $\mathcal{A}_{l'}$ all activities in $\mathcal{A} \setminus \mathcal{A}_{OD}$ that are included in a directed line $l' \in \mathcal{L}'$.

Model 3. Given the input data from Notation 1, find a feasible public transport plan $(\mathcal{L}, \mathcal{R})$ with minimal costs, i.e., minimizing $g(\mathcal{L}, \mathcal{R})$.

$$\min \sum_{r\in[R]} \text{cost}_r$$

$$\text{s.t.} \quad T\text{dur}_r \geq \cdot \sum_{l'\in\mathcal{L}'} x_{l',r} \cdot \text{dur}_l + \sum_{l'_1,l'_2\in\mathcal{L}'} x_{(l'_1,l'_2),r} \cdot \text{dur}_{l'_1,l'_2} \quad \forall r \in [R] \tag{30}$$

$$\text{length}_r \geq \sum_{l'\in\mathcal{L}'} x_{l',r} \cdot \text{length}_l + \sum_{l'_1,l'_2\in\mathcal{L}'} x_{(l'_1,l'_2),r} \cdot \text{length}_{l'_1,l'_2} \quad \forall r \in [R] \tag{31}$$

$$\text{cost}_r \geq c_{\text{length}} \cdot \text{length}_r + c_{\text{time}} \cdot \text{dur}_r \quad \forall r \in [R] \tag{32}$$

$$\sum_{l^*\in\mathcal{L}'} x_{(l',l^*),r} = x_{l',r} = \sum_{l^*\in\mathcal{L}'} x_{(l^*,l'),r} \quad \forall l' \in \mathcal{L}', \forall r \in [R] \tag{33}$$

$$\sum_{r\in[R]} x_{l',r} = \sum_{r\in[R]} x_{b(l'),r} \quad \forall l' \in \mathcal{L}' \tag{34}$$

$$\text{Cap} \cdot \sum_{r\in[R]} x_{l',r} \geq \sum_{u,v\in V} f_{a,(u,v)} \quad \forall l' \in \mathcal{L}', \forall a \in \mathcal{A}_{l'} \tag{35}$$

$$\sum_{\substack{i\in\mathcal{E}:\\(i,j)=a\in\mathcal{A}}} f_{a,(u,v)} = \sum_{\substack{i\in\mathcal{E}:\\(j,i)\in\mathcal{A}}} f_{a,(u,v)} \quad \forall u,v \in V, \forall j \in \mathcal{E} \setminus \mathcal{E}_{OD} \tag{36}$$

$$\sum_{\substack{i\in\mathcal{E}:\\(i,j)=a\in\mathcal{A}_{\text{OD}}}} f_{a,(u,v)} = W_{uv} \quad \forall u,v \in V, \forall j = (v,\text{arr}) \in \mathcal{E}_{OD} \tag{37}$$

$$\sum_{\substack{i\in\mathcal{E}:\\(j,i)=a\in\mathcal{A}_{\text{OD}}}} f_{a,(u,v)} = W_{uv} \quad \forall u,v \in V, \forall j = (u,\text{dep}) \in \mathcal{E}_{OD} \tag{38}$$

$$\sum_{l'\in\mathcal{L}'} v_{(d,l'),r} \leq \sum_{l'\in\mathcal{L}'} x_{l',r} \quad \forall r \in [R] \tag{39}$$

$$v_{(l'_1,l'_2),r} \leq x_{(l'_1,l'_2),r} \cdot |\mathcal{L}'| \quad \forall l'_1,l'_2 \in \mathcal{L}', r \in [R] \tag{40}$$

$$\sum_{l'\in\mathcal{L}'} x_{(d,l'),r} = 1 \tag{41}$$

$$\sum_{l'_1\in\mathcal{L}'} v_{(l'_1,l'),r} - \sum_{l'_1\in\mathcal{L}'} v_{(l',l'_1),r} = x_{l',r} \quad \forall l' \in \mathcal{L}', r \in [R] \tag{42}$$

$$\text{dur}_r \in \mathbb{N}, \text{length}_r, \text{cost}_r \in \mathbb{R} \quad \forall r \in [R]$$

$$x_{l',r} \in \{0,1\} \quad \forall r \in [R], l' \in \mathcal{L}'$$

$$x_{(l'_1,l'_2),r} \in \{0,1\}, v_{(l'_1,l'_2),r} \in \mathbb{N} \quad \forall r \in [R], l'_1,l'_2 \in \mathcal{L}'$$

$$f_{a,(u,v)} \in \mathbb{N} \quad \forall a \in \mathcal{A}, (u,v) \in E, u,v \in V$$

Coefficients:

- R: number of possible vehicle routes, we assume it to be sufficiently large
- $\mathcal{L}'$: the set of all possible directed lines in the network, $b(l')$ denotes the backwards direction for a directed line l', l is the corresponding undirected line.

Variables:

94

- $x_{l',r}$ – is 1 iff the directed line l' is part of route r
- $x_{(l'_1,l'_2),r}$: is 1 iff lines l'_1 and l'_2 are served directly after each other in route r
- cost_r – the costs of route r
- dur_r – the duration of route r
- length_r – the length of route r
- $f_{a,(u,v)}$ – the number of passenger traveling from u to v using activity a

This model finds a cost-optimal public transport plan (i.e., line plan, timetable and vehicle schedules). The f variables determine the passenger flow, satisfying the classical flow conservation constraints ((36)-(38)) and creating coupling constraints for the vehicle routes r in (35), determined by the x-variables. The duration and length of the routes are determined in (30) and (31) and then combined in (32) to determine the costs. Of course, the vehicle routes need to satisfy flow conservation as well (see (33)). (**??**) are the subtour elimination constraints. Constraint (34) ensures that every line is served in both directions. Since this is a rather large program, we prove formally that it is working as intended.

Theorem 23. *Model 3 is a correct formulation for (cost-opt).*

Proof. We prove the theorem in the following three steps:

1. For every optimal solution for Model 3 there is a feasible solution for (cost-opt)
2. For every feasible solution for (cost-opt) there is a feasible solution for Model 3
3. The objective values coincide for optimal solutions

Step 1: Let $(x, f, \text{cost}, \text{dur}, \text{length})$ be an optimal solution for Model 3. We construct a feasible solution to (cost-opt), i.e., a feasible public transport plan $(\mathcal{L}, \mathcal{R})$. For the line concept, set

$$f_l = \sum_{r\in[R]} x_{l,r}$$

and let $l \in \mathcal{L}$ if $f_l > 0$. Due to (34), this is well defined and only both or no direction of a line will be served. The vehicle routes for the vehicle schedule can easily be constructed using the x variables.
In order to check feasibility of the line concept, we transform the passenger weights f in the EAN to weights w_p in the PTN for each passenger. Then for every $e \in E$ it holds that

$$\begin{aligned}\sum_{\substack{p\in P_{\text{all}}:\\ e\in p}} w_p &= \sum_{l'\in\mathcal{L}'} \sum_{\substack{a\in\mathcal{A}_{l'}:\\ a \text{ corr. to } e}} \sum_{u,v\in V} f_{a,(u,v)} \\ &\overset{(35)}{\leq} \text{Cap} \sum_{l'\in\mathcal{L}'} \sum_{\substack{a\in\mathcal{A}_{l'}:\\ a \text{ corr. to } e}} \underbrace{\sum_{r\in[R]} x_{l,r}}_{=f_l} \\ &= \text{Cap}|\{l \in \mathcal{L} : e \in l\}|\end{aligned}$$

Therefore constraint (1) is satisfied and the constructed line concept is feasible. Regarding the feasibility of the vehicle schedule, the subtour elimination constraints (**??**) ensure that all lines in a route are distinct and every line is covered exactly once due to the construction of $\mathcal{L}$ and the optimality of the solution.
Step 2: Let now $(\mathcal{L}, \mathcal{R})$ be a feasible public transport plan with corresponding passenger paths P_{all}. Then there exist passenger flows $f_{u,v}$ in the EAN for all OD-pairs such that

$$\sum_{u,v \in V} f_{a,(u,v)} \leq \text{Cap} \qquad a \in \mathcal{A}_{l'},\, l' \in \mathcal{L}', \tag{43}$$

since (1) is satisfied and passengers can choose an arbitrary line for each edge in their path. Set x variables according to $\mathcal{R}$, i.e., set $x_{l',r} = 1$ iff line $l' \in \mathcal{L}'$ is covered in $r \in \mathcal{R}$ and $x_{(l'_1,l'_2),r} = 1$ iff line l'_2 is directly behind line l'_1 in $r \in \mathcal{R}$. Then the constructed solution is feasible for Model 3, since (36)-(38) are satisfied due to the construction of f and P_{all}, (**??**) holds since the given vehicle routes are feasible, (35) holds due to the construction of x and (43), (34) holds due to the construction of x and $\mathcal{L}'$, (33) holds due to the construction of x and the feasibility of $\mathcal{R}$. The remaining constraints (30)-(32) are no feasibility constraints.
Step 3: The objective value of the solutions does not change when using above constructions. Note, that the $\lceil \cdot \rceil_T$-operator is replaced by multiplication with $\frac{1}{T}$ and the integer constraint of dur_r.
Together, these three steps prove the correctness of the proposed Model 3. □

With this, the following relations between the Models 1, 2 and 3 can be formally stated.

Corollary 24. *Model 1 and Model 2 are relaxations of Model 3.*

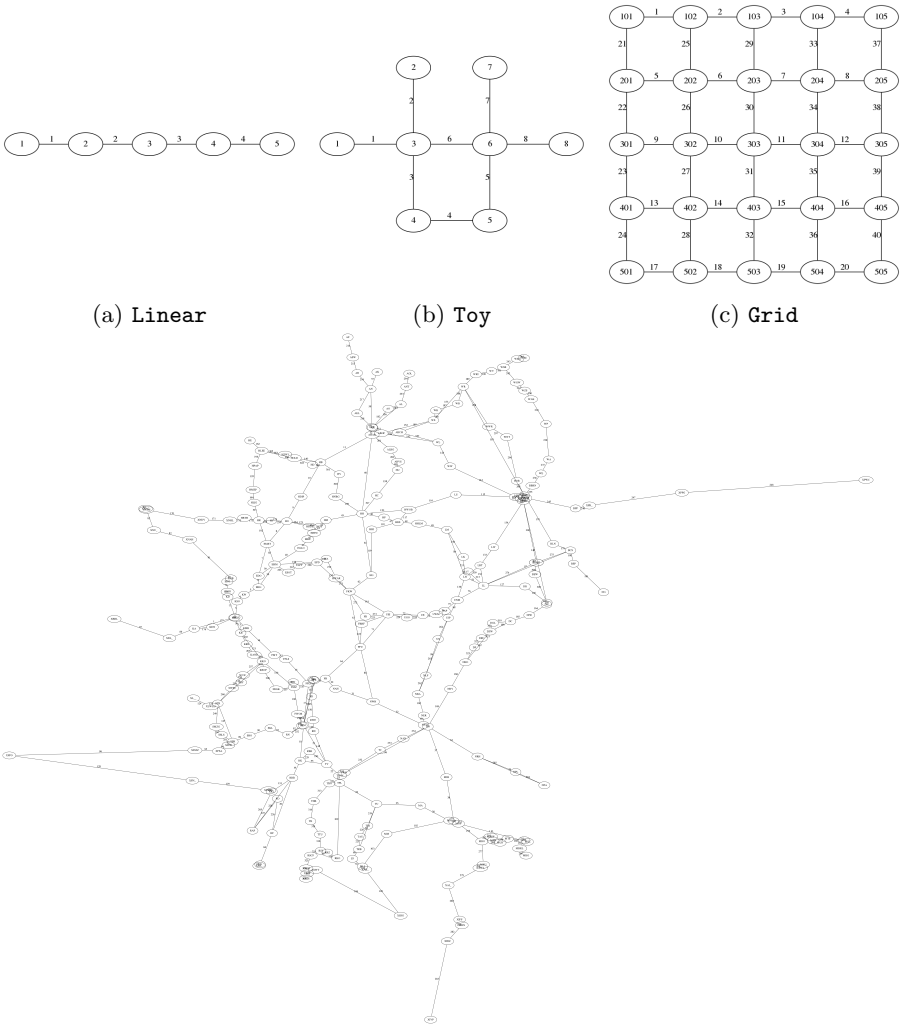
Proof. Directly follows from the proof of Theorems 4, 13 and 23. □

Model 3 is too large to be solved for realistic instances. As can be seen in the computational experiments in Section 7, the integrated problem cannot be solved – even for instances of small size. This is due to its enormous number of variables including a trip for every possible line in the network. Nevertheless, Model 3 can be used if enough variables are fixed. We hence can combine it with Model 2 by fixing the lines in Model 3 to the optimal lines computed by Model 2. This means that we only need to consider the constraints (30)-(33) and (**??**), additionally guaranteeing that every trip in $\mathcal{L}'$ is covered exactly once. The result is a tractable model for medium-sized instances.
Other possibilities to reduce the size of Model 3 would be to start with a line pool of limited size (e.g., as generated in [GHS17] or from Model 2) or to use column generation approaches as in [BGP07].

7 Experiments

In the computational experiments we implemented the three proposed models with the open source library LinTim (see [APS+, GSS13, SAP+18]) and tested them on four different datasets. These datasets are described in Table 1 and depicted in Figure 6.

96



(a) `Linear` (b) `Toy` (c) `Grid`

(d) `Germany`

Figure 6: Networks of the datasets used in the experiments

Instance	Nodes	Edges	Passengers
`Linear`	5	4	141
`Toy`	8	8	2622
`Grid`	25	40	2546
`Germany`	250	326	385868

Table 1: Dataset properties

We implemented Model 1, Model 1*, Model 2, Model 2* and Model 3 using Gurobi 8.0 as a MIP solver with default settings. We tested all implementations on a compute server (6 cores of Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, 78

97

GB RAM) with a time limit of 3 hours per test case. For each model and each instance we considered two different cases: Either $L^{\text{turn}} = L^{\text{wait}}$ or $L^{\text{turn}} > L^{\text{wait}}$ to distinguish the cases where Model 1* is able to find an optimal solution and where it is not. We obtained the results depicted in Tables 2 and 3. A symbol $^\circ$ denotes that the problem has not been solved to optimality and hence only the best found upper or lower bound is presented.

Instance	Model 1		Model 2		Model 3	
	Model 1	Model 1*	Model 2	Model 2*	lb	ub
`Linear`	80	80	80	130	80	80
`Toy`	1424	1424	1424	1696	1270°	1460°
`Grid`	1034	1034	1034	1034	–	–
`Germany`	73321°	84694°	54148°	–	–	–

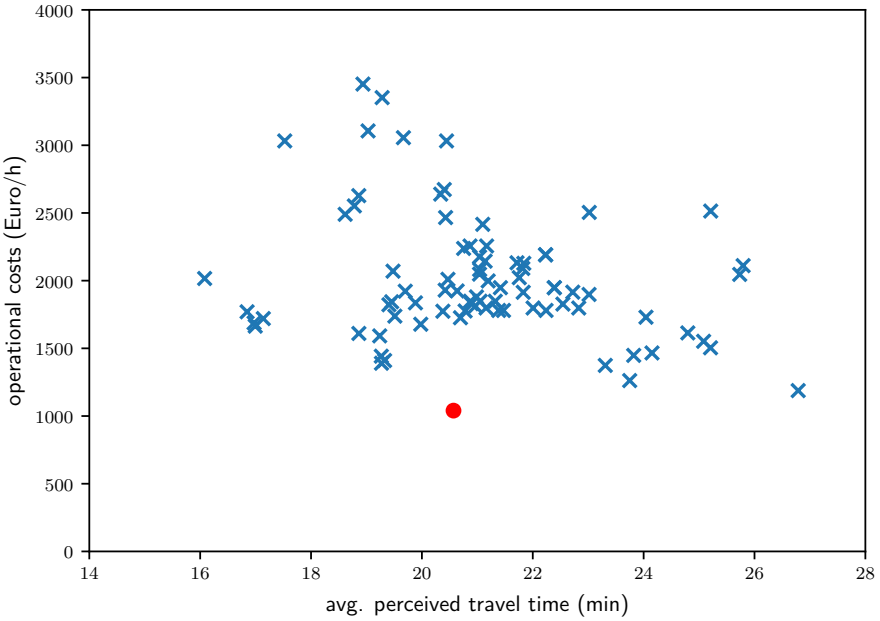
Table 2: Objective values for the case of $L^{\text{turn}} = L^{\text{wait}}$

Instance	Model 1		Model 2		Model 3	
	Model 1	Model 1*	Model 2	Model 2*	lb	ub
`Linear`	80	130	130	130	130	130
`Toy`	1424	1474	1424	1696	1288°	1539°
`Grid`	1034	1134	1030°	1140	–	–
`Germany`	74462°	85612°	54148°	–	–	–

Table 3: Objective values for the case of $L^{\text{turn}} > L^{\text{wait}}$

For each of the three models there exist two columns. The left column contains a lower bound to (cost-opt), whereas the right column contains an upper bound, i.e., the objective value of the best found feasible solution.
We observe for Model 1 that in the case $L^{\text{turn}} = L^{\text{wait}}$ it almost always finds the optimal objective value within the specified time limit of 3 hours. Only in our biggest instance we cannot get an optimal solution within the time limit (we still have a gap of 13.7% here). For the case $L^{\text{turn}} > L^{\text{wait}}$ there exists a gap between the lower and the upper bound of Model 1, but this model still obtains the best solutions.
Model 2 can solve the two smallest instances easily, but starts having trouble with the time limit for `Grid`. For `Germany` it is not able to find a feasible solution within the specified time limit. Regarding the solution quality, we see that the lower bound given by Model 2 is only in a single case sharper than the lower bound given by Model 1. On the other hand, the upper bounds found by Model 2* never have smaller objective values than Model 1*. Note, that the values for L provided by Theorem 21 are close to the number of used lines in the optimal solutions found by Model 2*, e.g., for dataset `Grid`, L^{ub} is 15 and 13 lines are used in the computed optimal solution.
Model 3 is already on the toy instance not able to find an optimal solution within 3 hours. The obtained objective values for `Linear` and the bounds for

98



`Toy` are consistent with the values given in Models 1 and 2. For the bigger instance, even the precomputation of the complete line pool for Model 3 was not possible anymore.

We illustrate our results on the dataset `Grid` (see [FHSS17, FOR]) and compare them to previously knwon solutions on this dataset. All solutions are evaluated with respect to their costs and their traveling times. The solutions shown in Figure 7 have been computed sequentially, contrary to the integrated approach presented in this work. We see that the sequential solutions with smallest costs are A4 (computed in [PSSS17]) and P5 (computed in [Lie18].) For this instance of the dataset `Grid` it holds that $L^{\text{turn}} = L^{\text{wait}}$. Hence, we were able to compute a cost-minimal solution by using Model 1. Its objective value is depicted as a red line, since the traveling times are not computed for this model. The optimal solution improves the costs by 23% compared to the best existing solution.

The traveling time of the cost-minimal solution is hard to evaluate: Assigning passengers to travel on their shortest paths in the EAN, as done for the other solutions in Figure 7, would lead to a traveling time of only 20.57. We did not depict this objective value in the figure since in this solution the passengers are far away from using the paths computed for them in Model 1 and hence the solution would have heavily overloaded vehicles. On the other hand, using a capacitated evaluation, i.e., finding a system optimal solution for the passengers, where no overcrowding in the vehicles occur, will lead to a perceived travel time of 23.86. But since this evaluation is not consistent with the evalaution strategy used for the other solutions depicted in Figure 7, we chose to only depict the cost value in the figure.

We finally investigate the influence of valid inequalities introduced in Lemma 5 on the runtime of Model 1. We restricted this investigation to `Grid`, since the runtime for the smallest two instances is already less than a second, and for

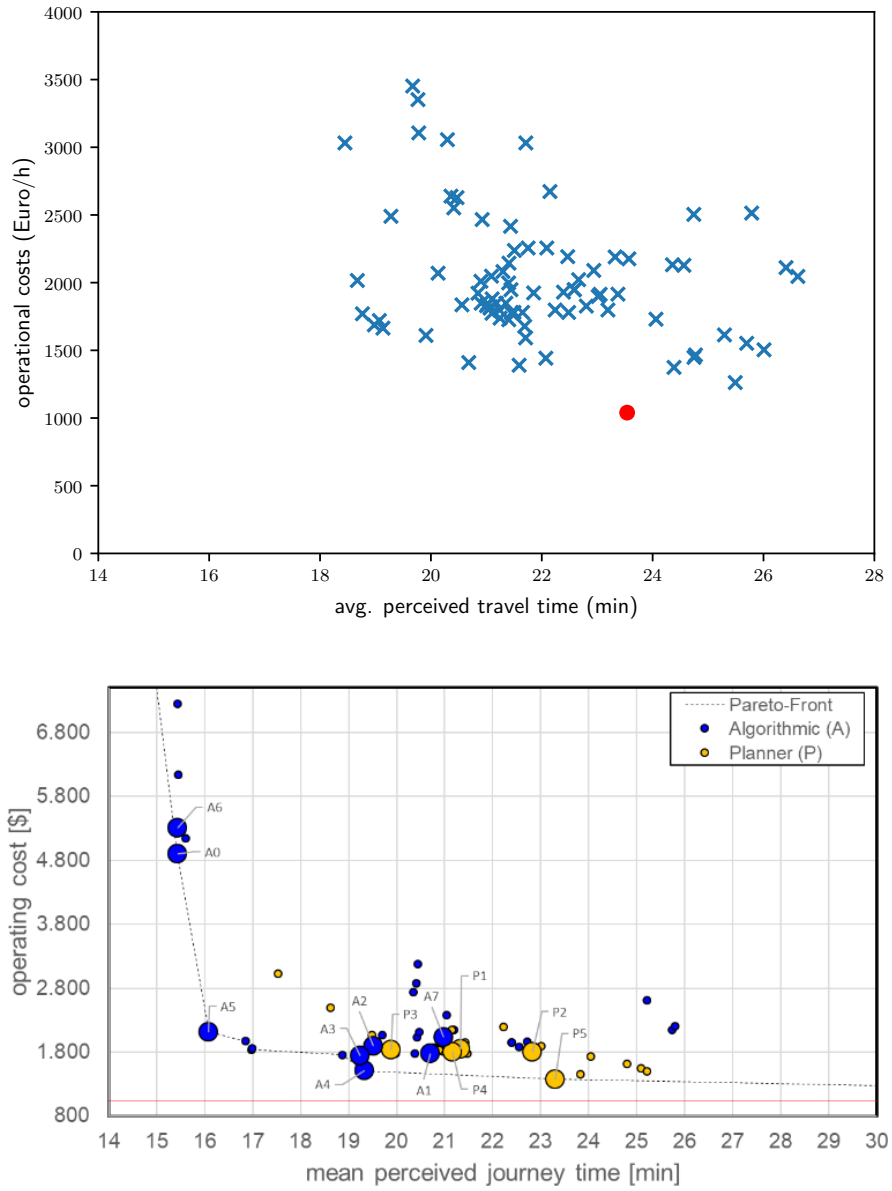


Figure 7: Multiple solutions for `Grid` (see [FOR]), evaluated by their cost per hour and traveling time (perceived journey time meaning traveling time plus a time penalty for every occurring transfer). With our models we were able to find a cost-minimal solution. Its objective value is depicted by a red line.

`Germany` it is already non-trivial to determine "good" cuts of the network. For `Grid`, however, we took all horizontal and vertical cuts of the network, whose PTN is depicted in Figure 6, into the model. With this improvement we were able to speed up the solution process significantly with respect to runtime and number of explored MIP nodes, as can be seen in Table 4.

100

Parameters	No Cuts		Cuts	
	Model 1	Model 1*	Model 1	Model 1*
Nodes explored	46557	26391	2398	3845
Runtime in sec	23.18	12.6	10.61	8.99

Table 4: Runtime improvements with Lemma 5 on `Grid` for $L^{\text{turn}} > L^{\text{wait}}$

8 Outlook

In this work we propose three models to compute cost-optimal public transport plans. For an overview, see Table 5. For the first two models we derived optimality conditions and bounds to the optimal solution. With the third model we present an IP formulation for the integrated exact model. The computational experiments show that the implementation of the models is computationally tractable.

Model	Advantages	Disadvantages
Model 1	Very low computation time, able to provide solutions for real-world instances Find optimal solution under (weak assumptions)	Low theoretical bound quality
Model 2	Low computation time Finds optimal line-pure solution Better bound quality than Model 1	May not find optimal solution for non line-pure vehicle schedules Dependent on choice of L
Model 3	Integrated model, finding the optimal solution to the problem	High computation time, only able to provide solutions for small instances

Table 5: Overview of the different models presented in this paper

Model 1 is able to compute cost-optimal solutions up to `Grid` outperforming previous approaches to tackle this problem. For large networks the model provides bounds of good quality in a reasonable amount of time. Model 2 finds optimal line-pure public transport plans and constitutes a trade-off between computation time and solution quality. Finally, Model 3 yields a cost-optimal public transport plan without requiring any further assumptions.

For future work we plan to sharpen the formulation of Model 1 by identifying good cuts. It would hopefully be the case that better cuts lead to a further decrease of the computation time, especially for the large instances.
Furthermore it would be interesting to not only find a solution with minimal costs, but to find a *lexicographic* solution, i.e., the cost-optimal solution with the best traveling time for the passengers. To this end, we can include the passengers' traveling time in Model 3 which will most likely further increase the computation time of the model. To use this model effectively, more work

in speed-up techniques is necessary. Promising ideas include column generation and decomposition techniques, similar to the methods presented in [LPSS].

References

[APS+] S. Albert, J. Pätzold, A. Schiewe, P. Schiewe, and A. Schöbel. LinTim - Integrated Optimization in Public Transportation. Homepage. see http://lintim.math.uni-goettingen.de/.

[BBLV17] S. Burggraeve, S.H. Bull, R.M. Lusby, and P. Vansteenwegen. Integrating robust timetabling in line plan optimization for railway systems. *Transportation Research C*, 77:134–160, 2017.

[BGP07] R. Borndörfer, M. Grötschel, and M. Pfetsch. A Column-Generation Approach to Line Planning in Public Transport. *Transportation Science*, 41:123–132, 2007.

[BHK17] R. Borndörfer, H. Hoppmann, and M. Karbstein. Passenger routing for periodic timetable optimization. *Public Transport*, 9(1-2):115–135, 2017.

[BK09] S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317, 2009.

[BKLL18] R. Borndörfer, M. Karbstein, C. Liebchen, and N. Lindner. A simple way to compute the number of vehicles that are required to operate a periodic timetable. In Ralf Borndörfer and Sabine Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, volume 65 of *OpenAccess Series in Informatics (OASIcs)*, pages 16:1–16:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BNP09] R. Borndörfer, M. Neumann, and M. E. Pfetsch. The line connectivity problem. In *Operations Research Proceedings 2008*, pages 557–562. Springer, 2009.

[Bus98] M.R. Bussieck. *Optimal lines in public transport*. PhD thesis, Technische Universität Braunschweig, 1998.

[CvDZ98] M.T. Claessens, N.M. van Dijk, and P.J. Zwaneveld. Cost optimal allocation of rail passenger lines. *European Journal on Operational Research*, 110:474–489, 1998.

[CW86] A. Ceder and N.H.M. Wilson. Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344, 1986.

[DC18] M. Darvish and L. Coelho. Sequential versus integrated optimization: Production, location, inventory control, and distribution. *European Journal of Operational Research*, 268(1):203–214, 2018.

[DH07] G. Desaulniers and M.D. Hickman. Public transit. *Handbooks in operations research and management science*, 14:69–127, 2007.

[DRB+17] S. Dutta, N. Rangaraj, M. Belur, S. Dangayach, and K. Singh. Construction of periodic timetables on a suburban rail network-case study from Mumbai. In *RailLille 2017—7th International Conference on Railway Operations Modelling and Analysis*, 2017.

[FHSS17] M. Friedrich, M. Hartl, A. Schiewe, and A. Schöbel. Angebotsplanung im öffentlichen Verkehr - planerische und algorithmische Lösungen. In *Heureka'17*, 2017.

[Fle91] H. Fleischner. X. 1 algorithms for eulerian trails. eulerian graphs and related topics: Part 1. *Annals of Discrete Mathematics*, 50:1–13, 1991.

[FOR] DFG research unit FOR 2083. Public Tranport Networks. https://github.com/FOR2083/PublicTransportNetworks.

[GGNS16] P. Gattermann, P. Großmann, K. Nachtigall, and A. Schöbel. Integrating Passengers' Routes in Periodic Timetabling: A SAT approach. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany, 2016.

[GHS17] P. Gattermann, J. Harbering, and A. Schöbel. Line pool generation. *Public Transport*, 9(1):7–32, 2017.

[GS13] M. Goerigk and A. Schöbel. Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, 40(5):1363–1370, 2013.

[GSS13] M. Goerigk, M. Schachtebeck, and A. Schöbel. Evaluating line concepts using travel times and robustness: Simulations with the lintim toolbox. *Public Transport*, 5(3), 2013.

[GvHK06] J. Goossens, C.P.M. van Hoesel, and L.G. Kroon. On solving multi-type railway line planning problems. *European Journal of Operational Research*, 168(2):403–424, 2006.

[KDC18] M. Kidd, M. Darvish, and L. Coelho. On the value of integration in supply chain planning. In *29th European Conference on Operational Research (EURO 2018)*, 2018.

[Lie06] C. Liebchen. *Periodic Timetable Optimization in Public Transport*. dissertation.de – Verlag im Internet, Berlin, 2006.

[Lie18] C. Liebchen. Nutzung graphentheoretischer Konzepte zur manuellen Erstellung effizienter Verkehrsangebote. In J. Schönberger and S. Nerlich, editors, *26. Verkehrswissenschaftliche Tage Dresden, Germany: Technische Universität Dresden*, pages 309–332, 2018.

[LLER11] R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR spectrum*, 33(4):843–883, 2011.

[LM04] C. Liebchen and R. Möhring. The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables - and Beyond. In *Proceedings of 9th meeting on Computer-Aided Scheduling of Public Transport(CASPT 2004)*. Springer, 2004.

[LPSS] M. Lübbecke, C. Puchert, P. Schiewe, and A. Schöbel. Detecting structures in network models of integrated traffic planning. Presentation at the Clausthal-Göttingen International Workshop on Simulation Science.

[MCZT18] L. Meng, F. Corman, X. Zhou, and T. Tang. Special issue on Integrated optimization models and algorithms in rail planning and control, 2018.

[Nac98] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. PhD thesis, University of Hildesheim, 1998.

[NO08] K. Nachtigall and J. Opitz. Solving periodic timetable optimisation problems by modulo simplex calculations. In *Proc. ATMOS*, 2008.

[Pee03] L. Peeters. *Cyclic Railway Timetabling Optimization*. PhD thesis, ERIM, Rotterdam School of Management, 2003.

[PK01] L. Peeters and L. Kroon. A Cycle Based Optimization Model for the Cyclic Railway Timetabling Problem. In S. Voß and J. Daduna, editors, *Computer-Aided Transit Scheduling*, volume 505 of *Lecture Notes in Economics and Mathematical systems*, pages 275–296. Springer, 2001.

[PK03] L. Peeters and L. Kroon. A variable trip time model for cyclic railway timetabling. *Transportation Science*, 37(2):198–212, 2003.

[PS16] J. Pätzold and A. Schöbel. A Matching Approach for Periodic Timetabling. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany, 2016.

[PSSS17] J. Pätzold, A. Schiewe, P. Schiewe, and A. Schöbel. Look-Ahead Approaches for Integrated Planning in Public Transportation. In Gianlorenzo D'Angelo and Twan Dollevoet, editors, *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–16, Dagstuhl, Germany, 2017.

104

[SAP+18] A. Schiewe, S. Albert, J. Pätzold, P. Schiewe, A. Schöbel, and J. Schulz. LinTim: An integrated environment for mathematical public transport optimization. Documentation. Technical Report 2018-08, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen, 2018.

[Sch12] A. Schöbel. Line planning in public transportation: models and methods. *OR Spectrum*, 34(3):491–510, 2012.

[Sch17] A. Schöbel. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research C*, 74:348–365, 2017.

[Sch18] P. Schiewe. *Integrated Optimization in Public Transport Planning*. PhD thesis, Georg-August-Universität Göttingen, 2018.

[SS06] A. Schöbel and S. Scholl. Line planning with minimal travel time. In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, number 06901 in Dagstuhl Seminar Proceedings, 2006.

[SS15] M. Schmidt and A. Schöbel. Timetabling with passenger routing. *OR Spectrum*, 37:75–97, 2015.

[SU89] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematic*, 2:550–581, 1989.

[TK00] W. Tan and B. Khoshnevis. Integration of process planning and scheduling - a review. *Journal of Intelligent Manufacturing*, 11(1):51–63, 2000.

[vdHvdAvK08] A. van den Heuvel, J. van den Akker, and M. van Kooten. Integrating timetabling and vehicle scheduling in public bus transportation. Technical report, Utrecht University, 2008.

[VKM17] L.P. Veelenturf, L.G. Kroon, and G. Maroti. Passenger oriented railway disruption management by adapting timetables and rolling stock schedules. *Transportation Research C*, 80:133–147, 2017.

[Zwa97] P.J. Zwaneveld. *Railway Planning — Routing of trains and allocation of passenger lines*. PhD thesis, School of Management, Rotterdam, 1997.

C. The Trickle-in Effect: Modeling Passenger Behavior in Delay Management

A. Schöbel, J. Pätzold, J. P. Müller
The Trickle-in Effect: Modeling Passenger Behavior in Delay Management
Submitted to ATMOS 2019.

The Trickle-in Effect: Modeling Passenger Behavior in Delay Management

Anita Schöbel[1], Julius Pätzold[2], and Jörg P. Müller[3]

[1] *Technical University of Kaiserslautern, Gottlieb-Daimler-Straße, 67663 Kaiserslautern, Germany, schoebel@mathematik.uni-kl.de and Fraunhofer-Institute for Industrial Mathematics ITWM, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany, anita.schoebel@itwm.fraunhofer.de*
[2] *University of Göttingen, Lotzestr. 16-18, 37083 Göttingen, Germany, j.paetzold@math.uni-goettingen.de*
[3] *Clausthal University of Technology, Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany, joerg.mueller@tu-clausthal.de*

May 15, 2019

Abstract

Delay management is concerned with making decisions if a train should wait for passengers from delayed trains or if it should depart on time. Models for delay management exist and can be adapted to capacities of stations, capacities of tracks, or respect vehicle and driver schedules, passengers' routes and further constraints. Nevertheless, what has been neglected so far, is that a train cannot depart as planned if passengers from another train trickle in one after another such that the doors of the departing train cannot close. This effect is often observed in real-world, but has not yet been taken into account in delay management.

We show the impact of this "trickle-in" effect to departure delays of trains under different conditions. We then modify existing delay management models to take the trickle-in effect into account. This can be done by forbidding certain intervals for departure. We present an integer programming formulation with these additional constraints resulting in a generalization of classic delay management models. We analyze the resulting model and identify parameters with which it can be best approximated by the classical delay management problem.

Experimentally, we show that the trickle-in effect has a high impact on the overall delay of public transport systems. We discuss the impact of the trickle-in effect on the objective function value and on the computation time of the delay management problem. We also analyze the trickle-in effect for timetables which have been derived without taking this particular behavioral pattern of passengers into account.

1 Introduction

Delays constitute a major source of uncertainty when operating a railway or bus system. If a train is delayed, many rescheduling decisions have to be made, disturbing the nominal schedule of a public transport system. The question, whether an otherwise punctual train should wait for a delayed feeder train in order to allow transferring passengers to reach their connections, is known as *delay management problem* and has been studied extensively in the literature. The first papers dealing with this kind of question date back

to [Sch01, SBK01]. Integer programming models have been developed in [Sch07, DHL08]. In order to make them more realistic, capacities along tracks have been included in [SS10], capacities at stations have been included in [DHK+15] and passenger re-routing has been studied in [DHSS12, SS15, RLB+17]. Rescheduling of timetables, rolling stock and crew is studied in [DHK+17]. For all these cases algorithms have been developed, see [DHSS18] for a recent survey on the state of the art.

Delay management aims at minimizing passengers' delays by taking the propagation of delays into account. Delays propagate along driving activities, i.e., if a train departs with some delay, then it also arrives at its next station with some delay – reduced by buffer time possibly included in the timetable. Delays also propagate along waiting activities in stations: If a train arrives at a station with some delay, it will probably also depart with some delay which again might have been reduced by buffer time. Finally, delays can propagate along changing activities as well. This is the case if a dispatcher decides that a connection from one train to another train should be maintained. Then the outgoing train will receive some delay by waiting for the delayed feeder train.
Nevertheless, there is an effect that has been neglected in the literature so far: A dispatcher may decide that a train should depart on time, but it may not be possible for the train to do so. To illustrate this issue, suppose a delayed train A arrives at a station and some of its passengers want to transfer at this station to another train B. The delay management problem requires a decision, whether train B should depart on time or wait for the passengers from train A.

- If train B is supposed to depart before train A has arrived, the delay management models work correctly. In this case, no delay propagates from train A to train B.
- If train B is supposed to wait long enough, the delay management models also work correctly and the delay propagates along the changing activity to the departure of train B.
- If, however, train B is supposed to depart shortly after train A has arrived without waiting for the passengers from train A, then the models fail. This is the case because normal delay management models assume that there is one common time that passengers need for walking from train A's platform to train B's platform. Instead, there may be quick and slow passengers. If the fastest passenger reaches train B before its departure, she can board. While getting onto train B, another fast passenger might arrive and while he boards, the next one will arrive, and so forth. In this way, all passengers might enter the train in a continuous stream preventing the train doors to close. Train B hence has to wait until finally even the slowest passengers from train A arrive and board train B. This effect has been simulated in [AKMS18] where it is called *trickle-in effect*.
- The same effect may also prolong the waiting time of train B in the case that B is supposed to wait for the passengers of train A since it may take longer to allow all passengers to trickle in than the lower bounds on the changing times suggest.

Note that the trickle-in effect is not only triggered by passengers not moving with the same speed, but also by the fact that passengers are not able to unboard train A instantaneously. Most readers will have experienced the situation of standing in a train corridor while waiting a decent amount of time for the passengers in front of them to unboard. This can result in a different transfer time of two passengers, even though they are able to walk with similar speed.
As a consequence, there exists a time interval in which train B is not able to depart, namely between the arrival of the fastest passenger and the arrival of the slowest passenger (assuming that there is no gap in speed of the passengers big enough to allow the doors of train B to close). We will call this interval *trickling interval*.

[AKMS18] show that the trickle-in effect, which can also be observed in many real-world situations, is in fact relevant. Our experiments (see Section 5) show that delay management decisions, which are optimal in the sense of classical delay management models, often schedule trains to depart in the "forbidden" trickling interval. If, for example, the trickling interval is $(2, 5)$ minutes and if all changing activities are distributed uniformly in $[2, 62)$ minutes (assuming a time period of 60 minutes), we can expect about 5% of all train departures to lie in the trickling interval. These departures are most likely not realizable and will cause additional delays. Hence, it is necessary to add this additional constraint to delay management models which is exactly what we do in this paper. We show how such an additional constraint can be included in classical delay management models, subsequently analyze the mathematical relation between the classical model and the one with the additional constraints, and finally show in experiments that delay management strategies change if the trickle-in effect is considered. We believe that by adding this detail we take a further step in bringing delay management models closer to practice.

The remainder of the paper is structured as follows. In Section 2 we recap the classical model for delay management. Section 3 models the trickle-in effect by introducing an additional constraint to the classical delay management model. We investigate theoretical consequences when adding the trickle-in effect to the classical delay management model in Section 4. Section 5 studies its practical effects in an experimental study on close-to-real-world data from LinTim [GSS13, SAP+18]. Integrating the trickle-in effect in models for (periodic) timetabling is identified as an extension and briefly discussed in Section 6, where we also conclude the contributions, discuss limitations of our work as well as venues of future research.

2 The Classical Delay Management Model

The *delay management problem* is defined as follows: Given an event-activity network, a timetable and some source delays, decide which connections should be maintained and which should be dropped such that the average delay of all passengers at their final destinations is minimal. The delay management problem was first introduced in [Sch01], a recent overview is give in [DHSS18].

We hence have to first introduce the concept of *event-activity networks* (see [Nac98] for its application in timetabling and [Sch07] for its application in delay management). An event-activity network is a directed graph $\mathcal{N} = (\mathcal{E}, \mathcal{A})$, where $\mathcal{E}$ consists of arrival and departure events $\mathcal{E}_{\text{arr}}$ and $\mathcal{E}_{\text{dep}}$, respectively. A timetable $\pi \in \mathbb{N}^{|\mathcal{E}|}$ assigns each event $i \in \mathcal{E}$ to a time $\pi_i \in \mathbb{N}$. If a delay occurs, the given timetable π has to be updated to a so-called *disposition timetable* $x \in \mathbb{N}^{|\mathcal{E}|}$. To represent the constraints that have to be satisfied by a (disposition) timetable, we need the following types of activities, $\mathcal{A} = \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{change}}$. Each of them is assigned to a minimal duration $L_a > 0$. The meaning of these activities is given as follows (see also Figure 1).

- *Driving activities* $\mathcal{A}_{\text{drive}} \subset \mathcal{E}_{\text{dep}} \times \mathcal{E}_{\text{arr}}$ model the driving of a train between two consecutive stations, i.e. a driving activity connects a departure event of some train with its next arrival event. The duration $L_a > 0$ of a driving activity $a = (i, j)$ represents the minimal necessary driving time between the departure event i and the arrival event j. Note that turnaround edges may be handled in the same way as driving activities.
- *Waiting activities* (also called *dwelling activities*) $\mathcal{A}_{\text{wait}} \subset \mathcal{E}_{\text{arr}} \times \mathcal{E}_{\text{dep}}$ represent the time period in which a train is waiting at a station to let passengers get on or off; a waiting activity hence connects an arrival event of some train with its next departure event. Its duration $L_a > 0$ describes the minimal time required to allow boarding and unboarding; sometimes it also includes exchanging train crews or other actions.

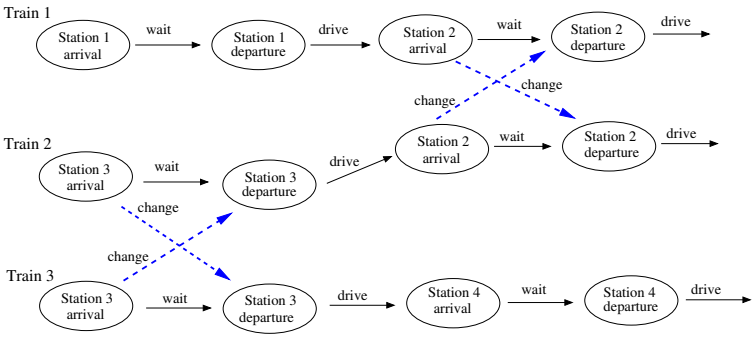


Figure 1: An event-activity network with three trains and four stations. The solid (black) arcs represent driving and waiting activities of the trains. The dashed (blue) arcs represent changing activities which are possible between Train 2 and Train 3 at Station 3 and between Train 1 and Train 2 at Station 2.

If two events $i, j \in \mathcal{E}$ are connected by an activity $(i,j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}$, then event i has to be performed before event j can take place. In particular, the disposition timetable x has to satisfy

$$x_j - x_i \geq L_a$$

for all $a = (i,j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}$.

- *Changing activities* $\mathcal{A}_{\text{change}} \subset \mathcal{E}_{\text{arr}} \times \mathcal{E}_{\text{dep}}$ allow passengers to transfer from an incoming train to an outgoing train. Hence, a changing activity connects an arrival event of some train at some station with a departure event of another train at the same station. The lower bound $L_a > 0$ refers to the minimum time a passenger needs to transfer between both trains. In order to solve the delay management problem we have to decide for each changing activity if it should be kept or if it can be deleted. In case that a changing activity $a = (i,j)$ is kept, the disposition timetable x must satisfy $x_j - x_i \geq L_a$. If the changing activity is deleted, the outgoing train can depart without waiting for the incoming train and this inequality does not need to be satisfied anymore.

We remark that other types of activities such as *headway activities* or *turnaround activities* may be added, see [DHSS18] for the respective models. Notwithstanding that, in this work we focus on the classical model.

To formulate an integer programming model of the delay management problem, we next have to formally introduce the delays. We assume that a set of unexpected *source delays* is known, e.g., caused by signaling problems, construction work, accidents, or bad weather conditions. These source delays cause *secondary delays*, e.g., for the same train at subsequent stations or for other trains that wait for the delayed train. In our work we allow two types of source delays: The first type is a delay $d_i \in \mathbb{N}$ at an event $i \in \mathcal{E}$ (e.g., staff coming too late to their duty) referring to a fixed point of time. In this case, $x_i \geq \pi_i + d_i$ is required. The second type of source delay is a delay d_a which increases the duration of an activity $a = (i,j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}$, e.g., an increase of traveling time between two stations due to construction work. Such a delay d_a has to be added to the minimal duration L_a of activity a. If an event or an activity has no source delay, we assume $d_i = 0$ or $d_a = 0$, respectively, to simplify the notation.

In the objective function we evaluate the disposition timetable from the passengers' point of view. To this end, let w_i be the number of passengers unboarding the train at event

$i \in \mathcal{E}$ (thus, $w_i = 0$ for all $i \in \mathcal{E}_{\text{dep}}$) and w_a be the number of passengers who want to use a changing activity $a \in \mathcal{A}_{\text{change}}$. We assume $w_a > 0$ for all $a \in \mathcal{A}_{\text{change}}$ – otherwise, the changing activity could be removed from the network, since nobody uses it. We further assume that all lines have a common period T, i.e., every line is served by a train every T minutes. Note that this assumption can be relaxed by introducing periods T_a for all changing activities $a \in \mathcal{A}_{\text{change}}$.

We can now state the integer programming formulation for the basic version of the delay management problem. To model the wait-depart decisions, i.e., whether some train should wait for some other train at a station or not, we introduce binary variables

$$z_a = \begin{cases} 0 & \text{if changing activity } a \text{ is maintained} \\ 1 & \text{otherwise} \end{cases}$$

for all changing activities $a \in \mathcal{A}_{\text{change}}$. The integer programming formulation then reads as follows:

$$\min \sum_{i \in \mathcal{E}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} z_a w_a T \tag{DM}$$

$$\begin{aligned}
\text{s.t.} \quad x_i &\geq \pi_i + d_i && \forall i \in \mathcal{E} && (1)\\
x_j - x_i &\geq L_a + d_a && \forall a = (i,j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} && (2)\\
M z_a + x_j - x_i &\geq L_a && \forall a = (i,j) \in \mathcal{A}_{\text{change}} && (3)\\
x_i &\in \mathbb{N} && \forall i \in \mathcal{E} \\
z_a &\in \{0,1\} && \forall a \in \mathcal{A}_{\text{change}}
\end{aligned}$$

where M is a fixed constant. The meaning of the objective function and of the constraints is explained next.

The first term of the objective function minimizes the sum of all delays of all events. If all connections were maintained, this would be the sum of delays for all passenger at their final destination. The second term adds the weighted sum of all missed connections with a penalty of one time period T (or T_a if we drop the assumption of a common period of all lines) a passenger has to wait for the next train of the same line. The objective function is hence an approximation of the sum of all delays over all passengers. It has been shown in [Sch07] that it is not an approximation, but exactly computes the sum of all passengers' delays if the so-called never-meet property holds.

Constraints (1) and (2) ensure that the delay is passed on correctly along driving and waiting activities. (3) does the same for maintained changing activities (i.e. if $z_a = 0$). If, however, $z_a = 1$, constraints (3) get redundant if M is chosen big enough. If no capacity constraints are considered and $d_a = 0$ for all $a \in \mathcal{A}$, [Sch07] shows that choosing M as the largest source delay $\max_{i \in \mathcal{E}} d_i$ is sufficient. Solution methods for (DM) mainly rely on integer programming, see [DHSS18] and references therein.

3 Modeling the Trickle-in Effect

In this section we adapt the classical delay management model (DM) by taking the following two phenomena into account:

1. Passengers do not change with the same speed. There may be fast and slow passengers and a decision for keeping a changing activity means practically that the train waits for all (even for the slowest) passengers.

2. Due to the trickle-in effect, trains are not able to depart while passengers are still boarding.

The first point is modeled by using a time interval $(L_a^{\min}, L_a^{\max})$ instead of a fixed time L_a to describe the duration of the changing activities. We hence replace L_a in constraints (3) by $L_a^{\max}$. The second point implies that a train can either depart before the fastest passenger has arrived or after the slowest one has boarded, i.e., it cannot depart in the interval $(x_i + L_a^{\min}, x_i + L_a^{\max})$. This restriction is modeled by adding new constraints as follows.

Lemma 1. *Let $a = (i, j) \in \mathcal{A}_{\text{change}}$. There exists $z_a \in \{0, 1\}$ such that*

$$Mz_a + x_j - x_i \geq L_a^{\max} \tag{4}$$
$$M(z_a - 1) + x_j - x_i \leq L_a^{\min} \tag{5}$$

are both satisfied if and only if

$$x_j \notin (x_i + L_a^{\min}, x_i + L_a^{\max}). \tag{6}$$

Proof. Let (4) and (5) hold for some $z_a \in \{0, 1\}$. If $z_a = 0$, (4) reduces to $x_j \geq x_i + L_a^{\max}$. On the other hand, if $z_a = 1$ then (5) reduces to $x_j \leq x_i + L_a^{\min}$. In both cases, $x_j \notin (x_i + L_a^{\min}, x_i + L_a^{\max})$.

Vice versa, let $x_j \notin (x_i + L_a^{\min}, x_i + L_a^{\max})$. If $x_j \leq x_i + L_a^{\min}$ we choose $z_a = 1$ to see that both, (4) and (5) hold. On the other hand, if $x_j \geq x_i + L_a^{\max}$ then $z_a = 0$ guarantees that (4) and (5) are satisfied. □

The proof of Lemma 1 specifies two possible cases for a dispatcher:

- The changing activity is maintained ($z_a = 0$) if and only if the train departs after the last passengers have boarded $x_j \geq x_i + L_a^{\max}$.
- The changing activity is dropped ($z_a = 1$) if and only if the train departs before the first passengers have boarded $x_j \leq x_i + L_a^{\min}$.

The resulting model (DM-trick) is hence given as

$$\min \quad \sum_{i \in \mathcal{E}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} z_a w_a T \tag{DM-trick}$$

$$\begin{aligned}
\text{s.t.} \quad x_i &\geq \pi_i + d_i && \forall i \in \mathcal{E} && (7)\\
x_j - x_i &\geq L_a + d_a && \forall a = (i, j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} && (8)\\
Mz_a + x_j - x_i &\geq L_a^{\max} && \forall a = (i, j) \in \mathcal{A}_{\text{change}} && (9)\\
M(z_a - 1) + x_j - x_i &\leq L_a^{\min} && \forall a = (i, j) \in \mathcal{A}_{\text{change}} && (10)\\
x_i &\in \mathbb{N} && \forall i \in \mathcal{E}\\
z_a &\in \{0, 1\} && \forall a \in \mathcal{A}_{\text{change}}
\end{aligned}$$

We remark that (DM-trick) contains (DM) as a special case by setting $L_a^{\max} := L_a$ and $L_a^{\min} := L_a$ for all $a \in \mathcal{A}$, i.e., it is a proper extension of the classical delay management model. Trickle-in constraints can also be combined with all other extensions known for delay management, i.e., it is possible to consider headway constraints as in [SS10], station capacity constraints as in [DHK+15], or passenger routing constraints as in [DHSS12]. For the sake of simplicity we compare (DM) and (DM-trick) in their basic versions as given above.

4 Analyzing the New Model

As already mentioned in Section 2, for the classical delay management problem it suffices to choose M as large as the largest source delay $D := \max_{i \in \mathcal{E}} d_i$ if all $d_a = 0$. This does not hold any more for (DM-trick), but still the size of M can be bounded. To this end, we need the following two lemmas, both dealing with the original timetable π_i, $i \in \mathcal{E}$. For this chapter, we assume that the original timetable π is feasible, i.e., that

$$\pi_j - \pi_i \in [L_a, U_a] \quad \forall a = (i,j) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}, \tag{11}$$

for all driving and waiting activities. For the changing activities we assume that the trickling constraints (6) applied to the original timetable π

$$\pi_j \notin (\pi_i + L_a^{\min}, \pi_i + L_a^{\max}) \quad \forall a = (i,j) \in \mathcal{A}_{\text{change}} \tag{12}$$

are satisfied, i.e., either nobody can change or everybody can. However, *changing* activities are the ones that allow passengers to change, so the case $\pi_j - \pi_i \leq L_a^{\min}$ cannot hold. We hence may assume that

$$\pi_j - \pi_i \in [L_a^{\max}, T + L_a^{\min}] \quad \forall a = (i,j) \in \mathcal{A}_{\text{change}} \tag{13}$$

where the upper bound $T + L_a^{\min}$ holds since every line runs at least once per time period T.

In order to simplify the notation, we will sometimes use the *delay* y_i of an event $i \in \mathcal{E}$ in its disposition timetable, which is defined as

$$y_i := x_i - \pi_i.$$

Lemma 2. *Let π_i, $i \in \mathcal{E}$ be a feasible timetable. If all $d_a = 0$, then there exists an optimal solution with $y_j \leq D$ for all $a = (i,j) \in A$.*

Proof. The proof works by induction. Since the event-activity network does not contain any directed cycles, we can sort the events $i \in \mathcal{E}$ topologically. Let $i_1, \ldots, i_{|\mathcal{E}|}$ be the resulting order. Then the delay y_{i_1} of the first event i_1 is given by $d_{i_1} \leq D$. Now take any other event j and consider all of its incoming activities $(i,j) \in \mathcal{A}$. We now estimate how large the delay of event j can be. Note that there exists an optimal solution in which no disposition time can be reduced (i.e., which does not contain any unnecessary delays). This means one of the inequality constraints (7), (8), (9) is sharp.

- If (7) is sharp we get $x_j = \pi_j + d_j$, hence $y_j = x_j - \pi_j = d_j \leq D$.
- If (8) is sharp for $(i,j) \in \mathcal{A}$ we have that $x_j = x_i + L_a$, i.e., the delay of event j can be computed as

$$\begin{aligned} y_j &= x_j - \pi_j = L_a + x_i - \pi_j \\ &= L_a + y_i + \underbrace{\pi_i - \pi_j}_{\leq -L_a} \\ &\leq y_i \leq D \text{ by induction hypothesis} \end{aligned}$$

  where we have used feasibility of the timetable, see (11) and that event i is topologically smaller than event j.
- If (9) is sharp for $(i,j) \in \mathcal{A}$ we analogously have that $x_j = x_i + L_a^{\max}$, i.e., the delay of event j can be computed as

$$\begin{aligned} y_j &= x_j - \pi_j = L_a^{\max} + x_i - \pi_j \\ &= L_a^{\max} + y_i + \underbrace{\pi_i - \pi_j}_{\leq -L_a^{\max}} \\ &\leq y_i \leq D \text{ by induction hypothesis} \end{aligned}$$

where we have used the second feasibility constraint for the timetable, see (13) and again that event i is topologically smaller than event j.

□

Under the same conditions as in the above lemma we can hence estimate the size of big M, which is a bit larger than in (DM) but still of moderate size.

Lemma 3. *If $d_a = 0$ for all $a \in \mathcal{A}$, then $M = T + D$ is large enough to correctly solve Model* (DM-trick).

Proof. We have to find M that satisfies the following two conditions:

1. Constraint (4) should get redundant if $z_a = 1$, i.e., for an optimal solution we require that $M \geq L_a^{\max} + x_i - x_j$. We hence look for an upper bound of the right hand side:

$$\begin{aligned} L_a^{\max} + x_i - x_j &= L_a^{\max} + \pi_i + y_i - \pi_j - y_j \\ &= L_a^{\max} + \underbrace{\pi_i - \pi_j}_{\leq -L_a^{\max}} + \underbrace{y_i}_{\leq D} \underbrace{-y_j}_{\leq 0} \leq D, \end{aligned}$$

   where we again used feasibility of the timetable, see (13).

2. Constraint (5) should get redundant if $z_a = 0$, i.e., for an optimal solution we require that $M \geq x_j - x_i - L_a^{\min}$. We again need an upper bound of the right hand side:

$$\begin{aligned} x_j - x_i - L_a^{\min} &= \pi_j + y_j - \pi_i - y_i - L_a^{\min} \\ &= \underbrace{\pi_j - \pi_i}_{\leq T + L_a^{\min}} + \underbrace{y_j}_{\leq D} \underbrace{-y_i}_{\leq 0} - L_a^{\min} \leq T + D, \end{aligned}$$

   this time using the upper bound in (13).

We conclude that $M = D + T$ suffices for both constraints (4) and (5). □

In the case of $d_a > 0$, delays can increase for single trains and have to be bounded. This can theoretically be done by summing up all delays d_a or (better) by finding a longest path P in the event-activity network with respect to the weights d_a, see [SS10].

Let us now consider the case that the timetable is feasible according to its traditional definition *without* the trickle-in effect, i.e., it satisfies $\pi_j - \pi_i \in [L_a, T + L_a - 1]$ instead of (13) for some $L_a < L_a^{\max}$. Then the trickle-in effect may generate delays.
Let us illustrate this on a small example: Given a timetable π that schedules train A to arrive at 10:00 and train B to depart at 10:02 and given a trickling interval of $(1, 3)$ minutes, then the trickle-in effect is observable. The first passengers only need a little bit more than one minute to catch the train, but then a continuous stream of passengers boards the train leading to a delayed departure of train j at 10:03, i.e., to a delay of one minute. Thus, there may occur delays due to the trickle-in effect without the existence of any source delays.

However, even in this situation we can use (DM-trick) to find optimal wait-depart decisions dealing with both, source delays and delays occurring due to trickling constraints, and even in this situation we can bound M. To this end, assume a changing activity $a = (i, j)$ from event i to event j for which we have $L_a < \pi_j - \pi_i < L_a^{\max}$. Then the transfer of all passengers may take longer than the timetable allows. Hence, the trickle-in effect leads to a new type of "source delay" on this changing activity, namely a delay of $L_a^{\max} - (\pi_j - \pi_i)$. In order to find a bound for M we hence need to search for a longest path P' with respect to the weights

$$w'_a := \begin{cases} \max(0, d_a) & \text{if } a \in \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}} \\ \max(0, d_a, L_a^{\max} - (\pi_j - \pi_i)) & \text{if } a = (i, j) \in \mathcal{A}_{\text{change}} \end{cases}$$

and add its length to M. Hence, we receive a bound of

$$M = D + T + \text{length}(P')$$

in this case.
The computational experiments show that $M = D + T + \text{length}(P')$ is of reasonable size and hence a sufficient upper bound for M.

We now analyze the new model (DM-trick) with respect to the intervals $[L_a^{\min}, L_a^{\max}]$ for the changing activities. Varying both, the lower and the upper bound on the duration of the changing activities gives the following result.

Lemma 4. *Let $I_a(k) = [L_a^{\min}(k), L_a^{\max}(k)]$ for all $a \in \mathcal{A}_{change}$ be a sequence of nested intervals with*

$$L_a^{\min}(1) \leq L_a^{\min}(2) \leq \cdots \leq L_a \text{ and } L_a \leq \cdots \leq L_a^{\max}(2) \leq L_a^{\max}(1)$$

and let $z^*(k)$ be the optimal objective function value for (DM-trick) *with respect to the interval $I_a(k)$ and z^* be the optimal objective function value of* (DM)*. Then*

$$z^1(1) \geq z^*(2) \geq \cdots \geq z^*.$$

Proof. Since $I_a(k+1) \subseteq I_a(k)$ for all $a \in \mathcal{A}_{change}$, (DM-trick) with respect to the intervals $I(k+1)$ is a relaxation of (DM-trick) with respect to the intervals $I(k)$ and the result follows. □

As a consequence, (DM) is a relaxation of (DM-trick) whenever the changing times L_a in the classical model (DM) satisfy $L_a \in [L_a^{\min}, L_a^{\max}]$ for all $a \in \mathcal{A}_{change}$. Hence, solving (DM) gives a lower bound on (DM-trick). In the experiments in Section 5 we compare the gap between this lower bound and the real solution. The best approximation of (DM-trick) by (DM) is given if we set $L_a := L_a^{\max}$ for all $a \in \mathcal{A}_{change}$, i.e., making sure that also the slow passengers are able to board their next train. This is shown in the next Lemma.

Lemma 5. *Let $[L_a^{\min}, L_a^{\max}]$ for all $a \in \mathcal{A}_{change}$ be the given data for* (DM-trick)*. Let $z^*(L_a)$ be the optimal objective function value for* (DM) *with data L_a for all $a \in \mathcal{A}_{change}$. Then an optimal solution to*

$$\max\{z^*(L_a) : L_a \in [L_a^{\min}, L_a^{\max}] \text{ for all } a \in \mathcal{A}_{change}\}$$

is provided by setting $L_a = L_a^{\max}$ for all $a \in \mathcal{A}_{change}$, i.e., the best lower bound obtainable from the classical model (DM) *is provided by setting $L_a := L_a^{\max}$ for all $a \in \mathcal{A}$.*

Proof. From Lemma 4 we already know that all $L_a \in [L_a^{\min}, L_a^{\max}]$ provide lower bounds. We hence have to show that the largest of them is obtained by setting $L_a := L_a^{\max}$ for all $a \in \mathcal{A}$. To this end, let $L'_a \leq L_a^{\max}$ for all $a \in \mathcal{A}$. Let (x, z) be a solution of (DM) with respect to $L_a^{\max}$. It hence satisfies (3) with $L_a^{\max}$ on the right hand side and hence also with $L'_a \leq L_a^{\max}$ on the right hand side. Hence, (x, z) is also feasible for (DM) with respect to L'_a. We conclude that (DM) with respect to L'_a is a relaxation of (DM) with respect to $L_a^{\max}$, and hence

$$z^*(L'_a) \leq z^*(L_a^{\max}).$$

□

The computational results underline that using (DM) as a relaxation for (DM-trick) impose a good trade-off between computation time and (DM)'s quality as a lower bound.

5 Experiments

In this section we investigate the effects of the trickle-in effect computationally. To this end, we implemented (DM-trick) in LinTim, an open source software framework for public transport optimization, see [SAP+18, GSS13]. We focus on solving the *bahn* dataset, consisting of 250 nodes and 326 edges, modeling the German ICE network.

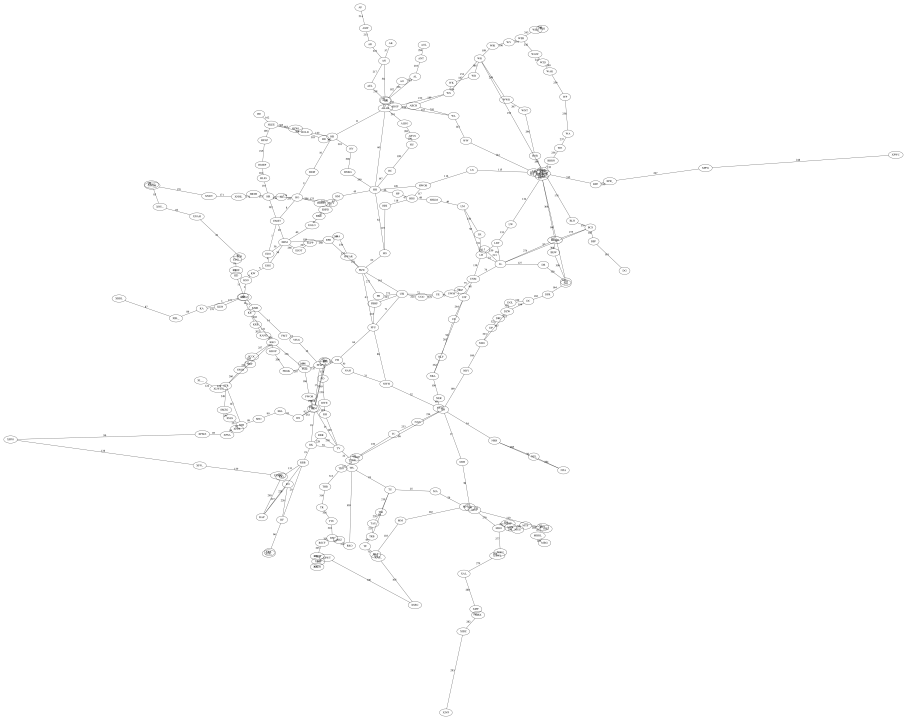


Figure 2: *bahn* dataset

For quickly determining the timetable we use the MATCH heuristic as described in [PS16] since it is faster than the modulo simplex [GS13, NO08] or integer programming approaches [LPW08]. We roll out the periodic timetable for 4 hours and receive an aperiodic event-activity-network with around 20000 events and 40000 activities. For generating delays we use a LinTim procedure which is parameterized to choose 1000 activities and to generate source delays uniformly distributed between 1 and 900 seconds for each of the chosen activities. In order to calculate a sufficiently big M as described in Section 4, we calculate length$(P) = 3500$ seconds and $D = 0$ (because we generate source delays only on activities) and T was chosen to be 3600 seconds, leading to a choice of $M = 7100$. We implemented (DM-trick) using Gurobi 8.0 with a relative optimality gap of 1% and run the experiments on a compute server with 12 cores of Intel(R) Xeon(R) CPU X5650 @ 2.67 GHz and 78 GB RAM.

In our first experiment we compare different trickling intervals with the lower bound $L_a^{\min}$ ranging from 60 to 180 seconds and $L_a^{\max}$ ranging from 180 to 300 seconds. The default minimum changing time L_a is assumed to be 180 seconds. The objective values, given in passengers times seconds, for solving (DM-trick) with these different intervals are depicted in Figure 3.

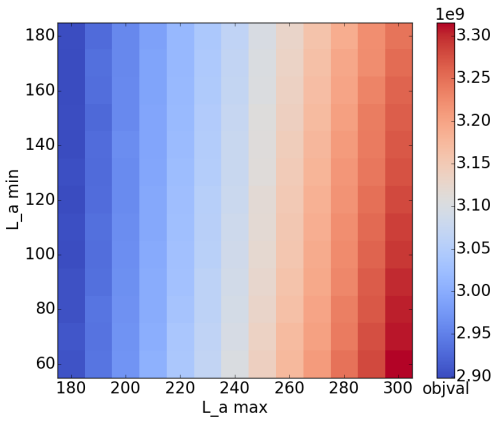


Figure 3: objective values for different trickling intervals

We see that the instance with the smallest trickling interval ($[180, 180]$ seconds) has the lowest objective value of about $2.9 \cdot 10^9$, whereas the instance with the largest trickling interval ($[60, 300]$ seconds) has the largest objective value ($3.3 \cdot 10^9$). This is consistent with the theory since small trickling intervals are a relaxation of larger trickling intervals, see Lemma 4. A higher objective value is equivalent to higher passenger delays in the event-activity-network which makes sense as a larger trickling interval potentially leads to longer waiting times for trains. In general, one can observe that a larger interval correlates with a higher objective value, and furthermore that a change in $L_a^{\max}$ has a higher impact on the objective value than a change in $L_a^{\min}$.
Figure 4 now depicts the runtimes for different choices of the trickling interval. Interest-

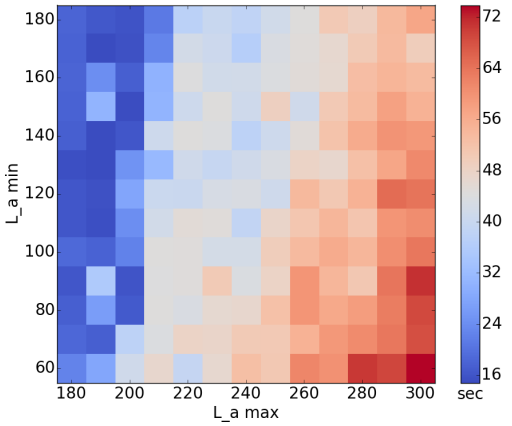


Figure 4: runtimes for different trickling intervals

ingly, also the instance with the smallest trickling interval has the lowest runtime and the instance with the largest trickling interval has the highest runtime. Also for the other

instances the runtimes correlate primarily with the size of the trickling interval (although not as smoothly as the objective value) and a change in the upper bound $L_a^{\max}$ has again a higher impact on the runtime than a change in $L_a^{\min}$. The correlation between size of the trickling interval and runtime can be explained by the nature of integer programming. If the size of the "forbidden" trickling interval is increased, we get a weaker linear relaxation of the integer problem and hence need longer to solve it, e.g., via branch-and-bound.

The next experiment investigates the difference between a disposition timetable found by (DM) and a disposition timetable that respects the trickle-in effect. To this end, Figure 5 depicts the number of changing activities of a disposition timetable from (DM) (with $L_a = 180$ seconds) that lie in the trickling interval. Hence, Figure 5 illustrates the difference between a disposition timetable found by solving (DM) and disposition timetables found by solving (DM-trick) for different trickling intervals. As can be seen in the figure, there exist up to 1194 infeasible change activities for a disposition timetable from (DM). In other words, if a disposition timetable from (DM) is found, it can be the case that 1194 changing activities (or about 6% of all changing activities) cause new delays due to the neglection of the trickle-in effect.

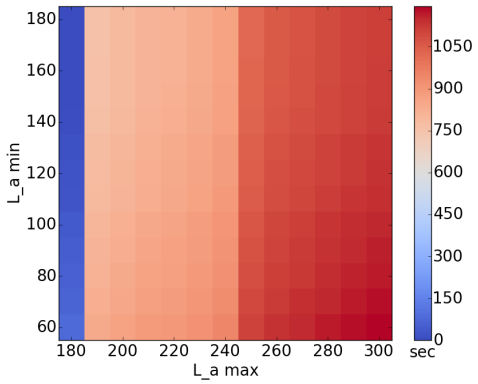


Figure 5: number of infeasible changing activities for a timetable from (DM) for different trickling intervals

Finally, we investigate the results of Lemma 5, i.e., that solving (DM) with $L_a \coloneqq L_a^{\max}$ yields the best approximation to (DM-trick). One can see in Figure 6 that the objective value indeed increases when L_a increases, culminating in a gap of only 3% if L_a is chosen to be $L_a^{\max}$. Hence, we get a reasonably good approximation of (DM-trick) by only solving an instance of (DM). Furthermore, it should be noted that solving (DM) takes only around 1 second, whereas solving (DM-trick) with trickling interval $[60, 300]$ seconds took around 77 seconds to solve. Hence, we indeed get a decent trade-off between computation time and solution quality.

As a final note, we also run the model (DM-trick) for the case if no source delays exist and received an objective value of around $5 \cdot 10^8$. This is roughly 15% of the objective value we encountered while working with the aforementioned 1000 source delays. Put differently, in this instance up to 15% of the delays might not be caused by source delays, but by the mere structure of the underlying periodic timetable and the trickle-in effect. Hence, the trickle-in effect has high relevance beyond delay management and should already be considered when planning a periodic timetable (which is not the case when using, e.g., MATCH for timetabling).

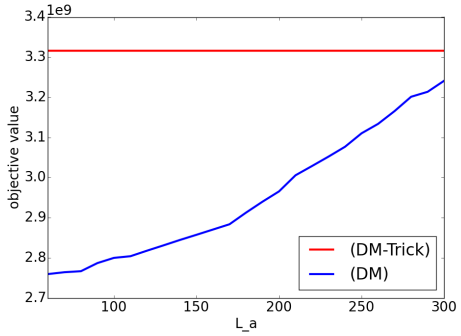


Figure 6: objective values for (DM) for different L_a

6 Conclusion and Suggestions for Further Research

In this paper we introduced the trickle-in effect, an observation on passenger behavior at train stations that highly influences delays in public transport. We introduced models for incorporating the trickle-in effect into standard delay management models and also showed how it already influences the periodic timetabling problem. We investigated mathematical properties of the resulting model and showed how (DM-trick) can be approximated best using the classical delay management problem. This allows to use approaches for classical delay management (such as [BS14, RLB+17, DHK+17, DH14]) for heuristically solving (DM-trick). The computational experiments underlined our hypothesis that the trickle-in effect has a high impact on delay management: Here, the objective value of (DM-trick) exceeds the objective value of (DM) up to 15%. Finally, since the computation times for (DM-trick) rise significantly, we still can get a decent approximation of (DM-trick) by solving a modified version of (DM).

Further research includes simulation approaches to better understand the behaviour of the passengers and to derive practically relevant trickling intervals. To this end, an agent-based simulation as in [ADKM18] is currently developed. We are also interested in adding the trickling constraints to more sophisticated delay management models including passengers' routing and capacity constraints.

Finally, there is another line of research, namely adding trickling constraints to the timetabling problem. In Section 4 we have already seen that considering the trickle-in effect in a timetable that is not feasible with respect to (13) might cause source delays. The experiments justify this theoretical observation: a timetable might get significant delays just because of the trickle-in effect, i.e., even if no other source delays occur. We hence suggest to consider the trickle-in effect already in the timetabling phase. This means to add constraints of type (12) in timetabling such that either all passengers or none of the passengers can make a transfer. Hence, $\pi_j - \pi_i \in [L_a^{\max}, T + L_a^{\min}]$ needs to be respected for all changing activities (i, j) and even more general for all activities (i, j) from an arrival event of an incoming train to a departure event of (another) outgoing train. These constraints can be transferred also to periodic timetabling and considered as additional constraints in the periodic event scheduling problem (PESP). The analysis of them (runtime, impact on resulting timetable) are an interesting topic for future research which seems to be challenging and highly relevant from a practical point of view.

References

[ADKM18] M. Aschermann, S. Dennisen, P. Kraus, and J.P. Müller. LightJason, a Highly Scalable and Concurrent Agent Framework: Overview and Application (demonstration paper). In M. Dastani, G. Sukthankar, E. Andre, and S. Koenig, editors, *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*, pages 1794 – 1796, 2018.

[AKMS18] S. Albert, P. Kraus, J.P. Müller, and A. Schöbel. Passenger-induced delay propagation: Agent-based simulation of passengers in rail networks. In *Simulation Science*, volume 889 of *Communications in Computer and Information Science (CCIS)*, pages 3–23. Springer, 2018.

[BS14] R. Bauer and A. Schöbel. Rules of thumb — practical online strategies for delay management. *Public Transport*, 6(1):85–105, 2014.

[DH14] T. Dollevoet and D. Huisman. Fast heuristics for delay management with passenger rerouting. *Public Transport*, 6(1-2):67–84, 2014.

[DHK+15] T. Dollevoet, D. Huisman, L. Kroon, M. Schmidt, and A. Schöbel. Delay management including capacities of stations. *Transportation Science*, 49(2):185–203, 2015.

[DHK+17] T. Dollevoet, D. Huisman, L.G. Kroon, L.P. Veelenturf, and J.C.Wagenaar. Application of an iterative framework for real-time railway scheduling. *Computers and Operations Research*, 78:203–217, 2017.

[DHL08] L. De Giovanni, G. Heilporn, and M. Labbé. Optimization models for the single delay management problem in public transportation. *European Journal of Operational Research*, 189(3):762–774, 2008.

[DHSS12] T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay management with rerouting of passengers. *Transportation Science*, 46(1):74–89, 2012.

[DHSS18] T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay propagation and delay management in transportation networks. In R. Borndörfer et al., editor, *Handbook of Optimization in the Railway Industry*. Springer, 2018.

[GS13] M. Goerigk and A. Schöbel. Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, 40(5):1363–1370, 2013.

[GSS13] M. Goerigk, M. Schachtebeck, and A. Schöbel. Evaluating line concepts using travel times and robustness: Simulations with the lintim toolbox. *Public Transport*, 5(3), 2013.

[LPW08] C. Liebchen, M. Proksch, and F.H. Wagner. Performances of algorithms for periodic timetable optimization. In *Computer-aided Systems in Public Transport*, pages 151–180. Springer, Heidelberg, 2008.

[Nac98] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. PhD thesis, University of Hildesheim, 1998.

[NO08] K. Nachtigall and J. Opitz. Solving periodic timetable optimisation problems by modulo simplex calculations. In *Proc. ATMOS*, 2008.

[PS16] J. Pätzold and A. Schöbel. A Matching Approach for Periodic Timetabling. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[RLB+17] R. Rückert, M. Lemnian, C. Blendinger, S. Rechner, and M. Müller-Hannemann. Panda: a software tool for improved train dispatching with focus on passenger flows. *Public Transport*, 9:307–324, 2017.

[SAP+18] A. Schiewe, S. Albert, J. Pätzold, P. Schiewe, A. Schöbel, and J. Schulz. LinTim: An integrated environment for mathematical public transport optimization. Documentation. Technical Report 2018-08, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen, 2018.

[SBK01] L. Suhl, C. Biederbick, and N. Kliewer. Design of customer-oriented dispatching support for railways. In S. Voß and J. Daduna, editors, *Computer-Aided Transit Scheduling*, volume 505 of *Lecture Notes in Economics and Mathematical systems*, pages 365–386. Springer, 2001.

[Sch01] A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.

[Sch07] A. Schöbel. Integer programming approaches for solving the delay management problem. In *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes in Computer Science, pages 145–170. Springer, 2007.

[SS10] M. Schachtebeck and A. Schöbel. To wait or not to wait and who goes first? Delay management with priority decisions. *Transportation Science*, 44(3):307–321, 2010.

[SS15] M. Schmidt and A. Schöbel. The complexity of integrating routing decisions in public transportation models. *Networks*, 65(3):228–243, 2015.

D. Approximate Cutting Plane Approaches for Exact Solutions to Robust Optimization Problems

J. Pätzold, A. Schöbel
Approximate Cutting Plane Approaches for Exact Solutions to Robust Optimization Problems
Submitted to European Journal of Operational Research (in revision), 2018.

Approximate Cutting Plane Approaches for Exact Solutions to Robust Optimization Problems

Julius Pätzold[a,1,*], Anita Schöbel[b]

[a] *Institute for Numerical and Applied Mathematics, University of Goettingen, Lotzestraße 16 - 18, 37083 Göttingen, Germany*
[b] *Department for Mathematics, TU Kaiserslautern and Fraunhofer Institute for Industrial Mathematics ITWM*

Abstract

In this paper we deal with cutting plane approaches for robust optimization. Such approaches work iteratively by solving a robust problem with reduced uncertainty set (*robustification step*) and determining a worst-case scenario in each iteration (*pessimization step*) which is then added to the reduced uncertainty set. We propose to enhance this scheme by solving the robustification and/or the pessimization step not exactly, but only approximately, that is, until an improvement to the current solution is possible. The resulting iterative approach is called *approximate* cutting plane approach.

We prove that convergence to an optimal solution for approximate cutting plane approaches is still guaranteed under similar assumptions as for classical cutting plane approaches, in which both robustification and pessimization problem are solved exactly in each iteration. Experimentally, we investigate robust mixed integer linear optimization problems for mixed-integer polyhedral uncertainty sets of different difficulties. Solving the robustification or pessimization problem only approximately increases the number of iterations. Nevertheless, our results show that the approximate cutting plane approach becomes more efficient, in particular, if the robustification or the pessimization problem is hard.

Keywords: Robustness and Sensitivity Analysis, Robust Optimization, Mixed Integer Programming, Cutting-Plane Methods

1. Introduction

Almost every optimization problem suffers to some extent from uncertainty. In mathematics there exist different approaches to overcome this issue. In comparison to stochastic optimization, robust optimization does not require any distributional assumptions for the uncertain data. Instead, robustness concepts minimize the performance of a solution in its worst case.

[*] Corresponding author
Email addresses: j.paetzold@math.uni-goettingen.de (Julius Pätzold), schoebel@mathematik.uni-kl.de (Anita Schöbel)
[1] Supported by the Simulation Science Center Clausthal/Göttingen.

Robust optimization started with the work of Soyster (1973) and was extensively researched later, e.g., in Ghaoui & Lebret (1997); Ben-Tal & Nemirovski (1998, 2000); Bertsimas & Sim (2004); Ben-Tal et al. (2006), see Ben-Tal et al. (2009) for a compendium on results for strict robustness, Kouvelis & Yu (1997) for algorithms and applications for strict and regret robustness, and Goerigk & Schöbel (2016) for a survey on less conservative robustness concepts.

Solving robust counterparts of optimization problems is a challenging task due to the added nonlinearities of considering an uncertainty set. Nevertheless, there exist cases for which uncertain optimization problems can be solved exactly via reformulation (see, e.g., Ben-Tal & Nemirovski (2000); Ben-Tal et al. (2009); Bertsimas et al. (2011) and Goerigk & Schöbel (2016) for more results). On the other hand, there exist iterative (or cutting plane) approaches for tackling robust optimization problems, which are also the main focus of this paper. The standard iterative approach works in the following manner: Beginning with a small number of scenarios we determine a solution to a reduced robust problem, in which only this small uncertainty set is considered. For the resulting solution, a new scenario is determined and added to the set of scenarios. The procedure is then repeated.

This iterative approach has often been used in previous research and has been proposed and used under many different names, including Outer Approximation Method (Reemtsen (1994), Bürger et al. (2014), Goerigk & Schöbel (2016)), (modified) Benders Decomposition Approach (Montemanni (2006), Siddiqui et al. (2011)), Implementor-Adversarial Framework (Bienstock (2007)), Cutting set/plane Method (Mutapcic & Boyd (2009), Fischetti & Monaci (2012)) or Scenario Relaxation Procedure (Assavapokee et al. (2008), Aissi et al. (2009)). A proof of convergence for this iterative approach for the class of robust convex problems under strict uncertainty has been given in Mutapcic & Boyd (2009) and stems from the proof for the original cutting plane method given in Kelley (1960). The applicability of the iterative approach and its computational competitiveness holds for a wide range of optimization problems, such as robust combinatorial optimization (see Aissi et al. (2009)), robust mixed-integer programming (Fischetti & Monaci (2012) and Bertsimas et al. (2016)) and robust convex programming (Mutapcic & Boyd (2009)).

In our research, we want to generalize this iterative approach. We investigate what happens if the robustification and/or pessimization problem in each step are not solved to optimality, but only approximately. We analyze these enhancements theoretically and experimentally and show that they improve the efficiency of the resulting iterative approach significantly if the robustification and/or pessimization problem is hard to solve. Related to our proposed solution approach there already exist methods (Calafiore & Campi (2005); Campi & Garatti (2008); Calafiore (2010)) to find feasible solutions to uncertain optimization problems by constraint randomization. In contrast to our proposed approaches the main concern of this stream of literature is to investigate the probability of a solution being feasible. In contrast,

our considered uncertain optimization problem bears its uncertainty in the objective function and the set of feasible solutions is assumed to be analytically given.

The remainder of the paper is structured as follows. In Section 2 we introduce the notation, repeat the basic scheme of cutting plane approaches for robust optimization and introduce the enhanced solution scheme. Section 3 provides the theoretical analysis of convergence, while Section 4 contains the experimental results. The paper is concluded in Section 5.

2. Iterative approaches for robust optimization

As usual in uncertain optimization (see, e.g., Ben-Tal et al. (2009)) we consider a family of parameterized optimization problems $(P(u) : u \in \mathcal{U})$ where $\mathcal{U} \subseteq \mathbb{R}^q$ is a given *uncertainty set* and each scenario $u \in \mathcal{U}$ defines an optimization problem

$$P(u) \qquad \min_{x \in X} g(x, u)$$

with a set $X \subseteq \mathbb{R}^n$ and an uncertain objective function $g : X \times \mathbb{R}^q \to \mathbb{R}$. Note that we assume the function g to be defined on $\mathbb{R}^q$ and not only on the scenarios $u \in \mathcal{U}$. As common in robust optimization let us assume that one specified *nominal scenario* (the most likely one, or the undisturbed one) $u_{\text{nom}} \in \mathcal{U}$ is given.

We are interested in finding a *robust* solution to $(P(u) : u \in \mathcal{U})$. There are many different definitions when to call a solution $x \in X$ robust for $(P(u) : u \in \mathcal{U})$. In this paper we consider the most prominent definition, namely the concept of strict robustness. A solution is called *(strictly) robust* (or *minmax robust*) if it is an optimal solution to its *robust counterpart*

$$\text{(RC)} \qquad g^* = \min_{x \in X} \sup_{u \in \mathcal{U}} g(x, u),$$

see Ben-Tal et al. (2009); Soyster (1973). We consider uncertainty only in the objective and not in the constraints.

2.1. Traditional cutting plane approach for robust optimization

In the following we describe and analyze a cutting plane approach to solve (RC), i.e., to find robust solutions for a given family of optimization problems $(P(u) : u \in \mathcal{U})$. Given a subset $\mathcal{U}' \subseteq \mathcal{U}$ we define a function $g_{\mathcal{U}'} : X \to \mathbb{R}$ by

$$g_{\mathcal{U}'}(x) := \sup_{u \in \mathcal{U}'} g(x, u).$$

We can now define the following two optimization problems:

- For $\mathcal{U}' \subseteq \mathcal{U}$ the *robustification problem*

$$\text{RC}(\mathcal{U}') \qquad g^*_{\mathcal{U}'} := \min_{x \in X} \sup_{u \in \mathcal{U}'} g(x, u)$$

  is given as a relaxed robust counterpart problem considering only a subset of scenarios.

- For $x \in X$ the *pessimization problem*

$$\text{Pes}(x) \qquad\qquad g_{\mathcal{U}}(x) := \sup_{u \in \mathcal{U}} g(x,u)$$

evaluates a solution $x \in X$ in its worst case.

Note that (RC) equals RC($\mathcal{U}$) and $g^*_{\mathcal{U}'} = \min_{x \in X} g_{\mathcal{U}'}(x)$. As we are interested in solving (RC), we define

$$g^* := g^*_{\mathcal{U}}$$

to be the objective value of an optimal solution to (RC).

In their basic version, see for example Mutapcic & Boyd (2009), iterative approaches for solving (RC) construct a sequence

$$\mathcal{U}_1 = \{u_{\text{nom}}\} \subseteq \mathcal{U}_2 \subseteq \mathcal{U}_3 \subseteq \ldots\ldots \qquad\qquad \subseteq \mathcal{U}$$

of nested scenario sets. In iteration k the algorithm finds an optimal solution x_k to the robustification problem RC($\mathcal{U}_k$). Then it determines a worst-case scenario u_k by solving the pessimization problem Pes(x_k). The scenario u_k is then added to $\mathcal{U}_k$ in order to obtain the new scenario set $\mathcal{U}_{k+1}$ for the next iteration, see Figure 1 as illustration.

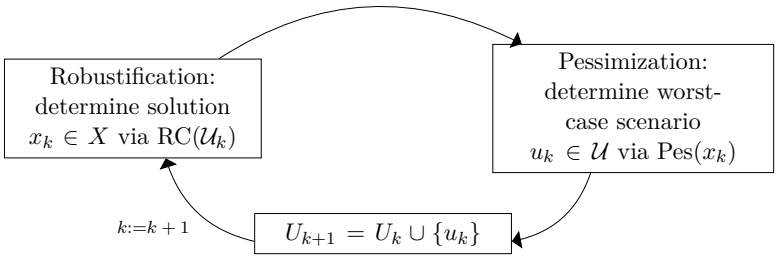


Figure 1: Basic Scheme of a cutting plane approach

We collect two elementary properties of this scheme.

Observation:

- For $\mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \mathcal{U}$ we have $g_{\mathcal{U}_1}(x) \leq g_{\mathcal{U}_2}(x) \leq g_{\mathcal{U}}(x)$ for all $x \in X$ and hence $g^*_{\mathcal{U}_1} \leq g^*_{\mathcal{U}_2} \leq g^*_{\mathcal{U}}$.
- Let $x_{\mathcal{U}'}$ be an optimal solution to RC($\mathcal{U}'$). Then g^* is bounded by

$$g^*_{\mathcal{U}'} = g_{\mathcal{U}'}(x_{\mathcal{U}'}) \leq g^* \leq g_{\mathcal{U}}(x_{\mathcal{U}'}). \tag{1}$$

Using (1) for $\mathcal{U}' = \mathcal{U}_k$, this gives bounds on g^* in each iteration k of the cutting plane approach: $g_{\mathcal{U}}(x_k)$ is an upper bound and $g_{\mathcal{U}_k}(x_k)$ is a lower bound on g^*.

Iterative approaches usually investigate ϵ-optimality to make the algorithm stop when the result is accurate enough since they come from the idea of cutting plane algorithms.

Notation: For a given optimization problem (RC) and some $\epsilon > 0$, define a solution $x \in X$ to be $\epsilon-$optimal with respect to (RC), if $g_{\mathcal{U}}(x) - g^* \leq \epsilon$ holds.

2.2. The approximate cutting plane approach for robust optimization

Now we are able to define our enhancements to the iterative approach described in Section 2.1. The idea is to avoid solving RC($\mathcal{U}_k$) and Pes(x_k) to optimality, but save computation time by solving these problems only approximately. As we will see, this can be done without losing solution quality; i.e., the approximate cutting plane algorithm still converges to an optimal solution under similar conditions as the traditional approach of the previous section. In order to formalize this enhancement we need to define approximate versions of (RC) and (Pes).

Definition 1. *For a given problem* RC($\mathcal{U}'$) *and some threshold* $t_{opt} \in \mathbb{R}$ *we define the approximate version of* RC($\mathcal{U}'$) *to return any solution* $x \in X$ *with* $g_{\mathcal{U}'}(x) \leq t_{opt}$ *or, if no such solution exists, an optimal solution to* RC($\mathcal{U}'$).
Similarly, for a given problem Pes(x) *and some threshold* $t_{pes} \in \mathbb{R}$ *we define the approximate version of* Pes(x) *to return any scenario* $u \in \mathcal{U}$ *with* $g(x,u) \geq t_{pes}$*, or, if no such scenario exists, a worst-case scenario to* x*, i.e., an optimal solution to* Pes(x).

A-RC($\mathcal{U}', t_{opt}$)	*Return some* $x \in X$ *with* $g_{\mathcal{U}'}(x) \leq t_{opt}$. *If such an* x *does not exist, return* $x \in X$ *optimal to* RC($\mathcal{U}'$)
A-Pes(x, t_{pes})	*Return some* $u \in \mathcal{U}$ *with* $g(x,u) \geq t_{pes}$. *If such a* u *does not exist, return* $u \in \mathcal{U}$ *optimal to* Pes(x).

This enhancement is motivated by the the following fact: If the robustification or pessimization is some mixed integer linear optimization problem, it is often very time-consuming for the MIP solver to close the last few percentage points of optimality gap. This gap-closing, however, may not be necessary since in some cases we do not need the exact solution for a certain subproblem or the exact worst-case scenario for a certain solution. Indeed, for many cases we only need some valid cut that discards the currently found solution in order to continue the iterative approach.

The choice of t_{opt} and t_{pes} determines how much computation time we need: The larger the threshold t_opt is in A-RC($\mathcal{U}', t_{opt}$), the more likely it is that we save computation time. For A-Pes(x, t_{pes}), the situation is reversed: The smaller the chosen threshold t_{pes}, the more computation time can be saved. Depending on t_{opt} and t_{pes} we can also force that A-RC($\mathcal{U}', t_{opt}$), or A-Pes(x, t_{pes}) are reduced to the exact problems:

Observation:

- Let lb be a lower bound on RC($\mathcal{U}'$), i.e., $lb \leq g^*_{\mathcal{U}'}$. Then for all $t_{opt} < lb$ we have that A-RC($\mathcal{U}', t_{opt}$) becomes RC($\mathcal{U}'$).
- Let ub be an upper bound on Pes(x), i.e., $ub \geq g_{\mathcal{U}}(x)$. Then for all $t_{pes} > ub$ we have that A-Pes(x, t_{pes}) becomes Pes(x).

We now formulate a version of the cutting plane algorithm, where both RC($\mathcal{U}_k$) and Pes(x_k) can be chosen to be solved approximately depending on the threshold parameters t_{opt} and t_{pes}. The algorithm is supposed to yield an ϵ-optimal solution $x_{ret} \in X$ to RC($\mathcal{U}$) and its pseudo-code is stated in Algorithm 1.

Algorithm 1: Approximate Cutting Plane Approach

Input: problem (RC), nominal scenario $u_{\text{nom}} \in \mathcal{U}$, stopping criterion $\epsilon > 0$
Output: ϵ-optimal solution $x_{ret} \in X$ to RC($\mathcal{U}$)

$U_1 \leftarrow \{u_{\text{nom}}\}$, $k \leftarrow 1$, $lb_1 \leftarrow -\infty$, $ub_1 \leftarrow \infty$
while $ub_k - lb_k > \epsilon$ **do**
  determine t_{opt_k}
  $x_k \leftarrow$ solution to A-RC(U_k, t_{opt_k})
  if A-RC(U_k, t_{opt_k}) *solved optimally and* $lb_k < g_{\mathcal{U}_k}(x_k)$ **then**
    $lb_{k+1} \leftarrow g_{\mathcal{U}_k}(x_k)$
  else
    $lb_{k+1} \leftarrow lb_k$
  determine t_{pes_k}
  $u_k \leftarrow$ solution to A-Pes(x_k, t_{pes_k})
  if A-Pes(x_k, t_{pes_k}) *solved optimally and* $ub_k > g(x_k, u_k)$ **then**
    $ub_{k+1} \leftarrow g(x_k, u_k)$
    $x_{ret} \leftarrow x_k$
  else
    $ub_{k+1} \leftarrow ub_k$
  $U_{k+1} \leftarrow U_k \cup \{u_k\}$
  $k \leftarrow k + 1$
end
return x_{ret}

The specific values for the threshold parameters $t_{opt_k}, t_{pes_k} \in \mathbb{R}$ are determined during the execution of the algorithm. We call the resulting sequences (t_{opt}) and (t_{pes}) *threshold sequences*. For the definition of the algorithm the threshold values could be chosen arbitrarily, but for the convergence proofs we will assume bounds on them. In the section of computational experiments choices for the threshold values will be investigated.

To ensure tractability of the algorithm, we make the following assumption.

Assumption: We assume that A-RC($\mathcal{U}_k, t_{opt_k}$) for finite $|\mathcal{U}_k|$ and A-Pes(x_k, t_{pes_k}) are solvable in a finite number of steps and their solutions are always finite.

With this assumption in mind we now investigate when Algorithm 1 converges.

3. Analysis of the approximate cutting plane approach

In this section we first give some lemmas that will be helpful to show correctness of Algorithm 1. We then prove its correctness in the case that the uncertainty set $\mathcal{U}$ is finite and then cover the case of a bounded uncertainty set $\mathcal{U}$ with possibly infinitely many scenarios. After this we analyze how the sequences t_{pes} and t_{opt} can be chosen and finally give some strategies for further speedup of the algorithm.

First, some basic properties of Algorithm 1 are collected.

Observation: For all $k \in \mathbb{N}$ it holds $lb_k \leq lb_{k+1}$, $ub_k \geq ub_{k+1}$, and $lb_k \leq g^* \leq ub_k$.

The next result shows the following: If the algorithm stops, it does so with an exact solution (up to ϵ).

Lemma 2. *If Algorithm 1 returns a solution $x_{ret} \in X$, it is an $\epsilon-$optimal solution with respect to (RC).*

Proof. Let the algorithm stop after n iterations. It hence returns a solution x_{ret} with $g_{\mathcal{U}}(x_{ret}) = ub_n$ and moreover $ub_n - lb_n \leq \epsilon$.
The lower bound lb_n at iteration n has been found in some iteration, say, $i \leq n$, i.e., $lb_n = lb_i = \min_{x \in X} g_{\mathcal{U}_i}(x)$. We get

$$ub_n - \epsilon \leq lb_n = lb_i = \min_{x \in X} g_{\mathcal{U}_i}(x) \leq \min_{x \in X} g_{\mathcal{U}}(x) = g^*$$

and hence $g_{\mathcal{U}}(x_{ret}) - g^* = ub_n - g^* \leq \epsilon$ i.e., x_{ret} is $\epsilon-$optimal. □

Next, we investigate under which conditions Algorithm 1 stops.

3.1. Convergence for finite uncertainty sets

Convergence proofs for the iterative approach in the case of a finite uncertainty set $\mathcal{U}$ are often omitted in the literature (e.g., in Aissi et al. (2009)) since it can be seen as a special case of Kelley's Cutting Plane Method (Kelley (1960)). For our proposed enhancements, however, the convergence proofs are not straightforward and we hence investigate the termination of Algorithm 1 if t_{opt} and t_{pes} are chosen dynamically during the iterations. We discuss the following different choices.

Lemma 3. *If $t_{opt_k} < ub_k$ and $t_{pes_k} > g_{\mathcal{U}_k}(x_k)$ for all $k \in \mathbb{N}$ the number of iterations for Algorithm 1 is bounded by $|X| + |\mathcal{U}|$.*

Proof. We show that in each iteration of the algorithm either a new scenario $u_k \notin \mathcal{U}_k = \{u_1, \ldots, u_{k-1}\}$ is generated or that if the current solution x_k occurs as a solution in a later iteration of the algorithm, the algorithm will terminate. Hence, the algorithm ends after at most $|X| + |\mathcal{U}|$ iterations.

In each iteration of Algorithm 1 the problem A-Pes(x_k, t_{pes_k}) is either solved optimally or approximately.

- If A-Pes(x_k, t_{pes_k}) is solved approximately, it returns u_k with $g(x_k, u_k) \geq t_{pes_k} > g_{\mathcal{U}_k}(x_k) = \max_{u \in \mathcal{U}_k} g(x_k, u)$. Hence, $u_k \notin \mathcal{U}_k$.
- If A-Pes(x_k, t_{pes_k}) is solved optimally, then we know that the objective value $g(x_k, u_k)$ either imposes a new upper bound (line 12 of algorithm 1), or the current upper bound is already equal or better than $g(x_k, u_k)$ (line 15). In either case we get $ub_{k+1} \leq g(x_k, u_k)$.

  For every $n > k$ it hence (and because of $u_k \in \mathcal{U}_n$) holds that $g_{\mathcal{U}_n}(x_k) \geq g(x_k, u_k) \geq ub_{k+1} \geq ub_n$. We now show that for every newly obtained solution x_n (with $n > k$) the converse, i.e., $g_{\mathcal{U}_n}(x_n) < ub_n$, is true, or the algorithm terminates.

  Let $n > k$. For showing $g_{\mathcal{U}_n}(x_n) < ub_n$ we have to distinguish two cases: A-RC$(\mathcal{U}_n, t_{opt_n})$ being solved approximately and solved optimally. For an approximate solution of A-RC$(\mathcal{U}_n, t_{opt_n})$ we get $g_{\mathcal{U}_n}(x_n) \leq t_{opt_n}$ by definition of A-RC and $t_{opt_n} < ub_n$ by assumption of the lemma. For an optimal solution x_n it follows that $g_{\mathcal{U}_n}(x_n) = \min_{x \in X} g_{\mathcal{U}_n}(x) \leq \min_{x \in X} g_{\mathcal{U}}(x) \leq ub_n$. The case $g_{\mathcal{U}_n}(x_n) = ub_n$ would imply that the algorithm terminates since $lb_{n+1} \geq g_{\mathcal{U}_n}(x_n)$ (by line 6 of the algorithm) holds, which implies $ub_{n+1} \leq ub_n = lb_{n+1}$, leading to the termination of Algorithm 1. Thus for A-RC$(\mathcal{U}_n, t_{opt_n})$ being solved optimally it either holds that $g_{\mathcal{U}_n}(x_n) < ub_n$ or that the algorithm terminates.

  Therefore we have on the one hand that $g_{\mathcal{U}_n}(x_k) \geq ub_n$ and on the other hand that $g_{\mathcal{U}_n}(x_n) < ub_n$ if the algorithm does not terminate in iteration n. This means that $x_n \neq x_k$ for all $n > k$.

□

Lemma 4. *If $t_{opt_k} \leq lb_k$ and $t_{pes_k} > g_{\mathcal{U}_k}(x_k)$ for all $k \in \mathbb{N}$ the number of iterations for Algorithm 1 is bounded by $|\mathcal{U}|$.*

Proof. First note that A-RC$(\mathcal{U}_k, t_{opt_k})$ returns an optimal solution in each iteration k since $t_{opt_k} \leq lb_k$. We now show that the algorithm stops if in iteration k some $u_k \in \{u_0, \ldots, u_{k-1}\} = \mathcal{U}_k$ is generated. This bounds the number of iterations by $|\mathcal{U}|$.

Let u_n in some iteration n be the solution of problem A-Pes(x_n, t_{pes_n}). We distinguish two cases: If A-Pes(x_n, t_{pes_n}) is solved approximately, we have that $g(x_n, u_n) \geq t_{pes_n} > g_{\mathcal{U}_n}(x_n) = \max_{u \in \mathcal{U}_n} g(x_n, u)$, hence $u_n \notin \mathcal{U}_n$. Now assume that A-Pes(x_n, t_{pes_n}) is solved optimally and returns a solution u_n with $u_n \in \mathcal{U}_n$, i.e., $\max_{u \in \mathcal{U}} g(x_n, u) = \max_{u \in \mathcal{U}_n} g(x_n, u) = g(x_n, u_n)$. Then it holds that

$$lb_{n+1} \underbrace{\geq}_{\text{line 6 of Algorithm 1}} g_{\mathcal{U}_n}(x_n) = g_{\mathcal{U}}(x_n) = g(x_n, u_n) \underbrace{\geq}_{\text{line 12 of Algorithm 1}} ub_{n+1},$$

hence $ub_{n+1} - lb_{n+1} < \epsilon$ and Algorithm 1 terminates (line 2), i.e., in the case that A-Pes is solved optimally either the algorithm terminates or $u_n \notin \mathcal{U}_n$.

Overall we get in both cases that $u_k \notin \mathcal{U}_k$ or termination of the algorithm, which bounds the maximal number of iterations by $|\mathcal{U}|$. □

Lemma 5. *If $t_{opt_k} < ub_k$ and $t_{pes_k} \geq ub_k$ the number of iterations for Algorithm 1 is bounded by $|X|$.*

Proof. We show that in every iteration n of the algorithm the solution x_n has two properties: First, $g(x_n, u_n) \geq ub_{n+1}$ holds, and second, either $g_{\mathcal{U}_n}(x_n) < ub_n$ or $g_{\mathcal{U}_n}(x_n) = lb_n$ holds. In the third step, we show that $x_n = x_k$ for some $n > k$ yields either a contradiction or termination of the algorithm. This implies that the maximal number of iterations is bounded by $|X|$.

Showing the first property: The assumption $t_{pes_k} \geq ub_k$ implies that in every iteration n of the algorithm the new scenario u_n, which is found by solving A-Pes(x_n, t_{pes_n}), satisfies either $g(x_n, u_n) \geq t_{pes_n} \geq ub_n \geq ub_{n+1}$ (if A-Pes is solved approximately) or $g(x_n, u_n) \geq ub_{n+1}$ directly (if A-Pes is solved optimally). Thus $g(x_n, u_n) \geq ub_{n+1}$ holds in every iteration n.

Showing the second property: Assume that A-RC$(\mathcal{U}_n, t_{opt_n})$ finds an optimal solution x_n that hence minimizes $g_{\mathcal{U}_n}(x)$, which means that $g_{\mathcal{U}_n}(x_n) = lb_n \leq ub_n$. If A-RC$(\mathcal{U}_n, t_{opt_n})$ finds an approximate solution, it holds that $g_{\mathcal{U}_n}(x_n) \leq t_{opt_n} < ub_n$. Thus $g_{\mathcal{U}_n}(x_n) < ub_n$ or $g_{\mathcal{U}_n}(x_n) = lb_n$ holds in every iteration n.

Let now x_n be a solution returned in iteration n. If $x_n = x_k$ for some $n > k$, then we get $g(x_n, u_k) = g(x_k, u_k) \geq ub_{k+1} \geq ub_n$ implying $g_{\mathcal{U}_n}(x_n) \geq ub_n$. Now, by the second property, there exists either a contradiction to $g_{\mathcal{U}_n}(x_n) < ub_n$ from above, or termination of the algorithm since $lb_n = g_{\mathcal{U}_n}(x_n) \geq ub_n$.
Hence, $x_n \neq x_k$ for all $k < n$ and thus in every iteration a new solution x_n is found, bounding the number of iterations by $|X|$. □

From the previous two lemmas we obtain the following corollary.

Corollary 6. *If $t_{opt_k} \leq lb_k$ and $t_{pes_k} \geq ub_k$ the number of iterations for Algorithm 1 is bounded by* $\min(|X|, |\mathcal{U}|)$. *This contains the case of* A-RC *and* A-Pes *always being solved exactly.*

Proof. First, the assumptions of Lemma 5 are satisfied. This means, the number of iterations is bounded by $|X|$ and $g_{\mathcal{U}_n}(x_n) \leq ub_k$ which was shown as second property in the proof of Lemma 5. The latter guarantees that also the assumptions of Lemma 4 hold, hence the number of iterations is also bounded by $|\mathcal{U}|$. The result follows. □

Going from A-RC and A-Pes solved exactly to both solved approximately increases the maximal number of iterations from $min(|\mathcal{U}|, |X|)$ to $|\mathcal{U}|+|X|$. In the following section these bounds will not help us as both $\mathcal{U}$ and X will be assumed to possibly have infinite cardinality.

3.2. Convergence for infinite uncertainty sets

The convergence of the iterative approach for A-RC and A-Pes being solved exactly is, depending on the assumptions of X and $\mathcal{U}$, a commonly known result, see Mutapcic & Boyd (2009). The following theorem generalizes the convergence result of this paper to the case of A-RC and A-Pes being solved exactly or approximately.

Definition 7. *Given two sets $X, \mathcal{U} \subseteq \mathbb{R}$ and a function $g : X \times \mathcal{U} \to \mathbb{R}$ we define g to be uniformly Lipschitz-continuous in x with constant L if for all $u \in \mathcal{U}$ it holds*

$$|g(x, u) - g(y, u)| \leq L\|x - y\| \quad \forall x, y \in X.$$

Theorem 8. *Let X be bounded and g be uniformly Lipschitz-continuous in x with constant L. Furthermore let $t_{opt_k} \leq ub_k - \epsilon$ and $t_{pes_k} \geq ub_k$. Then Algorithm 1 converges in a finite number of steps.*

Proof. Assume Algorithm 1 does not converge. Let x_l and x_k with $l < k$ be the two solutions in iteration l and k of Algorithm 1. We know by definition of Lipschitz continuity that

$$|g(x_l, u_l) - g(x_k, u_l)| \leq L\|x_k - x_l\|$$

holds. We will show that we can bound $g(x_l, u_l) \geq ub_k$ and $g(x_k, u_l) < ub_k - \epsilon$.

Bounding $g(x_l, u_l)$: By assumption on t_{pes} it holds that $g(x_l, u_l) \geq ub_{l+1}$. This is the case because if A-Pes(x_l, t_{pes_l}) is solved approximately, then $g(x_l, u_l) \geq t_{pes_l} \geq ub_l = ub_{l+1}$ holds. If A-Pes(x_l, t_{pes_l}) is solved optimally, either $g(x_l, u_l) \geq ub_l \geq ub_{l+1}$ (line 15 of Algorithm 1), or $ub_{l+1} \leftarrow g(x_l, u_l)$ (line 12 of Algorithm 1). Overall we get $g(x_l, u_l) \geq ub_{l+1} \geq ub_k$.

Bounding $g(x_k, u_l)$: Note that $u_l \in \mathcal{U}_k$. If RC$(\mathcal{U}_k, t_{opt_k})$ is solved approximately, we get $g(x_k, u_l) \leq g_{\mathcal{U}_k}(x_k) \leq t_{opt_k} \leq ub_k - \epsilon$. If RC$(\mathcal{U}_k, t_{opt_k})$ is solved optimally we know $g(x_k, u_l) \leq g_{\mathcal{U}_k}(x_k) = lb_{k+1}$. Furthermore, $lb_{k+1} \leq ub_k - \epsilon$, otherwise the algorithm would terminate (line 2 of Algorithm 1) and we are done.

Thus we get on the one hand that $g(x_l, u_l) \geq ub_k$ and on the other hand that $g(x_k, u_l) \leq ub_k - \epsilon$. Hence,

$$\epsilon = |ub_k - (ub_k - \epsilon)| < |g(x_l, u_l) - g(x_k, u_l)| \leq L\|x_k - x_l\|.$$

With this insight we know that for all x_k with $k \in \mathbb{N}$ it holds that $\|x_k - x_l\| \geq \frac{\epsilon}{L}$ for all $l < k$. Thus every solution x_k has a minimum distance to all other solutions within X. Since X is bounded, there can only be a finite number of x_k contained X, and Algorithm 1 has to terminate eventually. □

We remark that uniform Lipschitz continuity is a strong assumption if $\mathcal{U}$ is unbounded. In this case, even the scalar bilinear function $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, (x, u) \mapsto x \cdot u$ can not be uniformly Lipschitz continuous. Hence, for unbounded $\mathcal{U}$ uniformly Lipschitz continuity covers only a very restrictive (i.e., not linear, not convex, not polynomial) class of functions g. Nevertheless, this case is interesting from a theoretical point of view, see Mutapcic & Boyd (2009).

In order to get rid of the boundedness assumption of X, we can assume $\mathcal{U}$ to be bounded and with a slight modification on the assumptions for the threshold sequences we will also get a convergence result.

Theorem 9. *Let $\mathcal{U}$ be bounded and g be uniformly Lipschitz-continuous in u with constant L'. Let $t_{opt_k} \leq lb_k$ and $t_{pes_k} \geq g_{\mathcal{U}_k}(x_k) + \epsilon$. Then Algorithm 1 converges in a finite number of steps.*

Proof. Assume that Algorithm 1 does not converge. Let u_l and u_k with $l < k$ be the two scenarios found in iteration l and k of Algorithm 1. We know by definition of Lipschitz continuity that

$$|g(x_k, u_k) - g(x_k, u_l)| \leq L'\|u_k - u_l\|$$

holds for all solutions $x \in X$ and especially for x_k, the solution found in iteration k. We will show that we can bound $|g(x_k, u_k) - g(x_k, u_l)| \geq \epsilon$.

Assume that A-Pes(x_k, t_{pes_k}) has been solved approximately. Hence

$$g(x_k, u_k) \geq t_{pes_k} \geq g_{\mathcal{U}_k}(x_k) + \epsilon \geq g(x_k, u_k) + \epsilon$$

and we are done. If A-Pes(x_k, t_{pes_k}) has been solved to optimality, then we know (by lines 12 and 15 of the algorithm) that $g(x_k, u_k) \geq ub_{k+1}$. For $g(x_k, u_l)$ we will look at A-RC$(\mathcal{U}_k, t_{opt_k})$. Since the threshold value t_{opt_k} is always smaller or equal lb_k, the robustification is always solved exactly and we get

$$g(x_k, u_l) \leq lb_k \leq lb_{k+1}.$$

If it would be the case that $ub_{k+1} - lb_{k+1} < \epsilon$, then the algorithm would terminate (hence after a finite number of steps). Otherwise we get

$$g(x_k, u_l) \leq lb_k \leq lb_{k+1} \leq ub_{k+1} - \epsilon.$$

Thus it follows that

$$|g(x_k, u_k) - g(x_k, u_l)| \geq |ub_{k+1} - (ub_{k+1} - \epsilon)| = \epsilon.$$

With this insight we know that for all u_k with $k \in \mathbb{N}$ it holds that $\|u_k - u_l\| \geq \frac{\epsilon}{L'}$ for all $l < k$. Thus every scenario u_k has a minimum distance to all other scenarios within $\mathcal{U}$. Since $\mathcal{U}$ is bounded, there can only be a finite number of u_k contained $\mathcal{U}$, and Algorithm 1 has to terminate eventually.

□

The next theorem allows the more general assumption on the threshold sequences, but in order to ensure convergence we now have to assume X and $\mathcal{U}$ to be bounded.

Theorem 10. *Let X and $\mathcal{U}$ be bounded and g be uniformly Lipschitz-continuous in x with constant L. Further let g be uniformly Lipschitz-continuous in u with constant L'. Finally let $t_{opt_k} \leq ub_k - \epsilon$ and $t_{pes_k} \geq g_{\mathcal{U}_k}(x_k) + \epsilon$. Then Algorithm 1 converges in a finite number of steps.*

Proof. Consider the case that the algorithm does not converge. Then either the number of A-Pes(x_k, t_{pes_k}) solved optimally or the number of A-Pes$(x_k.t_{pes_k})$ solved approximately is infinitely large. Let $I \subseteq \mathbb{N}$ denote the indices of all iterations belonging to the respective case.

Case 1, A-Pes(x_k, t_{pes_k}) solved infinitely often to optimality: For $l \in I$ we know that $g(x_l, u_l) \geq ub_{l+1}$. But for arbitrary $k \in I$ with $l < k$ we know that $g(x_k, u_l) \leq g_{\mathcal{U}_k}(x_k)$ since $u_l \in \mathcal{U}_k$. If in iteration k the problem A-RC$(\mathcal{U}_k, t_{opt_k})$ was solved to optimality, we get

$$g_{\mathcal{U}_k}(x_k) \leq lb_{k+1} \underbrace{\leq}_{\text{otherwise termination}} ub_{k+1} - \epsilon \leq ub_{l+1} - \epsilon.$$

If A-RC$(\mathcal{U}_k, t_{opt_k})$ was solved approximately we get

$$g_{\mathcal{U}_k}(x_k) \leq t_{opt_k} \leq ub_k - \epsilon \leq ub_{l+1} - \epsilon.$$

Together, we get $g(x_k, u_l) \leq g_{\mathcal{U}_k}(x_k) \leq ub_{l+1} - \epsilon$. Hence,

$$\epsilon = |ub_{l+1} - (ub_{l+1} - \epsilon)| \leq |g(x_l, u_l) - g(x_k, u_l)| \leq L\|x_l - x_k\|.$$

So all pairs x_k, x_l have a minimum distance of $\frac{\epsilon}{L}$ to each other (in X). Since we have infinitely many x_k with $k \in I$, we get a contradiction due to the boundedness of X.

Case 2, A-Pes(x_k, t_{pes_k}) solved infinitely often approximately: In this case we know that $g(x_k, u_k) \geq g_{\mathcal{U}_k}(x_k) + \epsilon$ for every x_k with $k \in I$. So for any u_l with $l \in I, l < k$ it holds that $g_{\mathcal{U}_k}(x_k) = \max_{u \in \mathcal{U}_k} g(x_k, u) \geq g(x_k, u_l)$, since $u_l \in \mathcal{U}_k$. Thus we get

$$\epsilon = |g_{\mathcal{U}_k}(x_k) + \epsilon - g_{\mathcal{U}_k}(x_k)| \leq |g(x_k, u_k) - g(x_k, u_l)| \leq L'\|u_k - u_l\|.$$

Now we know (like above with the x_k) that every pair of u_l, u_k with $k, l \in I$, $k \neq l$ has a minimum distance of $\frac{\epsilon}{L'}$ to each other. Since we have infinitely many of these, we get a contradiction to the boundedness of $\mathcal{U}$.

Thus both cases can only occur a finite number of times, hence Algorithm 1 converges. □

From these convergence results, we see that it is important to know how t_{opt} and t_{pes} can be chosen. This is further explained in the next subsection.

3.3. Choice of threshold sequences

Now that it is shown under which assumptions Algorithm 1 converges, we give a motivation on how we will choose the threshold sequences later in our experiments.

For t_{opt} it is on the one required that $t_{opt_k} < ub_k$. This indeed makes sense as otherwise we would look for a solution whose objective value might be worse than the currently best found solution. On the other hand, if t_{opt_k} is chosen to be smaller or equal lb_k, the problem is solved exactly (since there does not exist a solution with objective smaller than lb_k). This leads to the recommendation of

$$t_{opt_k} \in [lb_k, ub_k - \epsilon].$$

For t_{pes_k} it was shown that to ensure convergence of Algorithm 1 it has to hold that $t_{pes_k} \geq g_{\mathcal{U}_k}(x_k) + \epsilon$. We do not have an upper bound in $\mathbb{R}$ that guarantees to solve A-Pes(x_k, t_{pes_k}) to optimality. Nevertheless, if $t_{pes_k} \geq ub_k$ we know for the currently considered solution x_k if its objective value $g_{\mathcal{U}}(x_k)$ is worse or better than the current best solution. Hence

$$t_{pes_k} \in [g_{\mathcal{U}_k}(x_k), \infty).$$

In order to reduce the complexity of choosing the threshold values we will choose the values by

$$\begin{aligned} t_{opt_k} &= lb_k + t_{opt}(ub_k - lb_k) \\ t_{pes_k} &= ub_k + t_{pes}(g_{\mathcal{U}_k}(x_k) - ub_k), \end{aligned}$$

where we vary the parameters $t_{opt_k} \in [0, 1)$ and $t_{pes} \in (-\infty, 1)$.
The impact of different choices for t_{opt} and t_{pes} will be investigated in a computational study in the next chapter and we will see that a choice of $t_{pes} < 0$ does not yield better results than $t_{pes} = 0$ and hence $t_{pes} \in [0, 1)$ is a sufficient choice.

3.4. Optimization strategies

Algorithm 1 works for a wide class of problems. In our computational study we restrict ourselves to the case of X and $\mathcal{U}$ being polyhedral as well as g linear in x and u. This has the consequence that A-RC and A-Pes are mixed-integer programs. For this special case we propose three enhancements for Algorithm 1 speeding up the solution process.

Warm start for robustification and pessimization:

- For solving A-RC($\mathcal{U}_k, t_{opt_k}$), we can provide the currently best known solution x_{ret} as a starting solution.
- For solving A-Pes(x_k, t_{pes_k}) we can give $\operatorname{argmax}_{u \in \mathcal{U}_k} g(x_k, u)$ as a starting solution.

This makes in particular sense, if a MIP solver is used to solve the robustification and the pessimization problems which is the case in the setting mentioned above.

Strengthening bounds: If A-RC and A-Pes are solved approximately and a MIP solver is used, then in each iteration of Algorithm 1 we get a lower bound (in the robustification step) and an upper bound (in the pessimization step) that is found by the solver. (This is also the case when RC and Pes are solved exactly, but then we have a new solution which coincides with this bound.) These bounds can also be considered and might strengthen the current bounds on the problem in some cases. Especially when the threshold values t_{pes_k} and t_{opt_k} are chosen dependent on the current bounds, this improvement might effect the runtime of the algorithm.

Branch-and-Cut Framework: If A-RC is a mixed integer program, that is solved by branch-and-bound, Algorithm 1 can be implemented as a branch-and-cut algorithm. This means that we can make use of the callback function of a MIP solver in order to run the pessimization step for some obtained solution x_k and to add some scenario u_k to the set of considered solutions (corresponds to adding a new constraint to the MIP). This, on the one hand, speeds up the solution process of the robustification step since we do not have to start solving a new problem in every iteration. On the other hand this means that we have no choice but to apply the pessimization step for every feasible solution that the solver finds. This is a necessity since a MIP solver working with a branch-and-bound algorithm has to know for every node it encounters if the obtained solution can be discarded or considered feasible.

For the iterative approach with A-RC and A-Pes solved exactly the warm-start and branch-and-cut techniques have been used in the literature before, see e.g., Pérez-Galarce et al. (2014). The technique of strengthening the bound makes only sense for the approximate cutting plane approach and is hence a novelty. Nevertheless, an investigation for all three speed-up-techniques is provided in the following chapter.

4. Computational experiments

To conduct our computational studies we restrict ourselves to solve a class of robust mixed-integer linear optimization problems with mixed-integer polyhedral uncertainty. This means we consider the problem

$$\begin{aligned} P(u) \qquad \min \quad & u^t x \\ \text{s.t.} \quad & Ax \geq b, \\ & x \in \mathbb{R}^{n-p} \times \mathbb{Z}^p, \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ and for some scenario $u \in \mathcal{U} := \{u \in \mathbb{R}^{n-q} \times \mathbb{Z}^q | Cu \leq d\}$ with $C \in \mathbb{R}^{m \times n}, d \in \mathbb{R}^m$.

The convergence of Algorithm 1 when applied to this class of problems is guaranteed by Theorem 10 since $X = \{x \in \mathbb{R}^{n-p} \times \mathbb{Z}^p | Ax \geq b\}$ and $\mathcal{U}$ will be generated as compact sets and the objective function $u^t x$ is continuous and together with the compactness uniformly Lipschitz-continuous in x and u.

The robustification problem for some finite set $\mathcal{U}'$ can be formulated as a mixed integer program, i.e.,

$$\begin{aligned} \mathrm{RC}(\mathcal{U}') \qquad \min \quad & \tau \\ \text{s.t.} \quad & Ax \geq b \\ & u^t x \leq \tau \quad \forall u \in \mathcal{U}', \\ & x \in \mathbb{R}^{n-p} \times \mathbb{Z}^p, \tau \in \mathbb{R}, \end{aligned}$$

and the pessimization step can also be modeled as a mixed integer program:

$$\begin{aligned} \mathrm{Pes}(x) \qquad \max \quad & u^t x \\ \text{s.t.} \quad & Cu \leq d, \\ & u \in \mathbb{R}^{n-q} \times \mathbb{Z}^q. \end{aligned}$$

Generating X: We generate the coefficients for every row a_i of the matrix A to be (uniform) randomly chosen from the interval $[0, 100]$. Additionally b_i is chosen to be $10n$. Furthermore we restrict every x_i to lie in the interval $[0, 10]$ in order to ensure compactness of X.

Generating $\mathcal{U}$: The uncertainty set $\mathcal{U}$ is generated similarly. That is, every coefficient of the matrix $C \in \mathbb{R}^{m \times n}$ is chosen randomly from the interval $[0, 100]$ and all d_i are set to be $10n$. We also restrict the $u \in \mathcal{U}$ to lie in the hyper-box $[0, 10]^n$. Note that we bound the constraints $C_i u \leq d_i$ from above, whereas $A_i x \geq b_i$ bounds the x-variables from below. This is done in order to create difficult constraints, since pessimization is a maximization problem, whereas robustification is a minimization problem.

Generating easy and hard instances: For both X and $\mathcal{U}$ we make two different parameter choices: Either $p = n$ or $p = 1$, corresponding that all variables need to be integer or that only one variable needs to be integer. Hence we say that an instance has a hard robustification problem if $X \subseteq \mathbb{Z}^n$ and easy if $X \subseteq \mathbb{R}^{n-1} \times \mathbb{Z}$. Analogously, we speak of an instance with hard pessimization problem if $\mathcal{U} \subseteq \mathbb{Z}^n$ and easy if $\mathcal{U} \subseteq \mathbb{Z}^{n-1} \times \mathbb{R}$. We hence receive four different sets of instances:

- Both, robustification and pessimization are easy,
- robustification is hard (and pessimization easy),
- pessimization is hard (and robustification easy), and
- both, robustification and pessimization are hard.

As we will see, this distinction between easy and hard highly impacts the complexity of robustification and pessimization and therefore the hardness of A-RC/A-Pes.

We implemented Algorithm 1 using Python 3.6 and Gurobi 7.5.2 as IP solver (with default settings). The implementation was tested on a compute server (78GB RAM, Intel(R) Xeon(R) CPU X5650 @ 2.67GHz with four used cores). For every instance we set an overall time limit of 10 minutes.

4.1. Comparing solving exactly vs. approximately

In this first comparison we investigate if and how much replacing RC by A-RC and Pes by A-Pes speed up the iterative approach. Therefore we define four different variants of Algorithm 1.

- app-app: Both, robustification and pessimization are solved approximately with $t_{opt} = 0.5$ and $t_{pes} = 0.5$.
- ex-app: Robustification is solved exactly, pessimization approximately with $t_{opt} = 0$ and $t_{pes} = 0.5$.
- app-ex: Robustification is solved approximately, pessimization exactly with $t_{opt} = 0.5$ and $t_{pes} = -\infty$.
- ex-ex: Both, robustification and pessimization are solved exactly, i.e., $t_{opt} = 0$ and $t_{pes} = -\infty$.

Thus ex-ex corresponds to the traditional cutting plane approach of Section 2.1 and the three other variants are modifications as described in Section 2.2.

We now run all four algorithms for the four different instance sets: Both easy, robustification hard (and pessimization easy), pessimization hard (and robustification easy) and both hard.

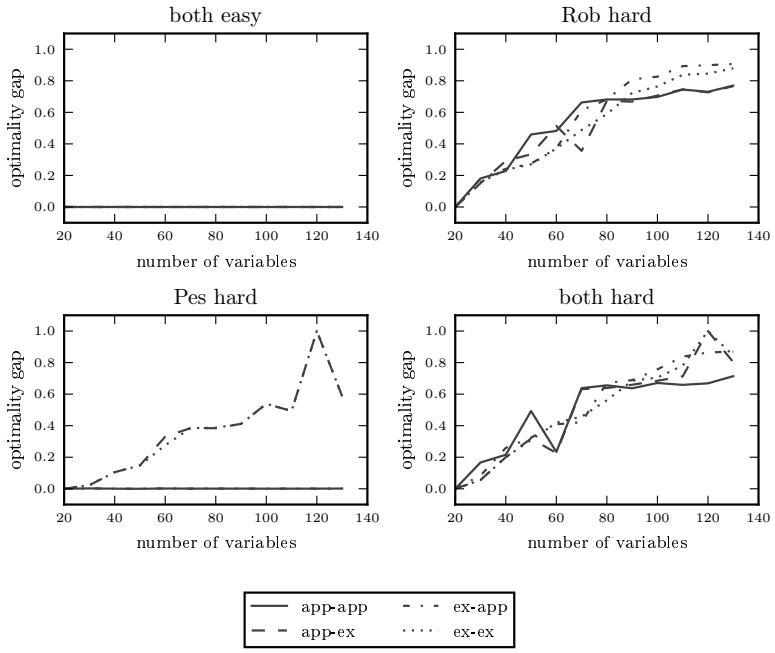


Figure 2: Comparison Exact vs. Approximately

In the first test we let the number of variables n vary between 20 and 130 and investigate the optimality gap, i.e., $\frac{ub-lb}{ub}$, obtained after hitting the time limit of 10 minutes.
For the easiest case of both robustification and pessimization easy, all algorithms find the optimal objective value within the time limit. However, when the robustification problem is hard, app-ex as well as app-app return solutions with a smaller gap than ex-app and ex-ex. If, on the other hand, the pessimization problem is difficult, the procedures app-app and ex-app have zero gap, wheres app-ex and ex-ex behave very similarly and have a large gap. If both problems are chosen to be difficult, then app-app returns the best solutions in many cases, especially if the number of variables is increasing (see Figure 2).

These results can be explained as follows: If one of robustification and pessimization is difficult and the other one is easy, solving the difficult one of them is the bottleneck. If this bottleneck problem is solved to optimality in each iteration, we might save some iterations, but we lose in terms of computation time. Therefore if the robustification problem is hard, it is preferable to use A-RC, i.e., app-ex or app-app. On the other hand, if the pessimization problem is hard it is preferable to use A-Pes, i.e., ex-app or app-app. Combining these two results suggests to use app-app if both problems are hard.

These results do not only hold for a time limit of 10 minutes, but also for the setting when

140

we let the algorithm solve instances to optimality. In Table 1 we show runtimes for this case.

	Both easy	RC hard	Pes hard	Both hard
App-App	0.482	44.224	379.172	14.922
App-Ex	0.48	45.01	928.298	17.278
Ex-App	0.474	100.574	379.122	21.09
Ex-Ex	0.474	98.102	929.43	21.2

Table 1: Average runtime in seconds for solving five instances with 20 variables to optimality.

Similar results are observable. If the robustification part is hard, then solving it only approximately is promising. Analogously, if the pessimization part is hard, it should be solved approximately. Overall we can say that Algorithm 1 pays off if the difficult problems are solved approximately.

4.2. Optimization strategies

We start by investigating two of the three improvements proposed in Section 3.4: Warm start and strengthening bounds. For all four different sets of instances we run four different versions of our algorithm.

- no improvements: Default version of the algorithm.
- only bounds: We update the lower bound lb_k with the lower bound found by the solver when solving A-RC and the upper bound ub_k by the upper bound from the solver when solving A-Pes.
- only warm start: When solving A-RC we provide x_{ret}, i.e., the currently best found solution, as a start solution for the robustification and when solving A-Pes we provide $\text{argmax}_{u \in \mathcal{U}_k} g(x_k, u)$, i.e., the currently worst scenario, as a start solution for the pessimization step.
- both improvements: we switch on both of the improvements.

	Both easy	RC hard	Pes hard	Both hard
both improvements	2.54	208.11	2461.19	75.41
only warm start	2.36	202.91	2500.05	74.15
only bounds	2.22	217.21	2465.08	112.8
no improvements	2.27	237.64	2658.07	106.51

Table 2: Average runtime in seconds for solving five instances with 30 variables to optimality.

We can see in Table 2 that on all instances both improvements tend to have shorter computation times, but only sometimes (for Both hard) there is a big impact on the overall computational time.

4.3. Comparison against branch-and-cut

Now we come to the third suggested improvement: The branch-and-cut framework. Based on our previous results we propose also here an approximate version of branch-and-cut in which we determine the worst-case scenario only approximately instead of solving the worst-case scenario exactly for each encountered solution as it is done in the standard branch-and-cut approaches. We hence give two versions of the branch-and-cut algorithm:

- BnC-ex is the standard case in which the worst-case scenario is determined exactly (i.e., Pes is solved for each encountered solution).
- In contrast, in the approximate version BnC-app we solve the pessimization approximately by using A-Pes with $t_{pes} = 0.5$.

We run these two algorithms on the four different classes of instances (both hard, optimization hard, pessimization hard, both easy) as in Section 4.1. For comparison we also show the solutions of app-app.

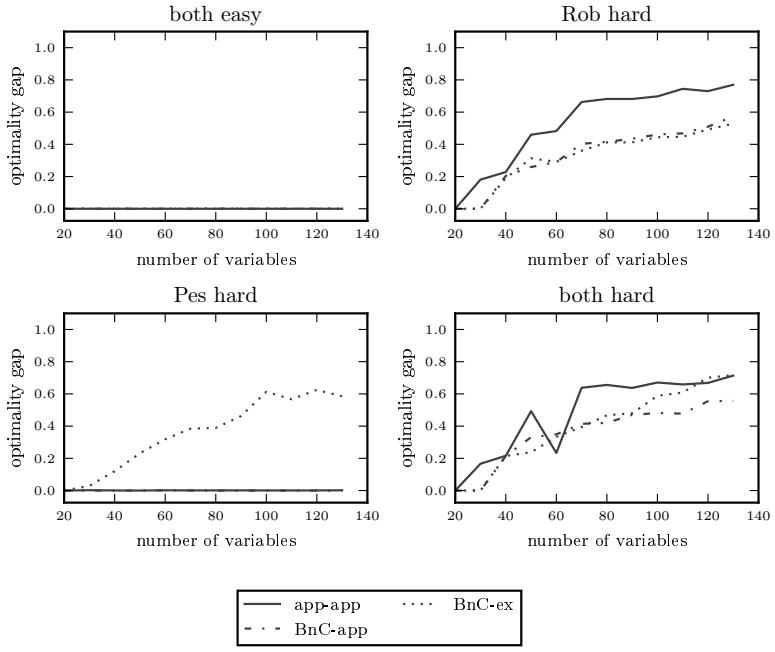


Figure 3: Comparison against branch-and-cut

It can be seen in Figure 3 that for the case of both problems being easy all algorithms return an optimal solution within the time limit. Furthermore when the robustification problem is hard the branch-and-cut algorithms seem to be a good choice. If, however, the pessimization

problem is hard, BnC-ex has a similar behaviour as ex-ex and app-ex from Section 4.1. In comparison both BnC-app and app-app find optimal solutions. When both problems are hard, we see that BnC-app yields the best results, followed by BnC-ex.

Table 3, in which we again let the algorithm solve instances to optimality, shows that app-app is the best approach if the pessimization problem is hard and the robustification is easy.

	Both easy	RC hard	Pes hard	Both hard
app-app	0.482	44.224	379.172	14.922
BnC-app	0.344	3.862	558.714	3.604
BnC-ex	0.512	5.24	1337.806	5.048

Table 3: Average runtime in seconds for solving five instances with 20 variables to optimality.

We conclude that it depends on the problem which implementation gives the best results. If the robustification problems are hard, branch-and-cut approaches seem to be superior while app-app has an advantage for instances with hard pessimization problems. This can be explained as follows: If applied to an instance with hard pessimization, the branch-and-cut approaches require too many iterations and hence have to solve too many pessimization problems, whereas app-app with a properly chosen t_{opt} does not need so many iterations and does not have to solve as many pessimization problems. This, however, might be highly dependent on the choice of t_{opt} and t_{pes}, which will be investigated in the next section.
This observation underlines a drawback of branch-and-cut: branch-and-cut can be seen as some robustification procedure with $t_{opt} = ub_k - \epsilon$, because in this branch-and-bound procedure we do not have any choice but to consider every feasible solution (and hence determine a scenario via pessimization) that has been found.

We also remark that branch-and-cut can only be applied to mixed integer linear programs while the approximate cutting plane approach app-app can be used for any kind of robust optimization problem.

4.4. Fine tuning of thresholds

Finally we investigate how different choices of t_{pes} and t_{opt} affect the solution quality. We tried different combinations for t_{opt} and t_{pes}, where a higher t_{opt} denotes, according to its definition, a higher degree of approximation ($t_{opt} = 0$ means robustification is solved exactly) and a higher t_{pes} means also a higher degree of approximation. We used instances with $n = 100$ variables and averaged over 5 instances for each problem class. In the heatmaps shown in Figure 4 the black color denotes a high optimality gap, whereas white denotes a small optimality gap.
If both problems are chosen to be easy, the problem instances could be solved for almost all parameter settings.

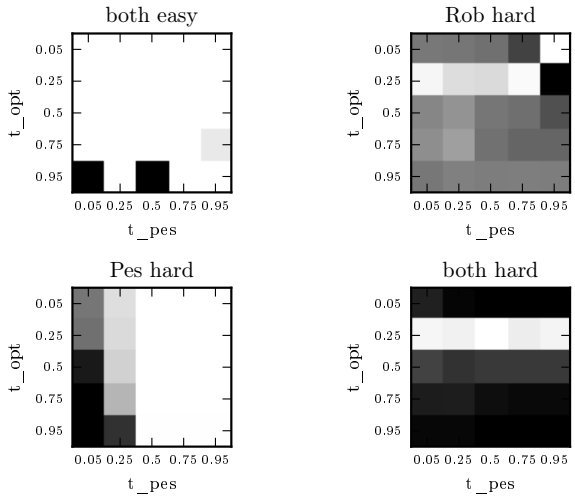


Figure 4: Different choices of the threshold parameters

If the pessimization problem is chosen to be the difficult one, we observe that the best results are obtained if the pessimization step is solved approximately with a high degree of approximation. The color gets lighter when t_{pes} increases. In this case, the results do not depend on the choice of t_{opt}. The reason is that the time spent in the robustification step is negligible as the robustification problem was chosen to be easy.

If both robustification and pessimization are hard, we see that a choice of t_{opt} of approximately 0.25 combined with a choice for t_{pes} of approximately 0.5 yields the best results. This can be explained as follows: If t_{pes} is chosen too small, we abort the pessimization step too early, hence we do not add a good cut to the problem. If, on the other hand, t_{pes} is too large, each pessimization step just takes too long. For the choice of t_{opt}, the explanation is similar. If t_{opt} is chosen too small, it takes too long to obtain the next solution. If it is chosen too large, it considers too many solutions that should be discarded.

When the robustification problem is chosen to be the difficult problem, the parameter choice behaves similarly to the case of both hard as we have seen already in the previous experiments. This could be explained by the fact that the robustification problem seems to be more difficult than the pessimization problem, which is further justified in the next section.

4.5. Summary of experiments

In general the time spent in pessimization correlates strongly with the number of iterations because finding a worst-case scenario is always the same problem. The overall time spent in robustification does not depend so much on the number of iterations. The time spent

for a single robustification namely increases with increasing iteration k since $|\mathcal{U}_k|$ increases. The pessimization on the other hand works faster because the threshold ub_k decreases with increasing k.

Overall, we have seen in this chapter that there are quite a few parameters that can be tuned in the implementation of Algorithm 1. Most importantly, we have seen that the runtime of the implemented algorithm highly depends on the choices of t_{pes} and t_{opt}. The key message to keep in mind is that a good choice highly depends on the problem type that is considered, i.e., if the pessimization or the robustification is the bottleneck. Furthermore, if the considered problems are of mixed-integer linear type, the three proposed improvements of warm start, strengthening bounds and branch-and-cut seem to be promising for speeding up the solution process.

5. Conclusion

We propose an approximate cutting plane approach for solving robust optimization problems. We introduced threshold parameters to control the degree of approximation for each subproblem (i.e., robustification and pessimization). We have shown that the approximate cutting plane approach converges to an exact optimal solution under similar conditions as traditional cutting plane approaches for robust optimization. For finite uncertainty sets and in combinatorial optimization, bounds on the number of iterations have been shown. In the computational experiments we were able to show for the broad class of robust mixed-integer linear programs that the approximate cutting plane approach outperforms the traditional one. We also propose and investigate several improvements for mixed integer linear optimization problems. The computational experiments also underline that the parameter choice should be done with precise problem knowledge in order to save significant amounts of computation time.

Further research on cutting plane approaches for robust optimization includes the following point. First of all, it is rather easy to get rid if the assumptions we made on the solvability of A-RC($\mathcal{U}_k, t_{opt_k}$) and A-Pes(x_k, t_{pes_k}). Even if these problems are not bounded, cutting plane algorithms can be formulated, but it becomes more technical since several cases have then to be distinguished in the algorithm. Second, we only considered uncertainty in the objective function. The cutting plane approach can also be formulated for the case of uncertainty in the constraints; here one even has several possibilities on how to define a worst-case scenario, and on choosing the cuts to be added to the robustification step.

Another line of research is to extend the analysis of this paper to other robustness concepts apart from strict robustness. For example, when determining regret robustness Kouvelis & Yu (1997) or recovery robustness (Liebchen et al. (2009); Goerigk & Schöbel (2014); Carrizosa et al. (2017)), we are faced with a hard pessimization step which makes the application of

our approach in particular promising. The approach can also be applied to light robustness Schöbel (2014). It is also ongoing research to use iterative approaches for multi-objective robust problems by applying recent approaches of Botte & Schöbel (2019); Schmidt et al. (2019).

Additionally the iterative approach could be further enhanced such that robustification and pessimization run in parallel: If a solution is found, the algorithm could on the one hand go to the pessimization step for this solution. But, on the other hand, the algorithm could continue solving the robustification problem and check if a better solution can be found. If in the pessimization step a new solution has been found, we can add it as a constraint and start solving robustification again and/or start pessimization for the next solution in queue. Especially within a branch-and-cut framework this enhancement could further improve the algorithm runtime significantly.

Finally, we plan to apply the approximate cutting plane approach to real-world robust optimization problems, in particular for robust load planning (as in Bruns et al. (2014)) and for finding robust timetables (see a very recent overview in Lusby et al. (2018)). The very recent study of Pätzold (2019) deals with an adjustable robust timetabling problem that includes delay management decisions in the adjusting phase. This problem could only be solved by applying the approximate cutting plane as proposed in this paper and hence proves the applicability of our method. We also plan to apply the approximate cutting plan method in combination with other heuristics (Goerigk & Schöbel (2013)) and to apply it to other variations for robustness concepts for timetabling such as the the scenario-based approach as described in Goerigk & Schöbel (2010).

References

Aissi, H., Bazgan, C., & Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, *197*, 427–438.

Assavapokee, T., Realff, M. J., Ammons, J. C., & Hong, I.-H. (2008). Scenario relaxation algorithm for finite scenario-based min–max regret and min–max relative regret robust optimization. *Computers & operations research*, *35*, 2093–2102.

Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. (Eds.) (2006). *Special issue on Robust Optimization* volume 107:1-2 of *Mathematical Programming B*. Springer.

Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. (2009). *Robust Optimization*. Princeton and Oxford: Princeton University Press.

Ben-Tal, A., & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, *23*, 769–805.

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

Bertsimas, D., Dunning, I., & Lubin, M. (2016). Reformulation versus cutting-planes for robust optimization. *Computational Management Science*, *13*, 195–217.

Bertsimas, D., Iancu, D. A., & Parrilo, P. A. (2011). A hierarchy of near-optimal policies for multistage adaptive optimization. *IEEE Transactions on Automatic Control*, *56*, 2809–2824.

Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research*, *52*, 35–53.

Bienstock, D. (2007). Histogram models for robust portfolio optimization. *Journal of computational finance*, *11*, 1–64.

Botte, M., & Schöbel, A. (2019). Dominance for multi-objective robust optimization concepts. *European Journal of Operational Research*, *273*, 430–440.

Bruns, F., Goerigk, M., Knust, S., & Schöbel, A. (2014). Robust load planning of trains in intermodal transportation. *OR Spectrum*, *36*, 631–668.

Bürger, M., Notarstefano, G., & Allgöwer, F. (2014). A polyhedral approximation framework for convex and robust distributed optimization. *IEEE Transactions on Automatic Control*, *59*, 384–395.

Calafiore, G., & Campi, M. C. (2005). Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, *102*, 25–46.

Calafiore, G. C. (2010). Random convex programs. *SIAM Journal on Optimization*, *20*, 3427–3464.

Campi, M. C., & Garatti, S. (2008). The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, *19*, 1211–1230.

Carrizosa, E., Goerigk, M., & Schöbel, A. (2017). A biobjective approach to recovery robustness based on location planning. *European Journal of Operational Research*, *261*, 421–435.

Fischetti, M., & Monaci, M. (2012). Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation*, *4*, 239–273.

Ghaoui, L. E., & Lebret, H. (1997). Robust solutions to least-squares problems with uncertain data. *SIAM Journal of Matrix Anal. Appl.*, *18*, 1035–1064.

Goerigk, M., & Schöbel, A. (2010). An empirical analysis of robustness concepts for timetabling. In T. Erlebach, & M. Lübbecke (Eds.), *Proceedings of ATMOS10* (pp. 100–113). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik volume 14 of *OpenAccess Series in Informatics (OASIcs)*. URL: `http://drops.dagstuhl.de/opus/volltexte/2010/2753`. doi:http://dx.doi.org/10.4230/OASIcs.ATMOS.2010.100.

Goerigk, M., & Schöbel, A. (2013). Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, *40*, 1363–1370.

Goerigk, M., & Schöbel, A. (2014). Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. *Computers and Operations Research*, *52*, 1–15.

Goerigk, M., & Schöbel, A. (2016). Algorithm engineering in robust optimization. In L. Kliemann, & P. Sanders (Eds.), *Algorithm Engineering: Selected Results and Surveys* (pp. 245–279). volume 9220 of *LNCS State of the Art*. URL: `http://arxiv.org/abs/1505.04901`.

Kelley, J. E., Jr (1960). The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, *8*, 703–712.

Kouvelis, P., & Yu, G. (1997). *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers.

Liebchen, C., Lübbecke, M., Möhring, R. H., & Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R. Möhring, & C. Zaroliagis (Eds.), *Robust and online large-scale optimization*. Springer volume 5868 of *Lecture Note on Computer Science*.

Lusby, R., Larsen, J., & Bull, S. (2018). A survey ob robustness in railway planning. *European Journal of Operationsaal Research*, *266*, 1–15.

Montemanni, R. (2006). A benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, *174*, 1479–1490.

Mutapcic, A., & Boyd, S. (2009). Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods & Software*, *24*, 381–406.

Pätzold, J. (2019). Finding Robust Periodic Timetables by Integrating Delay Management. Submitted.

Pérez-Galarce, F., Álvarez-Miranda, E., Candia-Véjar, A., & Toth, P. (2014). On exact solutions for the minmax regret spanning tree problem. *Computers & Operations Research*, *47*, 114–122.

Reemtsen, R. (1994). Some outer approximation methods for semi-infinite optimization problems. *Journal of Computational and Applied Mathematics*, *53*, 87–108.

Schmidt, M., Schöbel, A., & Thom, L. (2019). Min-ordering and max-ordering scalarization methods for multi-objective robust optimization. *European Journal of Operational Research*, *275*, 446–459.

Schöbel, A. (2014). Generalized light robustness and the trade-off between robustness and nominal quality. *MMOR*, *80*, 161–191.

Siddiqui, S., Azarm, S., & Gabriel, S. (2011). A modified benders decomposition method for efficient robust optimization under interval uncertainty. *Structural and Multidisciplinary Optimization*, *44*, 259–275.

Soyster, A. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, *21*, 1154–1157.

E. Finding Robust Periodic Timetables by Integrating Delay Management

J. Pätzold
Finding Robust Periodic Timetables by Integrating Delay Management
Submitted to Public Transport, 2019.

Finding Robust Periodic Timetables by Integrating Delay Management

Julius Pätzold
Institute for Numerical and Applied Mathematics
University of Goettingen
Lotzestraße 16 - 18
37083 Göttingen, Germany
j.paetzold@math.uni-goettingen.de

July 10, 2019

Abstract

This paper defines and solves a mathematical model for finding robust periodic timetables by proposing an extension of the Periodic Event Scheduling Problem (PESP). In order to model delayed and not nominal travel times already in the timetabling step, we integrate delay management into the periodic timetabling problem. After revisiting both (PESP) and delay management individually, we introduce a periodic delay management model capable of evaluating periodic timetables with respect to delay resistance. Having introduced periodic delay management, we define the Robust Periodic Timetabling problem (RPT). Due to the high complexity of (RPT) we propose two different simplifications of the problem and introduce solution algorithms for both of them. These solution algorithms are tested against timetables found by standard procedures for periodic timetabling with respect to their delay-resistance. The computational results show that our algorithms yield timetables which can cope better with occurring delays, even on large-scale datasets and with low computational effort.

1 Introduction

Public transportation planning can be conceived of as a range of different subproblems that need to be solved: On a strategic level, a public transportation networks need to be designed, operating lines need to be determined, and a timetable needs to be found. On an operational level, vehicles and crew need to be scheduled and finally delays have to be coped with. Each of these subproblems can be solved with respect to different objectives: From the viewpoint of the operating company, for example, the goal is to keep the costs of the public transport system low; passengers, on the other hand, want to have short travel times. Instead of viewing the planning process of public transportation as a whole, most research instead focuses on picking a certain subproblem, e.g., only line planning or only timetabling, and solving it individually. This process, however, merely leads to some local optimal solution. Nevertheless, recent work, e.g., [PLM$^+$13; Sch14; Sch17; BBV$^+$17; PSS$^+$17; PSS18] shows that the integration of subproblems is often superior to solving single problems sequentially. For example, [PSS18] show how the integration of line planning, timetabling and vehicle scheduling leads to cost-optimal public transport plans that still can be computed efficiently. In this paper,

though, we take on the passengers' view on a public transport system and concentrate on finding delay-resistant timetables. Delay-resistance is a key aspect when modeling a quality measure from a passengers' point of view. Nominal travel times, i.e., estimated travel times without delays, are certainly relevant for the attractiveness of traveling by train in general; Nevertheless, the amount of experienced delay (and uncertainty in general) impacts the passengers' reception on a public transport system even more, mostly in a negative way. Put differently, complaints about the nominal length of travel times are rare, whereas train punctuality is highly complained about and discussed by customers and press (see, e.g., [Con18]). Despite its high practical relevance, this aspect of considering delayed travel times is often neglected in periodic timetabling research. In our paper we take this factor into account and include a robustness measure – motivated by delay management considerations – and hence train punctuality into periodic timetabling. To meet this objective, we propose an approach to integrate the two problems of periodic timetabling and delay management by incorporating the concept of adjustable robust optimization.
The remainder of this paper is structured as follows: First, we introduce periodic timetabling and give a literature overview including how robustness concepts are applied to it. In the next section we define delay management and give a model for adapting it to periodic timetabling, hence defining a new evaluation of a periodic timetable. In Section 4, we formulate the problem of finding robust periodic timetables and give two solution approaches based on iterative cutting-plane techniques. We reinforce the quality of our approaches via computational experiments in Section 5, and conclude the paper in Section 6.

2 Periodic Timetabling

Periodic timetabling is one of the most difficult problems in public transport optimization. Since its introduction in [SU89] it has been studied extensively, see [Odi96; Nac98; Pee03; Lie07]. The problem first requires the definition of an event-activity-network.

Definition 1. *A periodic event-activity-network (EAN) is a directed graph $(\mathcal{E}, \mathcal{A})$. The nodes $\mathcal{E} = \mathcal{E}^{dep} \cup \mathcal{E}^{arr}$ are divided into departure and arrival events. Furthermore the activities are divided into the set of driving and waiting activities $\mathcal{A}^{dw}$ and the set of changing activities $\mathcal{A}^{ch}$, i.e., $\mathcal{A} = \mathcal{A}^{dw} \cup \mathcal{A}^{ch}$. We assume that A^{dw} corresponds to train lines and hence forms a set of node-disjoint paths $\{l_1, ..., l_n\}$ covering all events $\mathcal{E}$. Each activity $a \in \mathcal{A}$ has lower and upper bounds $[L_a, U_a]$ for the allowed duration and a weight w_a corresponding to the number of passengers traveling on that activity. For every event $i \in \mathcal{E}$ we are also given the number of passengers w_i unboarding the train at event i (for arrival events $i \in \mathcal{E}^{arr}$) or boarding the train at event i (for departure events $i \in \mathcal{E}^{dep}$).*

Now the problem can be stated as follows.

Definition 2 ([SU89])**.** *Given a periodic EAN $(\mathcal{E}, \mathcal{A})$, the Periodic Event Scheduling Problem*

assigns a time $\pi_e \in \mathbb{N}$ to each event $e \in \mathcal{E}$ in order to solve

$$\begin{aligned}
\min \quad & \sum_{a=(i,j)\in\mathcal{A}} w_a(\pi_j - \pi_i + z_a T) && \text{(PESP)}\\
s.t. \quad & \pi_j - \pi_i + z_a T \leq U_a \quad \forall a = (i,j) \in \mathcal{A}, && (1)\\
& \pi_j - \pi_i + z_a T \geq L_a \quad \forall a = (i,j) \in \mathcal{A}, && (2)\\
& z_a \in \mathbb{Z} \quad \forall a \in \mathcal{A},\\
& \pi_i \in \mathbb{N} \quad \forall i \in \mathcal{E}.
\end{aligned}$$

$T \in \mathbb{N}$ denotes the time period, which is usually assumed to be 60 minutes. For later use we define $\pi_a \coloneqq \pi_j - \pi_i + z_a T$ to be the time duration for all activities $a \in \mathcal{A}$ (with the summand $z_a T$ modeling the "modulo T" operation). We refer to a vector π as a timetable. The set of all feasible timetables to an EAN is $\Pi = \Pi(\mathcal{E}, \mathcal{A})$.

Several solution approaches have been proposed with the most prominent improvements being the cycle-base formulation (see [PK01; BHK+18]), the modulo simplex algorithm ([NO08; GS13]) and SAT-Formulations ([GHM+12]). Until today, there is an ongoing endeavor to improve on solving instances of PESP. Compared on the benchmark PESPlib (see [Goe]), the current best working solutions use a combination of several approaches to improve on the current best solutions: [GL17] combine modulo simplex and cycle-base formulations in an iterative manner. This paper's author extended [PS16] to a general divide-and-conquer approach and applied the cycle-base formulation with several heuristics to improve on their solutions. Recently, [BLR19] combine Modulo Simplex, SAT, IP approaches and heuristics to deliver the current best solutions for PESPlib.

When solving (PESP), the objective is to minimize the sum of all passengers' travel times, which seems to be reasonable at first glance. In practice, however, there exist at least two problems for adapting a timetable found by solving (PESP):
First, one does not know how passengers choose their routes in the EAN beforehand. For this problem, which formally boils down to the weights w_a being variable instead of fixed, several attempts have been made to formulate an optimization problem that integrates the choice of passenger routes, see [SG13; Sch14; SS15a; SS15b; GGN+16; BHK17].
The second problem is that there are usually delays in the network that need to be dealt with. The total amount of delay in a network may not be clear beforehand, but it is definitely unrealistic to completely neglect their existence and to optimize a timetable that is only guaranteed to work well in the nominal case of no delays (as is done when solving PESP). There exist several attempts to overcome this second problem of timetables being deficiently delay-resistant:
[Goe15] applies the concept of recovery robustness for periodic timetabling, which was first introduced in [LLM+09] and extended in [GS14] for aperiodic timetabling. It allows the modification of a solution after some scenario has been revealed. The aim of [Goe15] is to minimize the cost of recovering to a solution over all scenarios. A heuristic approach for large instances is also given by solving a bicriteria optimization problem with the two linear objectives of travel time and robustness. A different approach to solve this problem is given in [PBD+19]. The authors define a robust version of (PESP) by assuming that delays impact the interval $[L_a, U_a]$ for each activity and require a feasible robust timetable to be able to adjust its times in order to maintain feasibility for all activities. Another proposal

for finding delay-resistant timetables makes use of stochastic optimization: In [KMH+08], the authors sample scenarios and try to optimize a rolled-out version of the timetable for all disturbances occurring in each of these samples. By restricting themselves to keep the cyclic train order fixed and only optimizing the slack times that can be allocated, the authors maintain tractability of the problem. Further research on robust periodic timetabling research includes [KDV07; LSS+10; CFL+11; Mar17]. Next to robust periodic timetabling, there exist robust variants of the aperiodic timetabling problem, which is polynomially solvable in its nominal version. Literature includes [FSZ09; GS10; DHS+12; CCF12; LTZ+17]. For surveys on robustness in timetabling (periodic and aperiodic), see [CT12] and [LLB18].
Interestingly, many of the mentioned attempts to robustify periodic timetabling view the problem as a three-stage process, consisting of

$$\text{timetabling} \rightarrow \text{scenario reveals} \rightarrow \text{delay management}.$$

The third stage is just called differently: For example, [Goe15] call it recovery of the timetable, [PBD+19] call it adjustment by a linear decision rule and [KMH+08] omit the third stage by mentioning that their model does not include traffic control decisions. In this paper, we comprehend the third stage (delay management) as such by incorporating the original delay management problem as defined in the public transport optimization literature (see also next section). By doing so, we not only give a new model for robust periodic timetabling, but also model an integration of the periodic timetabling and the delay management problem. In the following we define the latter problem.

3 Delay Management and Periodicity

Delays constitute a major source of uncertainty when operating a bus or railway system. If a train is delayed, many rescheduling decisions have to be made, each of which may disturb the nominal schedule of a public transport system. The question of whether an otherwise punctual train should wait for a delayed feeder train in order to allow transferring passengers to reach their connection is known as delay management problem and has been studied extensively in the literature. The first papers dealing with this kind of question date back to [Sch01; SBK01]. Integer programming models have been developed in [Sch07; DHL08] and a recent survey about delay management models can be found in [DHS+18].
Delay management is traditionally carried out in an aperiodic way. To this end, a periodic EAN and timetable can be rolled out for a certain time horizon in order to receive their aperiodic pendants.

Definition 3. *Let a periodic EAN $(\mathcal{E}, \mathcal{A})$, a time period T and a timetable $\pi \in \Pi(\mathcal{E}, \mathcal{A})$ be given. For a time horizon $[L, U] \subset \mathbb{R}$ we define an aperiodic EAN $(\mathcal{E}^*, \mathcal{A}^*)$ by*

$$\mathcal{E}^* := \{(e, n) \in \mathcal{E} \times \mathbb{Z} \mid L \leq n \cdot T + \pi_e \leq U\},$$
$$\mathcal{A}^* := \{(i, n), (j, m) \in \mathcal{E}^* \times \mathcal{E}^* \mid (i, j) = a \in \mathcal{A} \ \wedge \ L_a \leq \pi_j - \pi_i + (m - n)T \leq U_a\},$$

and the aperiodic timetable $\pi^*_i = \pi_i + nT$ for all $(i, n) \in \mathcal{E}^*$.

To formulate an integer programming model for the delay management problem, we need to formally introduce delays. As is commonly done, we assume that a set of potentially expected source delays is known, e.g., caused by signaling problems, construction work, accidents, or

bad weather conditions. These source delays cause propagated delays, e.g., for the same train at subsequent stations or for other trains that wait for the delayed train. As in [SS10] we allow two types of source delays: The first type is a delay $s_e \in \mathbb{N}$ at an event $e \in \mathcal{E}$ (e.g., staff being late for their shift) referring to a fixed point in time. The second type of source delay is a delay s_a which increases the duration of an activity $a \in \mathcal{A}$, e.g., an increase of travel time between two stations due to construction work. Such a delay s_a has to be added to the minimal duration L_a of activity a. If an event or activity has no source delay, we assume $s_e = 0$ or $s_a = 0$, respectively. We hence define the set of possible source delays for an EAN as

Definition 4. *Given an EAN (periodic or aperiodic) $(\mathcal{E}, \mathcal{A})$, define a set of scenarios $\mathcal{S}$ as*

$$\mathcal{S}(\mathcal{E}, \mathcal{A}) \coloneqq \{s \in \mathbb{R}_{\geq 0}^{|\mathcal{E}|+|\mathcal{A}|} |\ s_i \leq \sigma\ \forall i \in \mathcal{E} \cup \mathcal{A}, \|s\|_1 \leq \rho\},$$

with $\sigma \geq 0$ as the maximum single delay and $\rho \geq 0$ as the maximum sum of all source delays. We assume that there are no source delays on the change activities ($s_a = 0$ for all $a \in \mathcal{A}^{ch}$).

Hence, we define the uncertainty set similar to [PBD+19], which originates from [BBC11] that introduce a parameter (here, ρ) to regulate the "budget of uncertainty" which corresponds in this case to the amount of source delay ρ that can be distributed to the activities $a \in \mathcal{A}$.
Having defined source delays, we can now state the integer programming formulation for the delay management problem. To model the wait-depart decisions, i.e., whether some train should wait for some other train at a station or not, we introduce binary variables

$$y_a = \begin{cases} 0 & \text{if changing activity } a \text{ is maintained,} \\ 1 & \text{otherwise,} \end{cases}$$

for all changing activities $a \in \mathcal{A}^{ch}$. The integer programming formulation then reads as follows:

Definition 5 ([Sch01])**.** *Given an aperiodic EAN $(\mathcal{E}^*, \mathcal{A}^*)$, an associated timetable $\pi \in \mathbb{N}^{|\mathcal{E}^*|}$, and source delays $s \in \mathcal{S}(\mathcal{E}^*, \mathcal{A}^*)$, define the delay management problem as*

$$\min \quad \sum_{i \in \mathcal{E}^{*arr}} w_i d_i + \sum_{a \in \mathcal{A}^{*ch}} y_a w_a T \tag{DM}$$

$$\text{s.t.} \quad d_i \geq s_i \qquad \forall i \in \mathcal{E}^*, \tag{3}$$

$$\pi_j - \pi_i + d_j - d_i \geq L_a + s_a \qquad \forall a = (i,j) \in \mathcal{A}^{*dw}, \tag{4}$$

$$M y_a + \pi_j - \pi_i + d_j - d_i \geq L_a \qquad \forall a = (i,j) \in \mathcal{A}^{*ch}, \tag{5}$$

$$d_i \in \mathbb{R} \qquad \forall i \in \mathcal{E}^*,$$

$$y_a \in \{0,1\} \qquad \forall a \in \mathcal{A}^{*ch}.$$

The new timetable, called disposition timetable, is now defined as $(\pi_i + d_i)_{i \in \mathcal{E}^*}$.

Self-evidently, the delays d_i have to be greater than the source delays at the respective events $i \in \mathcal{E}^*$. Then, for every driving or waiting activity $a \in \mathcal{A}^{*dw}$ the duration $\pi_j + d_j - (\pi_i + d_i)$ after disposition needs to be greater than the lower bound L_a plus some possible source delays on that activity s_a because we assume that the train cannot drive faster than that, i.e., (4). For the changing activities $a \in \mathcal{A}^{*ch}$ the model can decide if a change is maintained

($y_a = 0$). In this case this activity has to satisfy the same inequality as all driving and waiting activities. If, on the other hand, the change is not maintained, then big M in (5) is triggered and the inequality does always hold. In that case, however, the objective function adds a full time period T for every passengers that has missed the respective change. The objective function furthermore sums up the delay for all passengers up to the point at which they get off at their destination. This, of course, is only an approximation of reality. There exist more sophisticated models that take passenger rerouting into account, but they do so at the expense of a much more complicated and hence slower model.
The d-variables of model (DM) are defined slightly differently ($x_i := d_i + \pi_i$ for $i \in \mathcal{E}^*$, see, e.g., [Sch07]), but the above notation will be more convenient for later use. Also note that (DM) does not consider upper bounds U_a for activities $a \in \mathcal{A}$, which is reasonable since the duration π_a of an activity plus the propagated delay d_a should not be bounded by a model assumption, as this might lead to infeasibilities of the model, e.g., if $L_a + s_a > U_a$. Removing this assumption also from (PESP) would be a viable option, but is neglected here in order to maintain feasibility of the obtained timetable for (PESP).

There exist several shortcomings of this delay management formulation: The passenger distribution w, for example, is not fixed in reality, and also penalizing a missed change with exactly T minutes is not always correct. Nevertheless, (DM) can be regarded as a reasonable approximation for the delay management process, hence its establishment in the literature. In this paper we make use of the simplicity of (DM) by being able to fit a modified version of it into periodic timetabling. By doing so we are able to give an approximative evaluation of the behavior of a periodic timetable in (aperiodic) delay management, i.e., for (DM). The modification of (DM) to a periodic setting is the following: We assume that delays occur periodically and that, accordingly, the delayed timetable d also needs to work periodically. This yields the subsequent model.

Definition 6. *Given a periodic EAN $(\mathcal{E}, \mathcal{A})$, a periodic timetable $\pi \in \Pi(\mathcal{E}, \mathcal{A})$ and source delays $s \in \mathcal{S}(\mathcal{E}, \mathcal{A})$, we define the periodic delay management problem as*

$$\begin{aligned}
\min \quad & \sum_{a \in \mathcal{A}} w_a d_a + \sum_{i \in \mathcal{E}^{dep}} w_i d_i && && \text{(P-DM)}\\
s.t. \quad & d_i \geq s_i && \forall i \in \mathcal{E}, && (6)\\
& d_a = d_j - d_i && \forall a = (i,j) \in \mathcal{A}^{dw}, && (7)\\
& d_a = d_j - d_i + z_a T && \forall a = (i,j) \in \mathcal{A}^{ch}, && (8)\\
& \pi_a + d_a \geq L_a + s_a && \forall a = (i,j) \in \mathcal{A}, && (9)\\
& d_i \in \mathbb{R} && \forall i \in \mathcal{E} \cup \mathcal{A},\\
& z_a \in \mathbb{Z} && \forall a \in \mathcal{A}^{ch},
\end{aligned}$$

with the set of all feasible propagated delays d being

$$\mathcal{D} = \mathcal{D}(\pi, s) := \{d \in \mathbb{R}^{|\mathcal{E}|+|\mathcal{A}|} \mid \exists\, z \in \mathbb{Z}^{|\mathcal{A}^{ch}|} \text{ s.th. } (d, z) \text{ is feasible for (P-DM)}\}.$$

At first glance, (P-DM) looks fairly similar to (DM): The propagated delays should still be larger than the source delays, i.e., (6). In (7) we then introduce propagated delays on activities by setting $d_a = d_j - d_i$, which could also be done for (DM). The delay of a train adds up along driving and waiting activities (since these are executed by the same train) which is why

we have to respect periodicity for delays d_a only for changing activities, i.e., (8). A thorough example is given in Figure 1. Note that this model assumes (similar to (DM)) that the choice of passenger paths is fixed. Otherwise a source delay of T should not influence a periodic timetable because all passengers are able to take an earlier train which is then delayed by T minutes.
In the objective function we sum up the delayed activities and delayed departure events, which corresponds to summing up the delay on arrival events, since for every node-path $(e_1, ..., e_n)$ in the EAN it holds that

$$d_{e_n} = d_{e_1} + \sum_{i=2}^{n} d_{e_i} - d_{e_{i-1}} = d_{e_1} + \sum_{i=2}^{n} d_{(e_i, e_{i-1})}.$$

By summing up along the activities (and departure events) and due to the periodicity we do not have to deal with missed changes separately, since a missed change is modeled by the periodicity as a long-lasting change activity (cf. (d) in Figure 1). Whereas this provides a way to cope with cases of one train being delayed for more than one hour, the periodicity shrinks the space of source scenarios: In model (P-DM) every repetition of a train is assumed to have the same source delay and for each repeating train the same delay management strategy has to be chosen. Then again, it can be reasoned that it is of course possible to consider not only one but many different scenarios and to choose a timetable that is robust (or delay-resistant) to all of them. In Section 5 we will discuss whether a periodic timetable found by minimizing (P-DM) is also delay-resistant when rolled out to an aperiodic network and evaluated by (DM). As a side note, see that (P-DM) can also be used to model disturbances like construction work: In this case source delays occur periodically and a revised periodic timetable needs to be found.
The problem with delay management models by themselves, be it (DM) or (P-DM), is that the nominal timetable is fixed and hence the model can only minimize the delays d and not overall travel times $\pi + d$. This fact hence leads to the idea of combining the two objectives of nominal travel time π and delayed times d into a problem that evaluates a timetable by its overall behavior, meaning nominal travel time plus delays.

4 Robust Periodic Timetabling

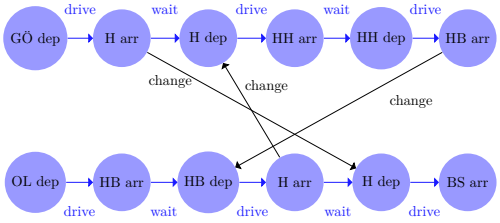
In this section we now define a problem that allows us to find delay-resistant timetables. To this end, we make use of the concept of adjustable robustness (cf. [BGG$^+$04]) which arises in robust optimization (see, e.g., [BEN09; GS16]) to define the following problem.

Definition 7. *Given a periodic EAN $(\mathcal{E}, \mathcal{A})$, we search for a timetable $\pi \in \Pi$ that has the best worst-case behavior with respect to all scenarios $s \in \mathcal{S} = \mathcal{S}(\mathcal{E}, \mathcal{A})$ that are likely to happen. Formally, we want to solve the problem*

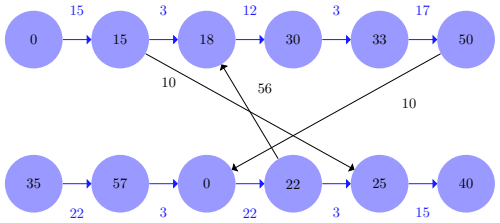
$$\min_{\pi \in \Pi} \sup_{s \in \mathcal{S}} \min_{d \in \mathcal{D}(\pi, s)} \tau(\pi, d) \tag{RPT}$$

with

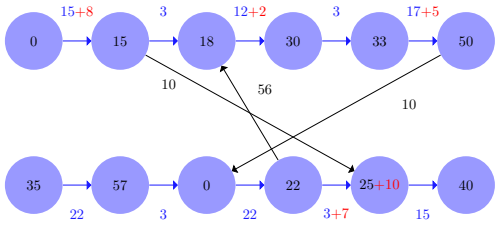
$$\tau(\pi, d) \coloneqq \sum_{a \in \mathcal{A}} w_a(\pi_a + d_a) + \sum_{i \in \mathcal{E}^{dep}} w_i d_i.$$



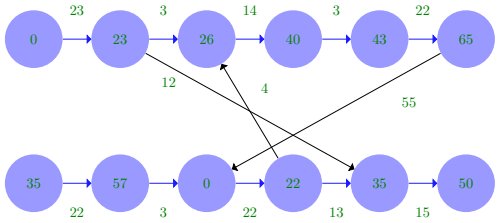
(a) Periodic EAN



(b) Periodic Timetable with all $\pi_i, i \in \mathcal{E}$ and $\pi_a, a \in \mathcal{A}$



(c) Periodic Source Delays added, i.e., s_i, $i \in \mathcal{E}$ and s_a, $a \in \mathcal{A}$



(d) Periodic Disposition Timetable, i.e., $\pi_i + d_i$, $i \in \mathcal{E}$ and $\pi_a + d_a$, $a \in \mathcal{A}$

Figure 1: Periodic Delay Management

The intention behind minimizing a timetable π against its worst-case scenario $s \in \mathcal{S}$ is that we want to require robustness against "small" perturbations of the nominal timetable. These

perturbations $\mathcal{S}$ are modeled by choosing σ and ρ to be rather small parameters. Details can be found in Table 1 in Section 5.
Of course, solving (RPT) is quite an ambitious endeavor: In Section 2 we already mentioned the intrinsic difficulty of solving (PESP) in itself. In (RPT), however, an additional layer of difficulty is added since, even if a timetable π is fixed, the remaining sup-min problem of determining the worst-case scenario is a discrete bilevel-optimization problem (or, more precisely, a continuous-discrete bilevel problem, as $\mathcal{S}$ is continuous, but the $z-$variables within the constraints of $\mathcal{D}$ are discrete), for which there is no known procedure to generally solve it to optimality (cf. [SMD18]). Nonetheless, for (RPT) in general we can at least show the existence of a minimal timetable $\pi \in \Pi$.

Lemma 8. *There exists an optimal solution $\pi \in \Pi$ to* (RPT)*.*

Proof. The objective function τ of (RPT) is linear in $d \in \mathcal{D}(\pi, s)$ as it just sums up the d-variables. The infimum of concave functions (i.e., especially linear functions) is again concave. Hence $\varphi(\pi, s) := \min_{d \in \mathcal{D}(\pi,s)} \tau(\pi, d)$ is concave. From the concavity of φ and compactness of $\mathcal{S}$ we derive that $\sup_{s \in \mathcal{S}} \varphi(\pi, s)$ is finite. Since Π is a finite set (all π_i can be assumed to lie in $[0, T-1]$ due to periodicity), we can enumerate all timetables and determine the one with the smallest supremum. This yields the minimal solution to (RPT). □

Thus, there exists an optimal timetable π (although there might not exist a corresponding worst-case scenario $s \in \mathcal{S}$ since $\varphi(\pi, s)$ might not be continuous due to the z-variables hidden in $\mathcal{D}$), but it is highly unlikely to determine it in reasonable time. We hence need to simplify (RPT) in order to achieve our goal of finding robust periodic timetables. To this end, we propose two different simplifications.

(A) We assume that the strategy for delay management can be expressed in terms of timetable and scenario. This can be done, for example, by enforcing a no-wait policy for trains. With the fixed strategy we can transform the inner max-min problem of finding the worst-case scenario to a mixed-integer maximization problem. The remaining min-max problem can then be solved by a cutting-planes approach for robust optimization problems.

(B) We find a solution to the inner max-min problem heuristically, e.g., by sampling scenarios until a bad-case (not worst-case) scenario is found. We can then solve an integrated timetabling-delay-management problem with respect to some finite scenario set $\mathcal{S}' \subset \mathcal{S}$ and iteratively increase the scenario set until we are not able to find a worse scenario for the currently best timetable.

We can show that the first simplification, if solved to optimality, leads to an upper bound of (RPT), whereas the second simplification leads to a lower bound on (RPT). In the following we describe these two approaches in more detail.

4.1 Simplification A: Fixed Delay Management Strategy

To carry out the first approach, we fix our delay management strategy. In this paper, we chose the no-wait strategy (which was successfully done in [LSS+10] for minimizing the expected delay of a periodic timetable), meaning that no train waits for delayed passengers from other trains. Thus, delays are only propagated along driving and waiting activities, leading to the following form of d (see also [Sch06]). Formally:

Definition 9. *Assume an EAN $(\mathcal{E}, \mathcal{A})$, a timetable π, and some source delays s and let $\mathcal{E}^{start} \subset \mathcal{E}$ be the set of events in the EAN such that every $e \in \mathcal{E}^{start}$ has no incoming driving or waiting activity. The* no-wait strategy *for delay management returns a solution d to* (P-DM) *such that*

$$d_j = \begin{cases} s_j, & j \in \mathcal{E}^{start}, \\ \max\{d_i + s_a - (\pi_a - L_a), s_j\}, & (i,j) = a \in \mathcal{A}^{dw}, \end{cases}$$

holds for all events $i \in \mathcal{E}$. We denote the set of all feasible d for which this property holds as $\mathcal{D}'(\pi, s) \subseteq \mathcal{D}(\pi, s)$.

The explanation of this definition is as follows: The first event of every line of the underlying line concept has no incoming driving or waiting activity. Hence, there is no delay that could possibly be propagated. Accordingly, we can set the propagated delay to its source delay, i.e., $d_i = s_i$ for $i \in \mathcal{E}^{\text{start}}$. Now we propagate this delay along the line: The next event j has an incoming driving activity $a = (i, j)$. We know that event i is delayed by d_i and that we have $\pi_a - L_a$ slack on the activity (meaning that we can save $\pi_a - L_a$ time by driving faster), but there also exists a source delay s_a on this driving activity. By utilizing the slack of activity a we can hence change the propagated delay to $d_i + s_a - (\pi_a - L_a)$ units of time. After having saved time, we encounter the source delay s_j on event j which leads to a propagated delay of $d_j = \max\{d_i + s_a - (\pi_a - L_a), s_j\}$ since d_j is constrained by $d_j \geq s_j$ (from (P-DM), i.e., we are not allowed to schedule events earlier than in the nominal timetable). Now the propagated delays d_j are chosen such that the train drives along its line as fast as possible, potentially ignoring passengers on change activities, and thereby yielding the no-wait strategy.
We now can formulate a program that determines a scenario $s \in \mathcal{S}$ and show that this program finds a scenario that is indeed a worst-case scenario for the no-wait strategy.

Definition 10. *Given a timetable $\pi \in \Pi$, we can find a scenario $s \in \mathcal{S}$ regarding the no-wait strategy by solving*

$$\begin{aligned}
\max \quad & \tau(\pi, d) && && \text{(F-WC}(\pi)\text{)} \\
s.t. \quad & d_i = s_i && \forall i \in \mathcal{E}^{start}, && (10) \\
& d_j \geq d_i + s_a - (\pi_a - L_a) && \forall a = (i,j) \in \mathcal{A}^{dw}, && (11) \\
& -M z_a + d_j \leq d_i + s_a - (\pi_a - L_a) && \forall a = (i,j) \in \mathcal{A}^{dw}, && (12) \\
& d_j \leq M(1 - z_a) + s_j && \forall a = (i,j) \in \mathcal{A}^{dw}, && (13) \\
& \pi_a + d_a \leq L_a + T - 1 && \forall a \in \mathcal{A}^{ch}, && (14) \\
& z_a \in \{0,1\} && \forall a \in \mathcal{A}^{dw}, \\
& d \in \mathcal{D}(\pi, s), \\
& s \in S.
\end{aligned}$$

Theorem 11. *A scenario found by solving* (F-WC(π)) *yields a worst-case scenario for the no-wait strategy, i.e.,*

$$\max_{s \in \mathcal{S}} \min_{d \in \mathcal{D}'(\pi, s)} \tau(\pi, d) \Leftrightarrow \text{(F-WC}(\pi)\text{)}.$$

Proof. First, it is worth mentioning that (F-WC(π)) is indeed a mixed-integer linear program: The set $\mathcal{S}$ has only linear constraints and also $d \in \mathcal{D}(\pi, s)$ is linear as it is only a short way

of repeating constraints (6)-(9). For showing the equivalence we rewrite the left side of (11) with the help of Definition 9:

$$\max_{s\in\mathcal{S}} \min_{d\in\mathcal{D}'(\pi,s)} \tau(\pi,d) \Leftrightarrow \quad \max_{s\in\mathcal{S}} \quad \min_{d\in\mathcal{D}(\pi,s)} \tau(\pi,d) \tag{*}$$

$$\begin{aligned} \text{s.t.} \quad & d_i = s_i && \forall i \in \mathcal{E}^{\text{start}}, \\ & d_j = \max\{d_i + s_a - (\pi_a - L_a), s_j\} && \forall (i,j) = a \in \mathcal{A}^{dw}. \end{aligned}$$

Now we can linearize the max operator by introducing a big M and auxiliary variables z_a:

$$(*) \quad \Leftrightarrow \quad \max_{s\in\mathcal{S}} \quad \min\ \tau(\pi,d) \tag{**}$$

$$\begin{aligned} \text{s.t.} \quad & d_i = s_i && \forall i \in \mathcal{E}^{\text{start}} \\ & d_j \le (1-z_a)M + s_j && \forall (i,j) = a \in \mathcal{A}^{dw}, \\ & d_j \ge d_i + s_a - (\pi_a - L_a) && \forall (i,j) = a \in \mathcal{A}^{dw}, \\ & -z_a M + d_j \le d_i + s_a - (\pi_a - L_a) && \forall (i,j) = a \in \mathcal{A}^{dw}, \\ & z_a \in \{0,1\} && \forall a \in \mathcal{A}^{dw}, \\ & d \in \mathcal{D}(\pi,s). \end{aligned}$$

Note that $d_j \ge s_j \ \forall\ j \in \mathcal{E}$ is ensured by $d \in \mathcal{D}(\pi,s)$.
Finally, we can restrict $\pi_a + d_a \in [L_a, L_a + T - 1]$ for all changing activities $a \in \mathcal{A}^{ch}$ since the d_a are minimized and do not impact any other constraints. Hence

$$\pi_a + d_a \le L_a + T - 1 \quad \forall a \in \mathcal{A}^{ch} \tag{14}$$

can be added to the model. With this final constraint at hand it is left to show that the min operator can be dropped. This is the case because at this point the d- and z-variables are determined uniquely for every $s \in \mathcal{S}$.
To show this, consider that if we fix some $s \in \mathcal{S}$, the d_i for $i \in \mathcal{E}^{\text{start}}$ are determined to be equal to s_i. Then, starting from these start events $\mathcal{E}^{\text{start}}$, the delays propagate along all driving and waiting activities uniquely via $d_j = \max\{d_i + s_a - (\pi_a - L_a), s_j\}$ for all $(i,j) = a \in \mathcal{A}^{dw}$. Hence, all d_i for $i \in \mathcal{E}$ and also d_a (and z_a) for $a \in \mathcal{A}^{dw}$ are determined uniquely. Additionally, (14) determines d_a (and z_a) for all $a \in \mathcal{A}^{ch}$.
Thus, the inner minimum can be dropped as (by uniqueness) there exists exactly one feasible set of variables $(d,z) \in \mathcal{E} \times \mathcal{A} \times \mathcal{A}$ leading to $(**) \Leftrightarrow$ (F-WC(π)) and thereby to the Theorem's statement. □

Note that in (F-WC(π)) we can write max instead of sup as opposed to in the notation in (RPT): Since the sum of all source delays is bounded, the propagated delays are also bounded. Hence we can estimate an upper bound on the modulo parameters z_a for all $a \in \mathcal{A}^{ch}$ (because the difference $d_j - d_i$ is also bounded). In doing so, we can enumerate all possible combinations of values for integer variables. For each of these enumerations we solve a linear optimization problem with no integer variables and the maximal of these finite number of different solutions is the maximal solution.
With the knowledge that (F-WC(π)) yields a worst-case scenario for a timetable π – keeping in mind that the delay management strategy is fixed – we can define the following reduced version of the overall problem (RPT).

Definition 12. *Assuming the no-wait strategy, problem* (RPT) *reduces to*

$$\min_{\pi \in \Pi} F\text{-}WC(\pi). \tag{F-RPT}$$

which can be reformulated as

$$\begin{aligned} \min \quad & t && && \text{(F-RPT}(\mathcal{S})\text{)} \\ s.t. \quad & t \geq \tau(\pi, d_s) && \forall s \in \mathcal{S}, \\ & d_s \in \mathcal{D}'(\pi, s) && \forall s \in \mathcal{S}, \\ & \pi \in \Pi,\ t \in \mathbb{R}. \end{aligned}$$

If $|\mathcal{S}|$ is finite, then (F-RPT) is a linear mixed-integer minimization problem, which is known to be solvable. If $|\mathcal{S}|$ is infinite, on the other hand, (F-RPT) can be solved via cutting planes, as illustrated in Algorithm 1 and Figure 2. A thorough investigation of cutting plane algorithms for min-max problems and convergence proofs are given in [PS18]. The authors also propose speed-up techniques for the cutting plane approach making use of the fact that the single robustification and pessimization steps do not necessarily need to be solved to optimality. We used some of these insights in our implementation and will explain them further in Section 5.

Algorithm 1: Cutting Plane Approach

Input: EAN $(\mathcal{E}, \mathcal{A})$, nominal scenario $s_{\text{nom}} \in \mathcal{S}$, stopping criterion $\epsilon > 0$
Output: ϵ-optimal solution $\pi \in \Pi(\mathcal{E}, \mathcal{A})$ to (F-RPT), upper bound ub_k to (RPT)

$\mathcal{S}_0 \leftarrow \{s_{\text{nom}}\}$, $k \leftarrow 0$, $lb_0 \leftarrow -\infty$, $ub_0 \leftarrow \infty$
while $ub_k - lb_k > \epsilon$ **do**
  $(\pi_k, d) \leftarrow$ solution to F-RPT$(\mathcal{S}_k)$
  $lb_{k+1} \leftarrow \tau(\pi_k, d)$
  $(s_k, d) \leftarrow$ solution to F-WC(π_k)
  $ub_{k+1} \leftarrow \min(ub_k, \tau(\pi_k, d))$
  $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_k \cup \{s_k\}$
  $k \leftarrow k + 1$
end
return π_k, ub_k

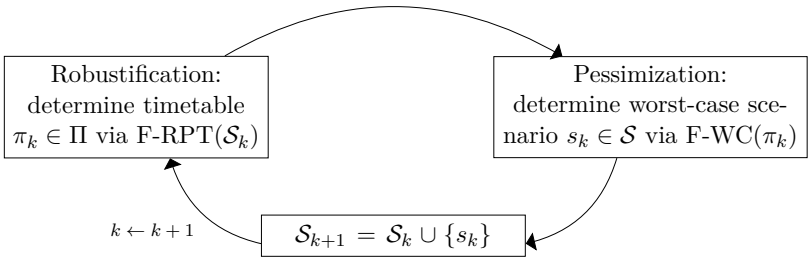


Figure 2: Cutting Plane Approach for Simplification A

With (F-RPT) we have found a simplification of (RPT), which is not only solvable, but also yields an upper bound for (RPT).

Lemma 13. (F-RPT) *is an upper bound for* (RPT).

Proof. It holds that

$$\begin{aligned}(\text{F-RPT}) \Leftrightarrow \min_{\pi\in\Pi}(\text{F-WC}(\pi)) \Leftrightarrow \min_{\pi\in\Pi}\max_{s\in\mathcal{S}}\min_{d\in\mathcal{D}'(\pi,s)}\tau(\pi,d)\\ \leq \min_{\pi\in\Pi}\sup_{s\in\mathcal{S}}\min_{d\in\mathcal{D}(\pi,s)}\sum_{a\in\mathcal{A}}\tau(\pi,d) \Leftrightarrow (\text{RPT}).\end{aligned}$$

□

Hence we have found a relaxation of (RPT). Before investigating it computationally in Section 5, we focus on the second approach that gives us a lower bound on (RPT).

4.2 Simplification B: Finding Bad-Case Scenarios

If we do not want to restrict ourselves to a fixed delay management strategy as in Simplification A, we can alternatively shrink the space $\mathcal{S}$ and consider only a finite number of scenarios. This problem yields a mixed-integer program:

Lemma 14. *RPT($\mathcal{S}'$) with finite $|\mathcal{S}'|$ is a mixed-integer program.*

Proof. We start by rewriting

$$\begin{aligned}(\text{RPT})(\mathcal{S}') \Leftrightarrow \min_{\pi\in\Pi}\max_{s\in\mathcal{S}'}\min_{d\in\mathcal{D}(\pi,s)}\tau(\pi,d) \Leftrightarrow \quad &\min && t && && (*)\\ &\text{s.t.} && t \geq \min_{d\in\mathcal{D}(\pi,s)}\tau(\pi,d) && \forall s\in\mathcal{S}',\\ & && \pi\in\Pi, &&\\ & && t\in\mathbb{R}, &&\end{aligned}$$

and by creating $|\mathcal{S}'|$ duplicates of d-variables we get

$$\begin{aligned}(*) \Leftrightarrow \quad &\min && t && \\ &\text{s.t.} && t \geq \tau(\pi,d_s) && \forall s\in\mathcal{S}',\\ & && d_s\in\mathcal{D}(\pi,s) && \forall s\in\mathcal{S}',\\ & && \pi\in\Pi, &&\\ & && t\in\mathbb{R}. &&\end{aligned}$$

Linearity of π and d_s in τ and $\mathcal{D}$ yield the statement. □

Solving this problem furthermore gives a lower bound on (RPT).

Lemma 15. (RPT)*($\mathcal{S}'$) is a lower bound on* (RPT)*($\mathcal{S}$) if $\mathcal{S}'\subseteq\mathcal{S}$.*

Proof. Fix a $\pi\in\Pi$. We get

$$\max_{s\in\mathcal{S}'}\min_{d\in\mathcal{D}(\pi,s)}\tau(\pi,d) \leq \max_{s\in\mathcal{S}}\min_{d\in\mathcal{D}(\pi,s)}\tau(\pi,d),$$

and since this holds for each $\pi\in\Pi$ the above result is true. □

The basic idea of our proposed solution algorithm to this second simplification is quite similar to the cutting plane algorithm in the previous subsection: We solve a relaxed version of (RPT), namely (RPT) on the finite scenario set $\mathcal{S}_k \subset \mathcal{S}$. Once we are given a solution π_k to this problem, we try and find a new scenario s_k. Desired properties for this scenario are that it is not contained in $\mathcal{S}_k$ and that (P-DM) on (π, s_k) yields a worse objective value than (P-DM) on (π, s) for any $s \in \mathcal{S}_k$. As mentioned earlier in this paper, an exact algorithm for determining a worst-case scenario has not yet been found. In our implementation, we will try to find worst-case scenarios by sampling many scenarios and solving (P-DM) on each of them. This idea is summarized in Algorithm 2 and Figure 3.

Algorithm 2: Iterative Improvement Heuristic

Input: EAN $(\mathcal{E}, \mathcal{A})$, nominal scenario $s_{\text{nom}} \in \mathcal{S}$, number of iterations $N \in \mathbb{N}$, number of sampled scenarios $M \in \mathbb{N}$

Output: Optimal Solution π_k to (RPT)$(\mathcal{S}_k)$ with $\mathcal{S}_k \subseteq \mathcal{S}$, lower bound lb to (RPT)

$\mathcal{S}_1 \leftarrow \{s_{\text{nom}}\}$
for $k = 1, \ldots, N$ **do**
  $(\pi_k, d^*) \leftarrow$ solution to RPT$(\mathcal{S}_k)$
  $lb \leftarrow \tau(\pi_k, d^*)$
  $ub \leftarrow 0$
  for $i = 1, \ldots, M$ **do**
    $s \leftarrow$ sampled from $\mathcal{S}$
    $d \leftarrow$ solution to (P-DM)(π_k, s)
    if $\tau(\pi_k, d) > ub$ *and* $s_k \notin \mathcal{S}_k$ **then**
      $ub \leftarrow \tau(\pi_k, d)$
      $s_k \leftarrow s$
    end
  end
  $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_k \cup \{s_k\}$
end
return π_k, lb

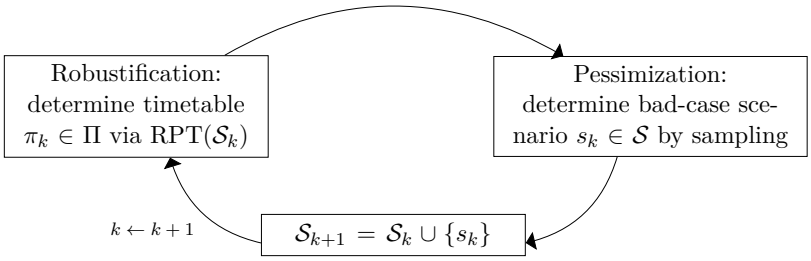


Figure 3: Iterative Approach for Simplification B

Algorithm 2 hence gives a lower bound on (RPT), but its convergence to the optimal solution of (RPT) is not guaranteed. The computational experiments show, however, that this scheme – despite being a heuristic – already yields good results.

5 Computational Experiments

Summarizing the previous chapter, we can conclude that instead of solving (RPT) we can

(A) Fix a delay management strategy and solve (F-RPT) via Algorithm 1,

(B) Find bad-case scenarios and iteratively solve (RPT)($\mathcal{S}'$) with increasing $\mathcal{S}'$, i.e. Algorithm 2.

In the following we describe the setup for determining and evaluating the robust timetables found by (F-RPT) and RPT($\mathcal{S}'$), and compare them against timetables found by MATCH, i.e., a strong heuristic for solving (PESP) which has been introduced in [PS16]. MATCH works by setting the buffer times of all waiting and driving times to zero and then heuristically merging line clusters by setting the time differences between them.
Our experiments are carried out on three different datasets which vary in size. *toy* is a small artificial dataset, consisting of 8 stops and 8 edges between them. The *grid* dataset is a 5 $\times$ 5 grid network with 40 edges that was created as a simplified version of the Stuttgart bus network. The final and biggest dataset, *bahn*, consists of 250 stations and 326 edges between them, and represents the ICE network of Germany. Further specifications of the datasets' resulting periodic EAN as well as the parameters of their respective uncertainty sets (cf. Definition 4) are given in Table 1. The parameter *passenger cutoff* is used for tractability reasons: The complexity of an EAN is highly influenced by the number of cycles it contains. Every changing activity produces a new cycle but does not influence the feasibility of a timetable (if they have no upper bound, as in our instances). Hence changing activities can be dropped in order to solve the model faster at the price of an inaccurate objective function (cf. [GL17] for details). The parameter *passenger cutoff* specfies that all changing activities having *passenger cutoff* or less passengers will be dropped, i.e., ignored by Algorithms 1 and 2. Nevertheless, these changing activities are, of course, considered for the evaluation in Figure 4 and also for plotting upper bounds in Figure 5.

	stops	edges	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}\rvert$	passengers	ρ	σ	passenger cutoff
toy	8	8	88	81	2622	5	50	0
grid	25	40	260	363	2546	5	100	10
bahn	250	326	4872	6925	385868	10	5000	300

Table 1: Dataset Specifications

We test the three different algorithms according to the workflow in Figure 4.
Consider the following notes to this workflow:

1. A timetable is retrieved by either using MATCH, Algorithm 1 or Algorithm 2. For Algorithm 1 we additionally insert a maximal number of 20 iterations as well as a time limit of 60 second for every robustification and every pessimization step. For Algorithm 2 we set the maximal number of iterations also to 20. For retrieving a bad-case scenario for a timetable π we sample 100 scenarios on the periodic EAN and solve each (P-DM) with a time limit of 10 seconds.

2. For the rollout we choose a time horizon of 8 hours. We hence generate an aperiodic EAN according to Definition 3, where the difference $U - L$ is set to 8 hours.

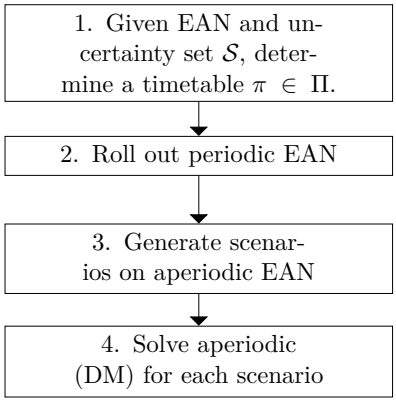


Figure 4: Workflow for testing the different timetabling algorithms

3. We generate 10 different scenarios for each instance. For generating delays on the aperiodic EAN we, accordingly, multiply ρ by the number of time periods (in our case 8) used for the rollout. Furthermore we only generate scenarios where the maximal sum of delays is really attained, meaning that our uncertainty set actually looks like

$$\mathcal{S} = \left\{ d \in \mathbb{R}^{|\mathcal{A}^{dw}|+|\mathcal{E}|} | 0 \leq d_i \leq \sigma, \sum_{i \in \mathcal{A}^{dw} \cup \mathcal{E}} d_i = \rho \tfrac{U-L}{T} \right\}.$$

4. With the aperiodic EAN we create for each scenario an instance of (DM) and solve it. We define the two evaluation criteria

$$\text{Nominal Travel Time} \coloneqq \frac{\sum_{a \in \mathcal{A}} \pi_a w_a}{\#\text{ passengers} \cdot \frac{U-L}{T}},$$

$$\text{Delayed Travel Time} \coloneqq \frac{\tau(\pi, d)}{\#\text{ passengers} \cdot \frac{U-L}{T}},$$

where the number of passengers is multiplied with the number of time periods in order to take into account the event and activity duplications in the rollout (cf. Definition 3).

We then implement Algorithms 1 and 2 in LinTim, a software framework for public transport planning (see [SAP+18]) using Python 3.7 and Gurobi 8.1. The tests are carried out on a standard notebook with 16 GB RAM and an Intel Core i5 processor (2× 2,3GHz).
Returning to the speed-up techniques for cutting-plane algorithms that are presented in [PS18] we make the following specifications: After preliminary testing, we set an objective value stopping criterion for F-WC(π_k) of ub_k (i.e., the solver can return the current solution if its objective exceeds ub_k). For F-RPT($\mathcal{S}_k$) we decide to specify a time limit instead of an objective value stopping criterion, and for Algorithm 2 we equip (P-DM) with an objective value stopping criterion of ub, which sped up the sampling process significantly as it quickly discards "good-case" scenarios.

After running the algorithms on all generated scenarios we got 10 different values for *Delayed Travel Time* for each algorithm-instance pair which are summarized by only considering minimum, maximum and average value of *Delayed Travel Time*. We retrieve the results in Table 3 which are all given in minutes.

	MATCH	(F-RPT)	RPT($\mathcal{S}'$)
toy	6.0	9.1	7.1
grid	20.4	24.6	24.1
bahn	166.4	177.6	185.8

Table 2: Nominal Travel Times in Minutes

toy	MATCH	(F-RPT)	RPT($\mathcal{S}'$)
Minimum	12.8	11.0	10.0
Maximum	14.6	11.8	12.1
Average	13.4	11.3	11.0

grid	MATCH	(F-RPT)	RPT($\mathcal{S}'$)
Minimum	29.3	26.9	27.8
Maximum	30.5	27.4	28.5
Average	29.9	27.3	28.3

bahn	MATCH	(F-RPT)	RPT($\mathcal{S}'$)
Minimum	205.0	204.0	201.7
Maximum	206.5	205.4	202.7
Average	205.9	204.6	202.1

Table 3: Delayed Travel Times in Minutes

Clearly, it can be seen that MATCH yields the best nominal travel times on all three instances. However, after carrying out delay management the travel times of MATCH are the worst among the three algorithms for all instances. F-RPT has worse nominal travel times than MATCH, but when taking Delay Management into account the travel times are up to 15% better than the travel times of MATCH. RPT($\mathcal{S}'$) outperforms F-RPT on *toy* and *bahn* with respect to the delayed travel times, but on *grid* (F-RPT) has better times after DM. We can hence see that our proposed algorithms behave better delayed travel times by paying the price of worse nominal travel times. There are at least two reasons to accept this trade-off to the detriment nominal travel times: The first reason is that in our scenario we only specified realistic delay scenarios that can occur every day: Every single source delay is bounded to be at most 5 (or 10) minutes and also the sum of all source delays was chosen to be reasonable and not extremely high. Thus, only considering the nominal scenario of no delays almost never occurs and hence should not be given so much importance. It is instead more important to find a timetable that has the property of being able to cope with "small" delays such as those specified in the uncertainty set. The second reason is that the propagated delays for MATCH are much larger than for F-RPT or RPT($\mathcal{S}'$). Considering *bahn*, MATCH yields an average delay of about 40 minutes per passenger (*Average Delayed Travel Time* minus *Nominal Travel Time*), whereas F-RPT yields 27 minutes and RPT($\mathcal{S}'$) only 16 minutes, cf. Tabular 4.

	MATCH	(F-RPT)	RPT($\mathcal{S}'$)
toy	7.4	2.2	3.9
grid	9.5	2.7	4.2
bahn	39.5	27.0	16.3

Table 4: Average Passenger Delay in Minutes

It is arguable that the size of the propagated delays has even higher importance than the Nominal Travel Time: As mentioned in the introduction, people complain much more about delays and not so much about long travel times. Hence, it would even be reasonable to assign the delays with an additional weight factor in order to model these preferences which reinforces the quality of the timetables found by F-RPT and RPT($\mathcal{S}'$). Thus, traditional periodic timetabling chooses a timetable according to Table 2, whereas it makes more sense to make the choice based on the values of Table 3. In order to avoid passenger complaints, the decision can even be based on the values of Table 4. Note that the consideration of the different timetable objective leads to a discussion on the compatibility of different evaluation functions which is analyzed in [HSF+19].
Another interesting observation is the improvement of the bounds of the algorithms for models (F-RPT) and RPT($\mathcal{S}'$). The bounds depict the values of lb_k and ub_k from Algorithm 1 for (F-RPT) and the values of lb and ub from Algorithm 2 for RPT($\mathcal{S}'$), each divided by (# passengers) to get average travel times. Note that ub of RPT($\mathcal{S}'$) can lie below lb (as for the *toy* example) since it is not a real upper bound as Algorithm 2 determines it by scenario sampling.

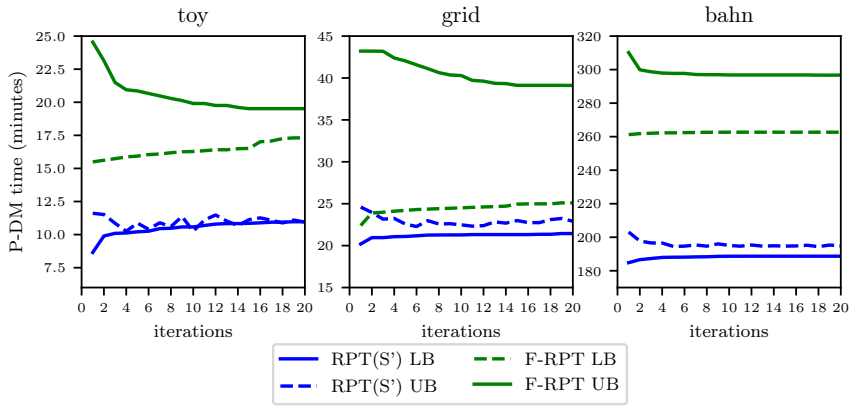


Figure 5: Solution improvements

Unfortunately, we see that the value of (RPT), which has to lie somewhere between the lower bound of RPT($\mathcal{S}'$) and the upper bound of (F-RPT), cannot be pinned down to a small interval. Nevertheless, one interesting observation is there to be made: We started with the timetable found by MATCH in all instances, meaning that the upper bound in every first iteration corresponds to the model's evaluation of the MATCH timetable. Interestingly, there is a significant decrease in the upper bound from iteration 1 to iteration 2 in all instances

for both models (except for (F-RPT) on *grid*). Hence, the models find quite bad scenarios for the MATCH timetable, leading to the conclusion that MATCH generates actually a quite non-robust timetable. Furthermore it can be seen that after 20 iterations the bounds do not change so much anymore. This is especially true for *bahn* and can be explained by the sheer size of the resulting problem. Thus, the models, despite being a simplification of (RPT), are still computationally challenging and there might also be room for improvements regarding algorithm design and parameter tuning. Still, the timetables we have found already yield an improvement with respect to robustness in comparison to MATCH.
Finally, if we compare the values for travel times found by our algorithms against the travel times estimated by the workflow we can see that the estimated travel times fall between the lower bounds for RPT($\mathcal{S}'$) and upper bounds for (F-RPT), but much closer to the values of RPT($\mathcal{S}'$). Hence, we can deduce that after a few iterations RPT($\mathcal{S}'$) yields a good approximation of the delayed travel times, whereas (F-RPT) provides merely an upper bound and is not very close to the values of the evaluated timetables in Table 3. We conclude that even though the two algorithms introduced in this paper may not serve to give precise bounds on (RPT), but they reliably find robust timetables.

6 Conclusion and Outlook

In this paper we have introduced a new model (RPT) for finding robust periodic timetables. Due to its high difficulty we introduced two relaxed versions of model (RPT), namely (F-RPT), which assumes a fixed delay management strategy, and RPT($\mathcal{S}'$), which solves the robust timetabling problem for a finite uncertainty set. For each of the two models we presented a solution algorithm, i.e., Algorithm 1 for (F-RPT) and Algorithm 2 for RPT($\mathcal{S}'$) which we tested computationally against MATCH, a standard (PESP) algorithm. The computational experiments show that for the timetables found by MATCH even small delays can induce huge passenger delays, raising concerns about the point of finding timetables that merely perform well in the absence of delays. Our proposed models, on the other hand, can cope significantly better with delays, but have worse nominal travel times – a trade-off that seems reasonable when considering customer opinions of public transport systems.
Further research can be carried out in different directions. First and foremost we can further improve the proposed solution algorithms: Obviously, there are a number of additional parameters involved in the implementation that can be optimized (or learned). What is more, hybrid strategies of the two algorithms are also possible to implement and might improve the results. Different relaxation strategies, for example by dualizing the linear relaxation of (P-DM), could also yield to strong solution algorithms. In general, it is possible to implement different delay management strategies instead of the no-wait strategy, as long as the new strategy can be formulated in the integer program. It would be very interesting to investigate the outcome of a different delay management strategy. We could, of course, also add more detail to the model. Adding vehicle schedules by using turnaround activities in the EAN could be an interesting starting point. These turnaround activities would, however, be problematic for the problem (F-WC(π)) as delays also would propagate along turnaround activities. Hence we would get no events $\mathcal{E}^{\text{start}}$ and would have to find a different formulation to get rid of the inner minimization problem in order to simplify (RPT). On the theoretical side it would be interesting to analyze the periodic delay management model (P-DM) in more detail: Especially, finding a concrete bound between the two models (DM) and (P-DM)

could further fortify the idea of considering delay management on periodic networks. Finally, defining a slightly different problem might also be a good way to craft a robustness measure: In (RPT) it might be better not to use the maximum among all scenarios but to sum over all scenarios in order to get a better robustness measure instead. The advantage would be that the maximum vanishes and (RPT) would reduce to a quite difficult minimization problem.
All things considered, there are many ways to continue the proposed path of finding robust periodic timetables by including delay management. We hope that it will become more popular to consider potential delays and not only nominal travel times when planning a periodic timetable.

References

[BBC11] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[BBV+17] S. Burggraeve, S. Bull, P. Vansteenwegen, and R. Lusby. Integrating robust timetabling in line plan optimization for railway systems. *Transportation Research Part C: Emerging Technologies*, 77:134–160, 2017.

[BEN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

[BGG+04] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[BHK+18] Ralf Borndörfer, Heide Hoppmann, Marika Karbstein, and Niels Lindner. Separation of Cycle Inequalities in Periodic Timetabling. eng. Technical report 18-16, ZIB, Takustr. 7, 14195 Berlin, 2018.

[BHK17] Ralf Borndörfer, Heide Hoppmann, and Marika Karbstein. Passenger routing for periodic timetable optimization. *Public Transport*, 9(1-2):115–135, 2017.

[BLR19] Ralf Borndörfer, Niels Lindner, and Sarah Roth. A Concurrent Approach to the Periodic Event Scheduling Problem. eng. Technical report 19-07, ZIB, Takustr. 7, 14195 Berlin, 2019.

[CCF12] Valentina Cacchiani, Alberto Caprara, and Matteo Fischetti. A lagrangian heuristic for robustness, with an application to train timetabling. *Transportation Science*, 46(1):124–133, 2012.

[CFL+11] Gabrio Caimi, Martin Fuchsberger, Marco Laumanns, and Kaspar Schüpbach. Periodic railway timetabling with event flexibility. *Networks*, 57(1):3–18, 2011.

[Con18] Kate Connolly. 'We are becoming a joke': Germans turn on Deutsche Bahn. December 2018. URL: `https://www.theguardian.com/world/2018/dec/20/trains-on-time-germans-deutsche-bahn-railway` (visited on 05/03/2019).

[CT12] Valentina Cacchiani and Paolo Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.

[DHL08] L. De Giovanni, G. Heilporn, and M. Labbé. Optimization models for the single delay management problem in public transportation. *European Journal of Operational Research*, 189(3):762–774, 2008.

[DHS+12] T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay management with rerouting of passengers. *Transportation Science*, 46(1):74–89, 2012.

[DHS+18] T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay propagation and delay management in transportation networks. In R. Borndörfer et al., editor, *Handbook of Optimization in the Railway Industry*. Springer, 2018.

[FSZ09] Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette. Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43(3):321–335, 2009.

[GGN+16] Philine Gattermann, Peter Großmann, Karl Nachtigall, and Anita Schöbel. Integrating passengers' routes in periodic timetabling: a sat approach. In *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[GHM+12] Peter Großmann, Steffen Hölldobler, Norbert Manthey, Karl Nachtigall, Jens Opitz, and Peter Steinke. Solving periodic event scheduling problems with sat. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 166–175. Springer, 2012.

[GL17] Marc Goerigk and Christian Liebchen. An improved algorithm for the periodic timetabling problem. In *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[Goe] Marc Goerigk. PESPlib - A benchmark library for periodic event scheduling. URL: `http://num.math.uni-goettingen.de/~m.goerigk/pesplib/` (visited on 05/03/2019).

[Goe15] M. Goerigk. Exact and heuristic approaches to the robust periodic event scheduling problem. *Public Transport*, 7(1):101–119, 2015.

[GS10] M. Goerigk and A. Schöbel. An empirical analysis of robustness concepts for timetabling. In Thomas Erlebach and Marco Lübbecke, editors, *Proceedings of ATMOS10*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 100–113, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. ISBN: 978-3-939897-20-0. DOI: `http://dx.doi.org/10.4230/OASIcs.ATMOS.2010.100`. URL: `http://drops.dagstuhl.de/opus/volltexte/2010/2753`.

[GS13] M. Goerigk and A. Schöbel. Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, 40(5):1363–1370, 2013.

[GS14] Marc Goerigk and Anita Schöbel. Recovery-to-optimality: a new two-stage approach to robustness with an application to aperiodic timetabling. *Computers & Operations Research*, 52:1–15, 2014.

[GS16] M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. In L. Kliemann and P. Sanders, editors, *Algorithm Engineering: Selected Results and Surveys*. Volume 9220, LNCS State of the Art, pages 245–279. 2016. URL: `http://arxiv.org/abs/1505.04901`.

[HSF+19] Johann Hartleb, Marie Schmidt, Markus Friedrich, and Dennis Huisman. A good or a bad timetable: do different evaluation functions agree? *ERIM Report Series*, 2019.

[KDV07] Leo G Kroon, Rommert Dekker, and Michiel JCM Vromans. Cyclic railway timetabling: a stochastic optimization approach. In *Algorithmic methods for railway optimization*, pages 41–66. Springer, 2007.

[KMH+08] Leo Kroon, Gábor Maróti, Mathijn Retel Helmrich, Michiel Vromans, and Rommert Dekker. Stochastic improvement of cyclic railway timetables. *Transportation Research Part B: Methodological*, 42(6):553–570, 2008.

[Lie07] Christian Liebchen. Periodic timetable optimization in public transport. In *Operations Research Proceedings 2006*, pages 29–36. Springer, 2007.

[LLB18] Richard M Lusby, Jesper Larsen, and Simon Bull. A survey on robustness in railway planning. *European Journal of Operational Research*, 266(1):1–15, 2018.

[LLM+09] Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization*, pages 1–27. Springer, 2009.

[LSS+10] C. Liebchen, M. Schachtebeck, A. Schöbel, S. Stiller, and A. Prigge. Computing delay-resistant railway timetables. *Computers and Operations Research*, 37:857–868, 2010.

[LTZ+17] Chao Lu, Jinjin Tang, Leishan Zhou, Yixiang Yue, and Zhitong Huang. Improving recovery-to-optimality robustness through efficiency-balanced design of timetable structure. *Transportation Research Part C: Emerging Technologies*, 85:184–210, 2017.

[Mar17] Gábor Maróti. A branch-and-bound approach for robust railway timetabling. *Public Transport*, 9(1-2):73–94, 2017.

[Nac98] Karl Nachtigall. Periodic network optimization and fixed interval timetables. *Deutsches Zentrum für Luft–und Raumfahrt, Institut für Flugführung, Braunschweig*, 1998.

[NO08] Karl Nachtigall and Jens Opitz. Solving periodic timetable optimisation problems by modulo simplex calculations. In *OASIcs-OpenAccess Series in Informatics*, volume 9. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[Odi96] Michiel A Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6):455–464, 1996.

[PBD+19] Gert-Jaap Polinder, Thomas Breugem, Twan Dollevoet, and Gábor Maróti. An adjustable robust optimization approach for periodic timetabling. 2019. URL: `http://hdl.handle.net/1765/113303`.

[Pee03] Leon LWP Peeters. *Cyclic railway timetable optimization*, number EPS-2003-022-LIS. 2003.

[PK01] Leon Peeters and Leo Kroon. A cycle based optimization model for the cyclic railway timetabling problem. In *Computer-aided scheduling of public transport*, pages 275–296. Springer, 2001.

[PLM+13] H. Petersen, A. Larsen, O. Madsen, B. Petersen, and S. Ropke. The Simultaneous Vehicle Scheduling and Passenger Service Problem. *Transportation Science*, 47(4):603–616, 2013.

[PS16] J. Pätzold and A. Schöbel. A Matching Approach for Periodic Timetabling. In Marc Goerigk and Renato Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016. ISBN: 978-3-95977-021-7. DOI: http://dx.doi.org/10.4230/OASIcs.ATMOS.2016.1. URL: http://drops.dagstuhl.de/opus/volltexte/2016/6525.

[PS18] J. Pätzold and A. Schöbel. Approximate cutting plane approaches for exact solutions to robust optimization problems. Submitted to European Journal of Operational Research (in revision), 2018.

[PSS+17] J. Pätzold, A. Schiewe, P. Schiewe, and A. Schöbel. Look-Ahead Approaches for Integrated Planning in Public Transportation. In Gianlorenzo D'Angelo and Twan Dollevoet, editors, *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–16, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. ISBN: 978-3-95977-042-2. DOI: 10.4230/OASIcs.ATMOS.2017.17. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7894.

[PSS18] Julius Pätzold, Alexander Schiewe, and Anita Schöbel. Cost-Minimal Public Transport Planning. In Ralf Borndörfer and Sabine Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, volume 65 of *OpenAccess Series in Informatics (OASIcs)*, 8:1–8:22, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. ISBN: 978-3-95977-096-5. DOI: 10.4230/OASIcs.ATMOS.2018.8. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9713.

[SAP+18] A. Schiewe, S. Albert, J. Pätzold, P. Schiewe, A. Schöbel, and J. Schulz. LinTim: An integrated environment for mathematical public transport optimization. Documentation. Technical report 2018-08, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen, 2018.

[SBK01] L. Suhl, C. Biederbick, and N. Kliewer. Design of customer-oriented dispatching support for railways. In S. Voß and J. Daduna, editors, *Computer-Aided Transit Scheduling*. Volume 505, Lecture Notes in Economics and Mathematical systems, pages 365–386. Springer, 2001.

[Sch01] A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.

[Sch06] A. Schöbel. *Optimization in public transportation. Stop location, delay management and tariff planning from a customer-oriented point of view*. Optimization and Its Applications. Springer, New York, 2006.

[Sch07] A. Schöbel. Integer programming approaches for solving the delay management problem. In *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes in Computer Science, pages 145–170. Springer, 2007.

[Sch14] M. Schmidt. *Integrating Routing Decisions in Public Transportation Problems*, volume 89 of *Optimization and Its Applications*. Springer, 2014.

[Sch17] A. Schöbel. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research C*, 74:348–365, 2017. DOI: 10.1016/j.trc.2016.11.018.

[SG13] M. Siebert and M. Goerigk. An experimental comparison of periodic timetabling models. *Computers and Operations Research*, 40(10):2251–2259, 2013.

[SMD18] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2018.

[SS10] M. Schachtebeck and A. Schöbel. To wait or not to wait and who goes first? Delay management with priority decisions. *Transportation Science*, 44(3):307–321, 2010. DOI: 10.1287/trsc.1100.0318.

[SS15a] M. Schmidt and A. Schöbel. The complexity of integrating routing decisions in public transportation models. *Networks*, 65(3):228–243, 2015.

[SS15b] M. Schmidt and A. Schöbel. Timetabling with passenger routing. *OR Spectrum*, 37:75–97, 2015.

[SU89] Paolo Serafini and Walter Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.

Acknowledgements

My first “thank you!” goes to Anita Schöbel for successfully guiding me through this project of pursuing a Ph.D., for always having an open door and open mind for new ideas, and especially for all the encouragement throughout the years.
Next, I would like to thank Anja Fischer for co-supervising me and for uncovering so many blind spots on my personal optimization map.
I want to thank Marc Goerigk for being in my thesis committee and, moreover, for your in-depth expertise and advice on timetabling and robust optimization.
Furthermore, the financial funding from the Simulation Science Center (SWZ), DFG FOR 2083 and DAAD is valued and gratefully acknowledged.
The next big “thank you!” goes to Marco for being the best office mate imaginable, for proofreading parts of this thesis, and simply for making the time spent in and outside the office so enjoyable. Great performance, after all!
Also, I would like to thank you, Sönke, for all the time spent together filled with talks, discussions, laughs, games, and challenges!
Alex and Philine, it was a pleasure traveling with you on so many business trips! Thanks for the enjoyable co-authorships and for proofreading parts of this thesis.
Thanks to the rest of the working group for creating such a nice working atmosphere filled with conversations, laughs, games and coffee. Thank you, in order of disappearance, Jonas, Mirko, Corinna, Lisa, Fabian, and Sebastian!
A special thanks goes to the Clausthal gang: Stephan, Martin, and Ferdinand, it has been fun spending time with you on so many different on- and off-topics!
Thanks to all my LinTim HiWis and interns, Felix, Florentin, Charlotte, Vitali, Tim, Anna, Benjamin, and Kim, for contributing to and hence improving LinTim.
My sincere gratitude goes to Bakhodir for introducing me to the world of competitive programming, which has already led us to so many exciting journeys!
A huge “thank you!” goes to my family and friends for the constant support and for offering so many invaluable and appreciated distractions from the crazy world of academics.
Finally, Carina, you are entitled to my utmost gratitude! Thanks for all your empathy, motivation and support! For enduring a mathematician’s eccentricities, for spotting clumsy formulations in this thesis, and for the continuous encouragement, especially in difficult times. With all my heart, thank you!