8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems

ATMOS 2008, September 18, 2008, Karlsruhe, Germany

Edited by

Matteo Fischetti

Peter Widmayer

Editors

Matteo Fischetti
DEI, Dipartimento di Ingegneria dell'Informazione
University of Padova
via Gradenigo 6/A
35131 Padova, Italy
matteo.fischetti@unipd.it

Peter Widmayer
Institut für Theoretische Informatik
Universitätstrasse 6
8092 Zürich, Switzerland
widmayer@inf.ethz.ch

ACM Classification 1998
F.2 Analysis of Algorithms and Problem Complexity, G.1.6 Optimization, G.2.2 Graph Theory, G.2.3 Applications

ISBN 978-3-939897-07-1

Published online and open access by
Schloss Dagstuhl – Leibniz-Center for Informatics GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.

Publication date
October, 2008.

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/OASIcs.ATMOS.2008.i

ISBN 978-3-939897-07-1 **ISSN 2190-6807** **http://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 2190-6807

www.dagstuhl.de/oasics

ATMOS 2008 Preface: Algorithmic Approaches for Transportation Modeling, Optimization, and Systems

Matteo Fischetti[1] and Peter Widmayer[2]

DEI, Dipartimento di Ingegneria dell'Informazione, University of Padova, Italy
matteo.fischetti@unipd.it
Institute of Theoretical Computer Science, ETH Zürich, Switzerland
widmayer@inf.ethz.ch

The 8th ATMOS workshop was held in Karlsruhe, September 18, 2008, within ALGO, a set of meetings related to algorithms. The series of ATMOS workshops, starting in Heraklion in 2001, continuing in Malaga in 2002, Budapest in 2003, Bergen in 2004, Palma de Mallorca in 2005, Zürich in 2006, and Sevilla in 2007 is by now an established series of meetings between algorithms researchers dealing with transportation problems, and practitioners, mainly from railways. The focus of ATMOS is on complex and large-scale network optimization problems that require new solution techniques and ideas from mathematical optimization and theoretical computer science. Tools and concepts are rooted in graph and network algorithms, combinatorial optimization, approximation and online algorithms, stochastic and robust optimization. Of particular interest are

- Infrastructure Planning
- Line Planning
- Timetable Generation
- Routing and Platform Assignment
- Vehicle Scheduling
- Crew and Duty Scheduling
- Rostering
- Demand Forecasting
- Design of Tariff Systems
- Maintenance and Shunting of Rolling Stock
- Delay Management
- Rolling Stock Rescheduling
- Simulation Tools for Railway Operations
- Timetable Information

More generally, ATMOS aims at the successful integration of several of these subproblems or planning stages, algorithms operating in an

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1593

online/realtime or stochastic setting and heuristic or approximate algorithms for real-world instances.

We received 15 submissions, out of which 12 were selected for presentation and inclusion in this volume, in a thorough reviewing process guided by the program committee consisting of

- Cynthia Barnhart, MIT
- Ralf Borndörfer, Zuse Institute Berlin
- Alberto Caprara, University of Bologna
- Jens Clausen, Technical University of Denmark
- Guy Desaulniers, GERAD, Ecole Polytechnique Montréal
- Matteo Fischetti, University of Padova (Co-Chair)
- Leo Kroon RSM Erasmus University and Netherlands Railways
- Marc Nunkesser, ETH Zürich
- Anita Schöbel, University of Göttingen
- Dorothea Wagner, University of Karlsruhe
- Peter Widmayer, ETH Zürich (Co-Chair)

In addition, Rolf Möhring gave an invited talk on "Timetabling and Robustness: Computing Good and Delay-Resistant Timetables".

We sincerely thank the program committee for the competent work in selecting the best papers and the external referees for their help, the organizer Marc Nunkesser for taking care of all arrangements, the ALGO organizing committee for embedding ATMOS so smoothly into the ALGO programme, the editors of the Dagstuhl Seminar Proceedings for accepting the publication of this volume within DROPS, and, last but not least, the participants for the lively interaction that is the ultimate goal of the meeting.

Padova and Zürich, October 2008

Matteo Fischetti and Peter Widmayer

ATMOS 2008 - Abstracts Collection
Selected Papers from the 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems

Matteo Fischetti[1] and Peter Widmayer[2]

University of Padova, IT
and ETH Zürich, CH

Abstract. Proceedings of the 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, held on Septmeber 18 in Karlsruhe, Germany.

ATMOS 2008 Preface – 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems

The 8th ATMOS workshop was held in Karlsruhe, September 18, 2008, within ALGO, a set of meetings related to algorithms. The series of ATMOS workshops, starting in Heraklion in 2001, continuing in Malaga in 2002, Budapest in 2003, Bergen in 2004, Palma de Mallorca in 2005, Zürich in 2006, and Sevilla in 2007 is by now an established series of meetings between algorithms researchers dealing with transportation problems, and practitioners, mainly from railways.

Joint work of: Fischetti, Matteo; Widmayer, Peter

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1593

Dynamic Algorithms for Recoverable Robustness Problems

Recently, the recoverable robustness model has been introduced in the optimization area. This model allows to consider disruptions (input data changes) in a unified way, that is, during both the strategic planning phase and the operational phase. Although the model represents a significant improvement, it has the following drawback: we are typically not facing only one disruption, but many of them might appear one after another. In this case, the solutions provided in the context of the recoverable robustness are not satisfying. In this paper we extend the concept of recoverable robustness to deal not only with one single recovery step, but with arbitrarily many recovery steps. To this aim, we introduce the notion of dynamic recoverable robustness problems. We apply the new model in the context of timetabling and delay management problems. We are interested in finding efficient dynamic robust algorithms for solving the timetabling problem and in evaluating the price of robustness of the proposed solutions.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1592

Keywords: Robustness, optimization problems, dynamic algorithms, timetabling, delay management

Joint work of: Cicerone, Serafino; Di Stefano, Gabriele; Schachtebeck, Michael; Schöbel, Anita

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1587

Efficient On-Trip Timetable Information in the Presence of Delays

The search for train connections in state-of-the-art commercial timetable information systems is based on a static schedule. Unfortunately, public transportation systems suffer from delays for various reasons. Thus, dynamic changes of the planned schedule have to be taken into account. A system that has access to delay information of trains (and uses this information within search queries) can provide valid alternatives in case a train change breaks. Additionally, it can be used to actively guide passengers as these alternatives may be presented before the passenger is already stranded at a station due to a broken transfer. In this work we present an approach which takes a stream of delay information and schedule changes on short notice (partial train cancellations, extra trains) into account. Primary delays of trains may cause a cascade of so-called secondary delays of other trains which have to wait according to certain waiting policies between connecting trains. We introduce the concept of a dependency graph to efficiently calculate and update all primary and secondary delays. This delay information is then incorporated into a time-expanded search graph which has to be updated dynamically. These update operations are quite complex, but turn out to be not time-critical in a fully realistic scenario. We finally present a case study with data provided by Deutsche Bahn AG showing that this approach has been successfully integrated into our multi-criteria timetable information system MOTIS and can handle massive delay data streams instantly.

Keywords: Timetable information system, primary and secondary delays dependency graph, dynamic graph update

Joint work of: Frede, Lennart; Müller-Hannemann, Matthias; Schnee, Mathias

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1584

Engineering Time-Expanded Graphs for Faster Timetable Information

We present an extension of the well-known time-expanded approach for timetable information. By remodeling unimportant stations, we are able to obtain faster query times with less space consumption than the original model. Moreover, we show that our extensions harmonize well with speed-up techniques whose adaption to timetable networks is more challenging than one might expect.

Keywords: Timetable information, shortest path, modeling

Joint work of: Delling, Daniel; Pajor, Thomas; Wagner, Dorothea

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1582

Integrated Gate and Bus Assignment at Amsterdam Airport Schiphol

At an airport a series of assignment problems need to be solved before aircraft can arrive and depart and passengers can embark and disembark. A lot of different parties are involved with this, each of which having to plan their own schedule. Two of the assignment problems that the 'Regie' at Amsterdam Airport Schiphol (AAS) is responsible for, are the gate assignment problem (i.e. where to place which aircraft) and the bus assignment problem (i.e. which bus will transport which passengers to or from the aircraft). Currently these two problems are solved in a sequential fashion, the output of the gate assignment problem is used as input for the bus assignment problem. We look at integrating these two sequential problems into one larger problem that considers both problems at the same time. This creates the possibility of using information regarding the bus assignment problem while solving the gate assignment problem. We developed a column generation algorithm for this problem and have implemented a prototype. To make the algorithm efficient we used a special technique called stabilized column generation and also column deletion. Computational experiments with real-life data from AAS indicate that our algorithm is able to compute a planning for one day at Schiphol in a reasonable time.

Keywords: Gate assignment, airports, integrated planning, column generation, integer linear programming

Joint work of: Diepen, Guido; van den Akker, Marjan; Hoogeveen, Han

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1591

IP-based Techniques for Delay Management with Priority Decisions

Delay management is an important issue in the daily operations of any railway company. The task is to update the planned timetable to a disposition timetable in such a way that the inconvenience for the passengers is as small as possible. The two main decisions that have to be made in this respect are the wait-depart decisions to decide which connections should be maintained in case of delays and the priority decisions that determine the order in which trains are allowed to pass a specific piece of track. They later are necessary in the capacitated case due to the limited capacity of the track system and are crucial to ensure that the headways between different trains are respected and that single-track traffic is

routed correctly. While the wait-depart decisions have been intensively studied in literature (e.g. [Sch06,Gat07]), the priority decisions in the capacitated case have been neglected so far in delay management optimization models. In the current paper, we add the priority decisions to the integer programming formulation of the delay management problem and are hence able to deal with the capacitated case. Unfortunately, these constraints are disjunctive constraints that make the resulting event activity network more dense and destroy the property that it does not contain any directed cycle. Nevertheless, we are able to derive reduction techniques for the network which enable us to extend the formulation of the never-meet property from the uncapacitated delay management problem to the capacitated case. We then use our results to derive exact and heuristic solution procedures for solving the delay management problem. The results of the algorithms are evaluated both from a theoretical and a numerical point of view. The latter has been done within a case study using the railway network in the region of Harz, Germany.

Joint work of: Schachtebeck, Michael; Schöbel, Anita

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1586

Line Planning on Paths and Tree Networks with Applications to the Quito Trolebús System

Line planning is an important step in the strategic planning process of a public transportation system. In this paper, we discuss an optimization model for this problem in order to minimize operation costs while guaranteeing a certain level of quality of service, in terms of available transport capacity. We analyze the problem for path and tree network topologies as well as several categories of line operation that are important for the Quito Trolebús system. It turns out that, from a computational complexity worst case point of view, the problem is hard in all but the most simple variants. In practice, however, instances based on real data from the Trolebús System in Quito can be solved quite well, and significant optimization potentials can be demonstrated.

Keywords: Line planning, computational complexity, optimization in transportation

Joint work of: Torres, Luis M.; Torres, Ramiro; Borndörfer, Ralf; Pfetsch, Marc E.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1583

Recoverable Robustness for Railway Rolling Stock Planning

In this paper we explore the possibility of applying the notions of Recoverable

Robustness and Price of Recoverability (introduced by [5]) to railway rolling stock planning, being interested in recoverability measures that can be computed in practice, thereby evaluating the robustness of rolling stock schedules. In order to lower bound the Price of Recoverability for any set of recovery algorithms, we consider an "optimal" recovery algorithm and propose a Benders decomposition approach to assess the Price of Recoverability for this "optimal" algorithm. We evaluate the approach on real-life rolling stock planning problems of NS, the main operator of passenger trains in the Netherlands. The preliminary results show that, thanks to Benders decomposition, our lower bound can be computed within relatively short time for our case study.

Joint work of: Cacchiani, Valentina; Caprara, Alberto; Galli, Laura; Kroon, Leo; Maróti, Gábor

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1590

Robust Line Planning under Unknown Incentives and Elasticity of Frequencies

The problem of robust line planning requests for a set of origin-destination paths (lines) along with their traffic rates (frequencies) in an underlying railway network infrastructure, which are robust to fluctuations of real-time parameters of the solution. In this work, we investigate a variant of robust line planning stemming from recent regulations in the railway sector that introduce competition and free railway markets, and set up a new application scenario: there is a (potentially large) number of line operators that have their lines fixed and operate as competing entities struggling to exploit the underlying network infrastructure via frequency requests, while the management of the infrastructure itself remains the responsibility of a single (typically governmental) entity, the network operator. The line operators are typically unwilling to reveal their true incentives. Nevertheless, the network operator would like to ensure a fair (or, socially optimal) usage of the infrastructure, e.g., by maximizing the (unknown to him) aggregate incentives of the line operators. We show that this can be accomplished in certain situations via a (possibly anonymous) incentive- compatible pricing scheme for the usage of the shared resources, that is robust against the unknown incentives and the changes in the demands of the entities. This brings up a new notion of robustness, which we call incentive-compatible robustness, that considers as robustness of the system its tolerance to the entities' unknown incentives and elasticity of demands, aiming at an eventual stabilization to an equilibrium point that is as close as possible to the social optimum.

Joint work of: Kontogiannis, Spyros; Zaroliagis, Christos

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1581

Simultaneous Network Line Planning and Traffic Assignment

One of the basic problems in strategic planning of public and rail transport is the line planning problem to find a system of lines and its associated frequencies. The objectives of this planning process are usually manifold and often contradicting. The transport operator wants to minimize cost, whereas passengers want to have travel time shortest routes without any or only few changings between different lines. The travel quality of a passenger route depends on the travel time and on the number of necessary changings between lines and is usually measured by a disutility or impedance function. In practice the disutility strongly depends on the line plan, which is not known, but should be calculated. The presented model combines line planning models and traffic assignment model to overcome this dilemma. Results with data of BerlinŠs city public transportion network are reported.

Keywords: Line planning problem, integer programming

Joint work of: Nachtigall, Karl; Jerosch, Karl

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1175

Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations

In the last 15 years periodic timetable problems have found much interest in the combinatorial optimization community. We will focus on the optimisation task to minimise a weighted sum of undesirable slack times. This problem can be formulated as a mixed integer linear problem, which for real world instances is hard to solve. This is mainly caused by the integer variables, the so-called modulo parameter. At first we will discuss some results on the polyhedral structure of the periodic timetable problem. These ideas allow to define a modulo simplex basic solution by calculating the basic variables from modulo equations. This leads to a modulo network simplex method, which iteratively improves the solution by changing the simplex basis.

Keywords: Periodic event scheduling problem, integer programming, modulo network simplex

Joint work of: Nachtigall, Karl ; Opitz, Jens

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1588

The Second Chvatal Closure Can Yield Better Railway Timetables

We investigate the polyhedral structure of the Periodic Event Scheduling Problem (PESP), which is commonly used in periodic railway timetable optimization. This is the first investigation of Chvátal closures and of the Chvatal rank of PESP instances. In most detail, we first provide a PESP instance on only two events, whose Chvatal rank is very large. Second, we identify an instance for which we prove that it is feasible over the first Chvatal closure, and also feasible for another prominent class of known valid inequalities, which we reveal to live in much larger Chvatal closures. In contrast, this instance turns out to be infeasible already over the second Chvátal closure. We obtain the latter result by introducing new valid inequalities for the PESP, the multi-circuit cuts. In the past, for other classes of valid inequalities for the PESP, it had been observed that these do not have any effect in practical computations. In contrast, the new multi-circuit cuts that we are introducing here indeed show some effect in the computations that we perform on several real-world instances - a positive effect, in most of the cases.

Joint work of: Liebchen, Christian; Swarat, Elmar

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1580

Towards Solving Very Large Scale Train Timetabling Problems by Lagrangian Relaxation

The train timetabling problem considered is to find conflict free routes for a set of trains in a given railway network so that cer- tain time window conditions are satisfied. We deal with the very large scale problem of constructing such timetables for the German railway network. A number of restrictions on different train types like freight trains or passenger trains have to be observed, e.g., sequence dependent headway times, station capacities, and stopping times. In order to handle the enormous number of variables and constraints we employ Lagrangian relaxation of the conflict constraints combined with a cutting plane approach. The model is solved by a bundle method; its primal aggregate is used for separation and as starting point for rounding heuristics. We present some promising results towards handling a test instance com- prising ten percent of the entire network.

Joint work of: Fischer, Frank; Helmberg, Christoph; Janßen, Jürgen; Krostitz, Boris

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1585

Dynamic Algorithms for Recoverable Robustness Problems*

S. Cicerone[1], G. Di Stefano[1], M. Schachtebeck[2], and A. Schöbel[2]

[1] Department of Electrical and Information Engineering, University of L'Aquila, Italy. {cicerone,gabriele}@ing.univaq.it
[2] Institute for Numerical and Applied Mathematics, Georg-August-University Göttingen, Germany. {schachte,schoebel}@math.uni-goettingen.de

Abstract. Recently, the *recoverable robustness model* has been introduced in the optimization area. This model allows to consider disruptions (input data changes) in a unified way, that is, during both the strategic planning phase and the operational phase. Although the model represents a significant improvement, it has the following drawback: we are typically not facing only one disruption, but many of them might appear one after another. In this case, the solutions provided in the context of the recoverable robustness are not satisfying.
In this paper we extend the concept of recoverable robustness to deal not only with one single recovery step, but with arbitrarily many recovery steps. To this aim, we introduce the notion of *dynamic recoverable robustness problems*. We apply the new model in the context of timetabling and delay management problems. We are interested in finding efficient dynamic robust algorithms for solving the timetabling problem and in evaluating the *price of robustness* of the proposed solutions.

Key words: Robustness, optimization problems, dynamic algorithms, timetabling, delay management.

1 Introduction

In many applications of optimization, the input data is subject to uncertainties and *disruptions* (input data changes). Thus, in most cases, it is desirable not to have a solution that is optimal for the undisturbed input data, but that is feasible even for disturbed input – at the cost of optimality.

Disruptions have to be considered both in the *strategic planning* phase and in the *operational* phase. The latter phase aims to have immediate reaction to disruptions that can occur when the system is running, while the former one aims to plan how to optimize the use of the available resources according to some objective function before the system starts operating.

To face disruptions in the operational phase, the approaches used are mainly based on the concept of *online algorithms* [5]. An online recovery strategy has

* Work partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/15871

to be developed when unpredictable disruptions in daily operations occur, and before the entire sequence of disruptions is known. The goal is to react fast, while retaining as much as possible of the quality of an optimal solution, that is, a solution that would have been achieved if the entire sequence of disruptions was known in advance.

To face disruptions in the strategic planning phase, the approaches used are mainly based on *stochastic programming* and *robust optimization*.

Within stochastic programming (e.g., see [4, 14, 16]), there are two different approaches: *chance constrained programming* aims to find a solution that satisfies the constraints in most scenarios (i.e. with a high probability) instead of satisfying them for all possible realizations of the random variables, while in *multi-stage stochastic programming*, an initial solution is computed in the first step, and each time when some new random data is revealed, a recourse action is taken. However, stochastic programming requires detailed knowledge on the probability distributions of the random variables.

In robust optimization (e.g., see [1–3, 7]), the objective – in contrast to stochastic programming – is purely deterministic. In the concept of *strict robustness*, the solution has to be feasible for all admissible scenarios from a set of input scenarios. The solution gained by this approach can then be fixed since by construction it needs not to be changed when disturbances occur. However, as the solution is fixed independently of the actual scenario, robust optimization leads to solutions that are too conservative and thus too expensive in many applications. One approach to compensate this disadvantage is the idea of *light robustness* introduced in [8]. This approach adds slacks to the constraints. A solution is considered as robust if it satisfies these relaxed constraints.

All the approaches above do not allow to consider disruptions in a unified way, that is, for both the strategic planning and the operational phases. Recently, a first contribution in this direction has been proposed in [15], where the *recoverable robustness* model has been presented. It starts from the practical point of view that a solution is robust if it can be recovered easily in case of a disturbance. This means the solution no longer has to be feasible for all possible scenarios, but a recovery phase is allowed to turn an infeasible solution into a feasible one. However, some limitations on the recovery phase have to be taken into account. For example, the recovery should be quick enough and the quality of the recovered solution should not be too bad.

The initial model of [15] has been extended and applied to shunting problems in [6]. There, the *price of robustness* is defined as the maximum ratio between the cost of the provided robust solution and the optimal solution. According to their price, robust algorithms may also be *exact* or *optimal*.

Although the recoverable robustness model represents a significant improvement in the optimization area, it has the following drawback: We are typically not facing only one disruption, but many of them might appear one after another. In this case, the solutions provided in the context of the recoverable robustness are not satisfying. In this paper we extend the concept of recoverable robustness presented in [6] to deal not only with one single recovery step, but with

arbitrarily many recovery steps. To this aim, we introduce the class DRRP(σ), $\sigma \in \mathbb{N}$, containing all the *dynamic recoverable robustness problems* which have to be solved against σ possible disruptions, appearing one by one. The model in [6] captures exactly the problems in DRRP(1), that is, the *static* recoverable robustness problems.

A concrete example of real world systems, where our model plays an important role, is the *timetable planning*. It arises in the strategic planning phase for transportation systems, and it requires to compute a timetable with e.g. minimal passenger waiting times. However, *many* disturbing events (caused by delays) might occur during the operational phase, and they might completely change the schedule. The problem of deciding which connections from a delayed train to a connecting train should be guaranteed is known in the literature as *delay management problem* [13, 17, 18]. This problem has been shown to be NP-hard in the general case, while it is polynomial solvable in particular cases (see [9–12, 17, 18]).

In this paper, we apply the recoverable robustness model in the context of timetabling and delay management problems. We are interested in finding efficient dynamic robust algorithms for solving the timetabling problem and in evaluating the price of robustness of the proposed solutions. In detail, we take two particular timetabling problems and turn them into problems in DRRP(σ) by defining specific modifications and recovery strategies. For one of such problems we show that finding a solution which minimizes the objective function of the corresponding timetable problem is NP-hard. In general, we propose dynamic robust algorithms and evaluate their prices of robustness. We also prove that such algorithms are optimal with respect to some specific instances.

The remainder of this paper is structured as follows: In Section 2, we show how the concept of recoverable robustness from [15] can be extended to the dynamic framework. Section 3 shows how this framework can be applied to the delay-resistant timetables. In Sections 4 and 5, we propose dynamic robust algorithms, evaluate their prices of robustness and prove optimality in specific instances.

Due to space limitations, some proofs have been omitted.

2 The model

In this section, we extend the model concerning robustness for optimization problems introduced in [6, 15]. We consider minimization problems P characterized by the following parameters:

- I, the set of instances of P;
- $F(i)$, the set of all (potential) feasible solutions for $i \in I$;
- $f\colon S \to \mathbb{R}^{>0}$, the objective function of P that has to be minimized, where $S = \bigcup_{i \in I} F(i)$.

In the dynamic robust optimization problem, we want to find a *robust plan* for some given initial instance $i \in I$ of P. Additional concepts to describe the robustness for the minimization problem P are needed:

- $M : I \to 2^I$ – a *modification* function for instances of P. This function models disturbances of the current scenario due to the following case. If $i \in I$ is the considered input (or scenario) of problem P, a *disturbance* is meant as a modification of i leading to another input scenario $i' \in I$. Such a modification i' depends on the current input i. In order to model this fact, we define the set $M(i)$ as the set of all instances which are modifications of the instance i, i.e. instances that can occur if i is disturbed. Note that the set of modifications may also depend on other information, e.g. on the data of the initial instance.
  Let s be the planned solution for the input i. When a disturbance $i' \in M(i)$ occurs, a new solution $s' \in F(i')$ has to be recomputed for P.
- σ – maximum number of expected modifications. In a practical scenario, several disruptions $i^1, i^2, \ldots, i^\sigma$ may occur. In this case, a task is to devise recovery algorithms that can recompute the solution for P after *each* disruption.
- $\mathbb{A}_{rec}$ – a class of *recovery algorithms* for P. Each element $A_{rec} : S \times I \to S$ works as follows: given a solution $s^0 \in S$ of P (for the current instance i^0) and a modification $i^1 \in M(i^0)$, then $A_{rec}(s^0, i^1) = s^1$, where $s^1 \in F(i^1) \subseteq S$ represents the recovered solution for P. We remark that s^0 and i^1 define the minimal amount of information necessary to recompute the solution. However, for specific cases, A_{rec} could require additional information. In general, when A_{rec} is used at the k-th step, it can use everything that has been processed in the previous steps (in particular, $i^0, ..., i^{k-1}$, and $s^0, ..., s^{k-1}$).

  In general, a class of recovery algorithms $\mathbb{A}_{rec}$ is defined in terms of some kind of *limitation*. In what follows we provide two examples for the class $\mathbb{A}_{rec}$.

  $\mathbb{A}^1_{rec}$**:** this class is based on a constraint on the solutions provided by the recovery algorithm. In particular, the new (recovered) solutions computed by an algorithm must not deviate too much from the original solution s, according to a distance measure d. Formally: given a real number $\Delta \in \mathbb{R}$ and a distance function $d : S \times S \to \mathbb{R}$, we define $\mathbb{A}^1_{rec}$ as the class of algorithms A_{rec} that satisfy the following constraint:
  - $\forall i \in I$, $\forall s \in F(i)$, $\forall i' \in M(i)$, $d(s, A_{rec}(s, i')) \leq \Delta$.

  $\mathbb{A}^2_{rec}$**:** this class is formulated by bounding the computational power of recovery algorithms. Formally: given a function $f : I \times S \times I \to \mathbb{N}$, we define $\mathbb{A}^2_{rec}$ as the class of algorithms A_{rec} that satisfy the following constraint:
  - $\forall i \in I$, $\forall s \in F(i)$, $\forall i' \in M(i)$, $A_{rec}(s, i')$ can be computed in $O(f(i, s, i'))$ time.

2.1 Static model

We first recall the basic definitions concerning robustness for optimization problems introduced in [6, 15].

Definition 1. [6] *A* recoverable robustness problem *is defined by the triple* $(P, M, \mathbb{A}_{rec})$. *All the recoverable robustness problems form the class* RRP.

Definition 2. [6] *Let* $\mathcal{P} = (P, M, \mathbb{A}_{rec})$ *be an element of* RRP. *Given an instance* $i \in I$ *for* P, *an element* $s \in F(i)$ *is a* feasible solution *for* i *with respect to* $\mathcal{P}$ *if and only if the following relationship holds:*

$$\exists A_{rec} \in \mathbb{A}_{rec} : \forall i' \in M(i),\ A_{rec}(s, i') \in F(i').$$

In other words, $s \in F(i)$ is feasible for i with respect to $\mathcal{P}$ if it can be *recovered* by applying some algorithm $A_{rec} \in \mathbb{A}_{rec}$ for each possible disruption $i' \in M(i)$. Hence, s is called *robust* solution. We define $F_{\mathcal{P}}(i)$ as the set of robust solutions for i with respect to $\mathcal{P}$.

Definition 3. [6] *Given* $\mathcal{P} = (P, M, \mathbb{A}_{rec}) \in$ RRP, *a* robust algorithm *for* $\mathcal{P}$ *is any algorithm* $A_{rob} : I \to S$ *such that, for each* $i \in I$, $A_{rob}(i)$ *is robust for* i *with respect to* $\mathcal{P}$.

2.2 Dynamic model

Here we extend the static model in order to deal with a sequence of $\sigma \geq 1$ modifications.

Definition 4. *A* dynamic recoverable robustness problem *is defined by* $(P, M, \mathbb{A}_{rec}, \sigma)$, $\sigma \in \mathbb{N}$. *The class* DRRP(σ) *contains all the problems that have to be solved against* σ *possible disruptions.*

Definition 5. *Let* $\sigma \in \mathbb{N}$ *and* $\mathcal{P} = (P, M, \mathbb{A}_{rec}, \sigma)$ *be an element of* DRRP(σ). *A pair of algorithms* (A_{rob}, A_{rec}) *is called* dynamic robust recovery pair *for* σ *and* $\mathcal{P}$ *if for each instance* $i^0 \in I$, $A_{rec} \in \mathbb{A}_{rec}$ *and the following relationships hold:*

$$s^0 := A_{rob}(i^0) \in F(i^0) \tag{1}$$

$$s^k := A_{rec}(s^{k-1}, i^k) \in F(i^k),\ \forall i^k \in M(i^{k-1}), \forall k \in [1..\sigma], \tag{2}$$

i.e. in the k*-th step, for any possible modification* $i^k \in M(i^{k-1})$ *and for any feasible solution* s^{k-1} *computed in the previous step, the output* s^k *of algorithm* A_{rec} *is a feasible solution for* i^k *with respect to* $\mathcal{P}$.

Note DRRP(1) = RRP. As a consequence, we refer to a *static problem* as a problem in DRRP(1). We use the notation $F_{\mathcal{P}}(i)$ to represent all robust solutions for i with respect to $\mathcal{P} \in$ DRRP(σ) If $\mathbb{A}_{rec}$ is the class of algorithms that never change the solution s for the input i, i.e., if each algorithm $A_{rec} \in \mathbb{A}_{rec}$ satisfies

$$\forall i \in I,\ \forall s \in S,\ \forall i' \in M(i),\ A_{rec}(s, i') = s,$$

then *dynamic recovery robustness* reduces to *strict robustness*. In this case, a robust algorithm A_{rob} for $\mathcal{P}$ must provide a solution s^0 for i^0 such that, for each possible modification $i^k \in M(i^{k-1})$, we have $s^0 \in F_{\mathcal{P}}(i^k)$ for all $k \in [1..\sigma]$. The meaning is the following: If A_{rec} has no recovery capability, then A_{rob} has to find solutions that "absorb" *any* possible sequence of disruptions.

2.3 Price of robustness

For every instance $i \in I$, the price of robustness of A_{rob} is given by the maximum ratio between the cost of the solution provided by A_{rob} and the optimal solution.

Definition 6. *The* price of robustness *of a robust algorithm A_{rob} for a problem $\mathcal{P} \in \mathrm{DRRP}(\sigma)$ is*

$$P_{rob}(\mathcal{P}, A_{rob}) = \max_{i \in I} \left\{ \frac{f(A_{rob}(i))}{\min\{f(x) : x \in F(i)\}} \right\}.$$

The price of robustness of a problem $\mathcal{P} \in \mathrm{DRRP}(\sigma)$ is given by the minimum price of robustness among all possible robust algorithms. Formally,

Definition 7. *The* price of robustness *of a problem $\mathcal{P} \in \mathrm{DRRP}(\sigma)$ is given by*

$$P_{rob}(\mathcal{P}) = \min\{P_{rob}(\mathcal{P}, A_{rob}) : A_{rob} \text{ is a robust algorithm for } \mathcal{P}\}.$$

If there are many robust algorithms possible for $\mathcal{P}$, we want to identify the "best" one:

Definition 8. *Let $\mathcal{P} \in \mathrm{DRRP}(\sigma)$ and let A_{rob} be a robust algorithm for $\mathcal{P}$. Then,*

- *A_{rob} is* exact *if $P_{rob}(\mathcal{P}, A_{rob}) = 1$;*
- *A_{rob} is* $\mathcal{P}$-optimal *if $P_{rob}(\mathcal{P}, A_{rob}) = P_{rob}(\mathcal{P})$. The solution computed by a $\mathcal{P}$-optimal algorithm is called $\mathcal{P}$-optimal.*

Note that the definition of an exact robust algorithm refers to the problem $\mathcal{P}$. We state a simple observation concerning the price of robustness.

Lemma 1. *For fixed P, M, and $\mathbb{A}_{rec}$, consider a family of problems $\mathcal{P}_\sigma = (P, M, \mathbb{A}_{rec})$ for different values of σ, i.e. these problems vary in the expected number of recoveries only. For $\sigma_1 < \sigma_2$, we have*

- *$F_{\mathcal{P}_{\sigma_2}}(i) \subseteq F_{\mathcal{P}_{\sigma_1}}(i)$ for all instances $i \in I$,*
- *$P_{rob}(\mathcal{P}_{\sigma_1}) \leq P_{rob}(\mathcal{P}_{\sigma_2})$, i.e. the price of robustness grows in the number of expected recoveries.*

Proof. Let $s \in F_{\mathcal{P}_{\sigma_2}}$, i.e. there exist (A_{rob}, A_{rec}) such that (2) holds for all $k = 1, \ldots, \sigma_2$, hence also for all $k = 1, \ldots, \sigma_1$. This yields $s \in F_{\mathcal{P}_{\sigma_1}}$. Moreover, it shows that A_{rob} is robust for $\mathcal{P}_{\sigma_1}$ if A_{rob} is robust for $\mathcal{P}_{\sigma_2}$, hence also the second statement holds. □

3 Application to Delay Management

In this section we apply the concept of dynamic robustness to a (simplified variant) of the *timetabling* problem in public transportation. Timetabling is a real-world problem which suffers a lot from disturbances or delays. Whenever an unexpected source delay occurs, the timetable has to be recovered to a *disposition timetable*. Recovering a given timetable in case of delays is known as the *delay management problem*. Usually, there is not only one (source) delay but many such delays during a day which occur one after another. Our concept of *dynamic recoverable robustness* is hence important for this application.

3.1 Notation and Definition

We first introduce the specific timetabling problem which we will consider, and describe the delay scenarios and the restrictions for the recovery we are looking at. We then investigate the recovery restrictions in detail.

The Initial Planning Problem. Let $G = (V, A)$ be a directed acyclic graph (DAG) with one specified node v_1 such that there exists a directed path from v_1 to each other node.

Referring to the notation common in timetabling in public transportation, let us call the nodes in V *events* and the edges A *activities*. The nodes refer to arrival and departure events and the activities to driving, waiting and changing activities. Moreover, we say a DAG $G = (V, A)$ is a *tree*, if it is an out-tree to some source node v_1 (i.e. the path from v_1 to each other node is unique), and a *linear graph* if $V = \{v_1, v_2, \ldots, v_n\}$ and $A = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$.

In order to define the timetabling problem, let us assume that we have given weights $w_u \in \mathbb{R}$ for each $u \in V$ representing the importance of the corresponding event and lower bounds $L_a^0 \in \mathbb{R}^{>0}$ indicating the minimal duration that is needed for activity $a \in \mathcal{A}$.

We are looking for a timetable $\pi : V \to \mathbb{R}^{\geq 0}$ assigning a point of time to each event $u \in V$. π is feasible if it respects the minimal duration of each activity (see (3)), i.e. our initial planning problem P can be stated as

$$(P) \quad \min f(\pi) = \sum_{u \in V} w_u \pi_u \qquad \text{s.t.}$$

$$\pi_v - \pi_u \geq L_a^0 \quad \forall a = (u, v) \in A \tag{3}$$

$$\pi_u \in \mathbb{R}^{\geq 0} \quad \forall u \in V. \tag{4}$$

An instance i of P is specified by $i = (G, w, L^0)$. Given $a = (u, v) \in A$, the amount $\pi_v - \pi_u - L_a$ is called *slack time* for the activity a. (P) can be solved in polynomial time by linear programming. We will consider two special cases of (P), both having the same constraints (3) and (4), but differ in their objective functions.

P1 Here we consider the objective which is usually used in timetabling, namely to minimize the sum of all activity lengths (or equivalently to minimize the slack times). The objective of this problem is

$$(P1) \quad \min f(\pi) = \sum_{a=(u,v) \in \mathcal{A}} w_a(\pi_v - \pi_u),$$

with $w_a \in \mathbb{R}^{\geq 0}$ for all $a \in \mathcal{A}$. It is a special case of (P), namely if $w_u = \sum_{a=(v,u) \in \mathcal{A}} w_a - \sum_{a=(u,v) \in \mathcal{A}} w_a$ for each $u \in V$.

P2 Here we consider the problem (P) but require $w_u \geq 0$ for all $u \in V$.

Note that (P2) can be efficiently solved by the forward phase of the critical path method of project planning (CPM): given an instance $i = (G, w, L^0)$ of (P2), the solution $\pi = CPM(i)$ for i can be computed as follows:

$$CPM(i) = \begin{cases} \pi_v = 0 & \text{if } v = v_1 \\ \pi_v = \max\{\pi_u + L_a \ : \ a = (u,v) \in A\} & \text{otherwise} \end{cases}$$

The following lemmata can be proven by induction.

Lemma 2. *Given an instance $i = (G, w, L^0)$ of (P1), $CPM(i)$ is optimal if G is a tree.*

Note that this needs not to be true if G is not a tree: It often makes sense to schedule events that do not belong to the critical path later than necessary to avoid slack on activities with high weights w_a.

Lemma 3. *Given an instance $i = (G, w, L^0)$ of (P2), $CPM(i)$ is optimal.*

In the next sections we will use these lemmata to give a general solution approach for some robustness versions of both (P1) and (P2).

The Static Problem. We first describe the problem $\mathcal{P}_{\sigma=1} = (P, M, \mathbb{A}_{rec}, 1) \in$ DRRP(1), where P corresponds to one of the timetabling problems (P1) or (P2) defined above, while M and $\mathbb{A}_{rec}$ are defined as follows:

- The modification function M for an instance $i^0 = (G, w, L^0)$ and a constant $\alpha \in \mathbb{R}^{>0}$ is defined as:

$$M(i^0) = \left\{(G, w, L^1) : \exists \overline{a} \in \mathcal{A} : L^0_{\overline{a}} < L^1_{\overline{a}} \leq L^0_{\overline{a}} + \alpha \text{ and } L^1_a = L^0_a \forall a \neq \overline{a}\right\}.$$

Using this function, we represent the delay of an activity a by increasing the initial value L^0_a to some value $L^1_a > L^0_a$. The definition ensures that only one delay is allowed (and bounded by α) and that nothing changes on all other activities.

- The feasible set of recovery algorithms $\mathbb{A}_{rec}$. We are going to investigate the following two limitations:

– **limited-events**: here we assume that there are resources to change the time for a limited number of events only. In particular, if π is a solution for P and x^1 is a disposition timetable computed by any recovery algorithm in $\mathbb{A}_{rec}$, then x^1 must satisfy

$$d(x^1, \pi) := \left|\{u \in V : x^1_u \neq \pi_u\}\right| \leq \Delta \tag{5}$$

for some given $\Delta \in \mathbb{N}$.

– **limited-delay**: as second limitation of the recovery algorithm, we again require that x^1 must not deviate "too much" from the initial timetable π, but this time we consider the sum of all deviations of all events. I.e. x^1 must satisfy

$$d(x^1, \pi) := \left\|x^1 - \pi\right\|_1 \leq \Delta \tag{6}$$

for some given $\Delta \in \mathbb{N}$.

In both cases, $\Delta = 0$ means strict robustness.

We are looking for a feasible pair (A_{rob}, A_{rec}) for $\mathcal{P}_{\sigma=1}$ (according to Definition 5). The result of A_{rob} is a robust solution $\pi = A_{rob}(i^0) \in F_{\mathcal{P}_{\sigma=1}}$ for each instance $i^0 = (G, w, L^0)$ of P. It has to satisfy the following constraints:

$$\pi_v - \pi_u \geq L^0_a, \quad \forall a = (u,v) \in A$$
$$\forall\ (G, w, L^1) \in M(i^0),\ \forall u \in V,\ \exists\ x^1_u \in \mathbb{R}^{\geq 0}: \tag{7}$$
$$x^1_v - x^1_u \geq L^1_a, \quad \forall a = (u,v) \in A \tag{8}$$
$$x^1_u \geq \pi_u, \quad \forall u \in V \tag{9}$$
$$d(x^1, \pi) \leq \Delta. \tag{10}$$

In Section 4.1 we show that it is NP-hard to compute the $\mathcal{P}_{\sigma=1}$-optimal solution for (P2) and the limited-delay case.

Concerning A_{rec}, let us assume that π is a solution computed by any robust algorithm A_{rob} with respect to the instance i^0. Let $i^1 = (G, w, L^1) \in M(i^0)$. We know that (7)–(10) hold, i.e. a disposition timetable x^1 exists. It can be computed by an *updating version* of CPM:

$$CPM(i^0, \pi, i^1) = \begin{cases} x^1_v = 0 & \text{if } v = v_1 \\ x^1_v = \max\left\{\pi_v, \max\left\{x^1_u + L^1_a\ :\ a = (u,v) \in A\right\}\right\} & \text{otherwise.} \end{cases}$$

This recovery algorithm computes the disposition timetable x^1 with the minimum value of $d(x^1, \pi)$ for both limitations, i.e. it minimizes $\left|\{u \in V : x^1_u \neq \pi_u\}\right|$ and $\left\|x^1 - \pi\right\|_1$ at the same time. Hence it is able to recover (if a recovery solution exists) or to find out that such a solution does not exist. Additionally, among all timetables satisfying constraints (8)–(10), the recovery algorithm provides the disposition timetable with the optimal value for (P2) and, if G is a tree, also for (P1).

The Dynamic Problem. We already remarked that in real-world operation, we have to expect more than one delay. We hence consider $\mathcal{P}_{\sigma\geq 1} = (P, M, \mathbb{A}_{rec}, \sigma) \in \mathrm{DRRP}(\sigma)$, $\sigma \in \mathbb{N}$. We formalize M and $\mathbb{A}_{rec}$ as follows:

- The modification function for an instance $i^{k-1} = (G, w, L^{k-1})$ and a constant $\alpha \in \mathbb{R}^{>0}$ is:

$$M(i^{k-1}) = \left\{(G, w, L^k) : \exists \overline{a} \in \mathcal{A} : L^k_a = L^{k-1}_a\ \forall a \neq \overline{a} \text{ and } L^0_{\overline{a}} \leq L^k_{\overline{a}} \leq L^0_{\overline{a}} + \alpha\right\}.$$

- $\mathbb{A}_{rec}$ is based on the same two limitations as in the static case. In particular, we require that a solution x^k computed by an algorithm in $\mathbb{A}_{rec}$ satisfies $d(x^k, \pi) \leq \Delta$, where d is defined according to (5) or according to (6).

Let $i^0 = (G, w, L^0)$ be an instance of $\mathcal{P}_{\sigma\geq 1}$. Again, each robust solution satisfies the (generalized) constraints (7)-(10). Analogously to the static case, the updating version of CPM can be used as recovery algorithm.

In Section 4 we address the problem of designing robust algorithms for $\mathcal{P}_{\sigma\geq 1} = (P, M, \mathbb{A}_{rec}, \sigma)$ where $P \equiv P2$ and $\mathbb{A}_{rec}$ is based on limited events

(see Eq. (5)). Similarly, in Section 5 we investigate the case in which $P \equiv P1$ and $\mathbb{A}_{rec}$ is based on the overall delay according to Eq. (6). In both sections we provide robust algorithms by using the following general approach: first, add an additional slack time s_a to the lower bounds L_a^0 of each activity $a \in A$; then, compute an optimal solution of the resulting instance and take it as a robust solution. Formally, we obtain an algorithm Alg_s^+ for each value of s:

ALGORITHM Alg_s^+	
INPUT:	An instance $i^0 = (G, w, L^0)$ of (P)
ALGORITHM:	1. Define $\bar{L}_a := L_a^0 + s$
	2. Solve $\bar{i} = (G, w, \bar{L})$ optimally.

The variant Alg_s^* differs from Alg_s^+ at Step 1: $\bar{L}_a := s \cdot L_a^0$, that is, instead of adding the slack time, all lower bounds are multiplied by some value.

According to Lemma 2, in the case of (P1), Step 2 can be done efficiently by the critical path method CPM when G is a tree. Otherwise, linear programming can be used. According to Lemma 3, in the case of (P2), Step 2 can be done efficiently by the critical path method CPM when G is a DAG.

For the recoverable robustness problems addressed in Sections 4 and 5, algorithms Alg_s^+ and Alg_s^* are robust if s is large enough. Moreover, the price of robustness increases in s. In particular:

- Alg_s^+ is strictly robust if $s \geq \alpha$. Alg_s^* is strictly robust if $s \geq \frac{L_a^0 + \alpha}{L_a^0}$ for all $a \in A$.
- Let Alg_s^+ (Alg_s^*) be robust. Then $\mathsf{Alg}_{s'}^+$ ($\mathsf{Alg}_{s'}^*$) is robust for all $s' \geq s$.
- Let $\mathsf{Alg}_{s_1}^+$ and $\mathsf{Alg}_{s_2}^+$ ($\mathsf{Alg}_{s_1}^*$ and $\mathsf{Alg}_{s_2}^*$), $s_2 \geq s_1$, be robust. Then

$$P_{rob}(\mathcal{P}_{\sigma \geq 1}, \mathsf{Alg}_{s_1}^+) \leq P_{rob}(\mathcal{P}_{\sigma \geq 1}, \mathsf{Alg}_{s_2}^+), \text{ and}$$

$$P_{rob}(\mathcal{P}_{\sigma \geq 1}, \mathsf{Alg}_{s_1}^*) \leq P_{rob}(\mathcal{P}_{\sigma \geq 1}, \mathsf{Alg}_{s_2}^*).$$

According to the above algorithmic approach, in order to minimize the price of robustness, the goal is to find the smallest value for s such that the respective algorithms are robust. However, in Section 4 we also provide robust algorithms based on a different approach (adding slack times only to specific activities).

4 Dynamic recovery with number of events as limitation

In this section we consider the first case of limitation: we have to find a timetable π such that in each recovered solution x^k, the times of up to Δ nodes may deviate from the original timetable, i.e. x^k must satisfy (5) for some given $\Delta \in \mathbb{N}$ and for all $k = 1, 2, \ldots, \sigma$. Moreover, we consider the problem $\mathcal{P}_{\sigma \geq 1}$ based on the initial planning problem (P2).

It is clear that each timetable is robust if $\Delta \geq |V| - 1$, so in the following, we will always assume $\Delta \leq |V| - 2$. If $\sigma > \Delta$, we need strict robustness to get a robust solution:

Lemma 4. *If $\sigma > \Delta$, then a timetable is robust if and only if the slack s satisfies $s_a \geq \alpha$ for each $a \in A$. In this case, we have strict robustness.*

From Lemma 1 we know that $F_{\mathcal{P}_{\sigma\geq 1}}(i^0) \subseteq F_{\mathcal{P}_{\sigma=1}}(i^0) \subseteq F(i^0)$. Since the set of robust solutions in the dynamic case is smaller than the same set in the static case, the price of robustness for $\mathcal{P}_{\sigma\geq 1}$ is smaller than or equal to the price of robustness for $\mathcal{P}_{\sigma=1}$. Now, we present an example showing that even $F_{\mathcal{P}_{\sigma\geq 1}}(i^0) \subsetneqq F_{\mathcal{P}_{\sigma=1}}(i^0) \subsetneqq F(i^0)$ holds.

Example 1. Consider a simple instance $i^0 = (G, w, L^0)$ of (P2), where: $G = (V, A)$ is a linear graph with four events and three activities, $w_u = 1$ for each $u \in V$, and $L^0_a = 1$ for each $a \in A$. Concerning $\mathcal{P}_{\sigma\geq 1}$, we fix $\alpha = 1$ and $\Delta = 1$.

Figure 1 shows different solutions (timetables) for the instance i^0. It is easy to see that timetable π_{CPM} (computed by CPM) is feasible for (P2). Conversely, any delay on the first activity implies that all the subsequent three events must be delayed. Hence, since $\Delta = 1$, then π_{CPM} is not in $F_{\mathcal{P}_{\sigma=1}}(i^0)$.

A solution belonging to $F_{\mathcal{P}_{\sigma=1}}(i^0)$ is π. In fact, each possible delay on the three activities, in order, is recovered by the three disposition timetables x, x', and x'', respectively. Note that these timetables differ from π by at most $\Delta = 1$ events. On the other hand, π is not in $F_{\mathcal{P}_{\sigma\geq 1}}(i^0)$. This fact can be observed by assuming that two delays, both of α time, occur on the first two activities. The best disposition timetable that recovers these delays starting from π is x''', but $d(\pi, x''') = 3 > \Delta$.

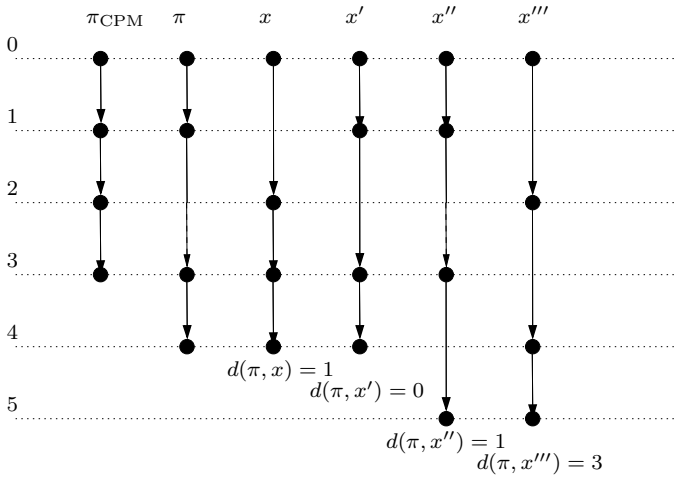


Fig. 1. A graphical representation of different timetables described in Example 1. Bullets represent events and arrows represent activities. Time assigned to each event corresponds to the integer associated to the horizontal line on which the event lies to. The dotted part of arrows represents slack times.

The following lemma implies that $F_{\mathcal{P}_{\sigma\geq 1}}(i^0) = F_{\mathcal{P}_{\sigma=1}}(i^0)$ for $\Delta = 0$.

Lemma 5. *Let $\mathcal{P}_{\sigma=1}$ and $\mathcal{P}_{\sigma\geq 1}$ defined with $\Delta = 0$. If A_{rob} is a robust algorithm for $\mathcal{P}_{\sigma=1}$, then A_{rob} is robust for $\mathcal{P}_{\sigma\geq 1}$.*

Proof. Let A_{rob} be a robust algorithm for $\mathcal{P}_{\sigma=1}$, and let $\pi = A_{rob}(i^0)$. We first show that π assigns a slack time of at least α time to each activity. By contradiction, let us assume that there exists an activity $a = (u, v) \in A$ such that $\pi_v - \pi_u - L_a < \alpha$. Now, if a modification $i^1 \in M(i^0)$ is such that $L^1_a = L_a + \alpha$, namely a delay of α time occurs on a, then $\pi_v - \pi_u < L^1_a$. This means that π is not feasible for i^1, a contradiction for $\Delta = 0$.

Since π assigns a slack of at least α time to each activity, it follows that A_{rob} is a robust algorithm for $\mathcal{P}_{\sigma\geq 1}$. □

4.1 The complexity of computing the price of robustness

We show that the problem of computing the $F_{\mathcal{P}_{\sigma=1}}$-optimal solution is NP-hard for (P2) (and hence also for (P)). This implies that computing the $F_{\mathcal{P}_{\sigma\geq 1}}$-optimal solution is NP-hard.

To capture the concept of *events affected by a delay*, that is, those events that must be postponed as a consequence of a given delay, we give the following definition.

Definition 9. *Given a DAG $G = (V, A)$, a function $s : A \to \mathbb{R}^{\geq 0}$, and a number $\alpha \in \mathbb{R}^{\geq 0}$, a vertex y is* α-influenced *by $(u, v) \in A$ (equivalently (u, v) α-influences y) if there exists a path $p = (u \equiv u_0, v \equiv u_1, \ldots, u_k \equiv y)$ in G such that $\sum_{i=1}^{k} s_{(u_{i-1},u_i)} < \alpha$.*

Remark 1. If x is α-influenced by a according to a path p, then all the vertices belonging to p but the source are α-influenced by a.

In the above definition, function s represents slack times associated to activities able to (partially) absorb a delay. Note that every robust algorithm A_{rob} must provide a timetable π such that each arc cannot influence more than Δ vertices, otherwise there exists no A_{rec} algorithm. In order to show the NP-hardness of computing the $F_{\mathcal{P}_{\sigma=1}}$-optimal solution, we introduce a corresponding decision problem called MRS (Minimum Robust Solution) that uses the concept of α-influence.

MRS Problem	
GIVEN:	DAG $G = (A, V)$, a function $L : A \to \mathbb{R}^{>0}$, a function $w : V \to \mathbb{R}^{\geq 0}$, and three numbers $\alpha \in \mathbb{R}^{>0}$, $\Delta \in \mathbb{N}, K \in \mathbb{N}$
PROBLEM:	Find a timetable $\pi : V \to \mathbb{R}^{\geq 0}$ such that each arc α-influences at most Δ vertices, according to the function $s : A \to \mathbb{R}$ defined as $s_{a=(i,j)} = \pi_j - \pi_i - L_a$, and such that $\sum_{u\in V} w_u \pi_u \leq K$.

Theorem 1. MRS *is NP-complete for any fixed $\Delta \geq 3$.*

Proof. Omitted. □

Corollary 1. *The problem of computing $P_{rob}(\mathcal{P}_{\sigma\geq 1})$ with number of events as limitation is NP-hard.*

4.2 Robust algorithms for $\sigma = 1$ on an arbitrary DAG.

We use the idea described in Section 3.1 (page 10) and add a slack to the minimal durations of all activities or multiply them with some number >1. To this end, let $i = (G, L, w)$, $\alpha \in \mathbb{R}^{\geq 0}$ and $\gamma \in \mathbb{R}^{>0}$, we denote $i_\alpha = (G, L + \alpha, w)$ and $i_\gamma = (G, \gamma L, w)$. We use the critical path method to define robust solutions for $\mathcal{P}_{\sigma=1}$ and $\mathcal{P}_{\sigma\geq 1}$. In particular, we use the algorithms Alg^+_α and Alg^*_γ defined as

- $\mathsf{Alg}^+_\alpha(i) = \mathit{CPM}(i_\alpha)$;
- $\mathsf{Alg}^*_\gamma(i) = \mathit{CPM}(i_\gamma)$.

Let L_{min} be the minimum value assigned by the function L with respect to all the possible instances of $\mathcal{P}_{\sigma\geq 1}$. In the following, let α be as in the definition of the modification function M of $\mathcal{P}_{\sigma\geq 1}$ and $\gamma = (1 + \frac{\alpha}{L_{min}})$. We use α and γ to get concrete instances of Alg^+_α and Alg^*_γ. According to the proof of Lemma 5, if $\mathcal{P}_{\sigma=1}$ is defined with $\Delta = 0$, then every robust algorithm for $\mathcal{P}_{\sigma=1}$ must provide solutions that assign a slack time of at least α to each activity. Then, it follows that Alg^+_α is a robust algorithm for $\mathcal{P}_{\sigma=1}$. To show that also Alg^*_γ is a robust algorithm for $\mathcal{P}_{\sigma=1}$, it is sufficient to observe that for each activity $a \in A$,

$$\gamma L_a = (1 + \frac{\alpha}{L_{min}})L_a = L_a + \alpha\frac{L_a}{L_{min}} \geq L_a + \alpha.$$

The following lemma shows the price of robustness of Alg^*_γ.

Lemma 6. *Let $\mathcal{P}_{\sigma=1}$ be defined with $\Delta = 0$. Then, $P_{rob}(\mathcal{P}_{\sigma=1}, \mathsf{Alg}^*_\gamma) = 1 + \alpha/L_{min}$.*

Proof. By definition,

$$P_{rob}(\mathcal{P}_{\sigma=1}, \mathsf{Alg}^*_\gamma) = \max_{i\in I}\left\{\frac{f(\mathsf{Alg}^*_\gamma(i))}{\min\{f(x) : x \in F(i)\}}\right\}.$$

Let $i = (G, L, w)$ be an instance of $\mathcal{P}_{\sigma=1}$. Denoting by π^γ the solution provided by $\mathsf{Alg}^*_\gamma(i)$, and by π the solution provided by $\mathit{CPM}(i)$, then

$$P_{rob}(\mathcal{P}_{\sigma=1}, \mathsf{Alg}^*_\gamma) = \max_{i=(G,L,w)\in I}\frac{\sum_{u\in V} w_u\pi^\gamma_u}{\sum_{u\in V} w_u\pi_u}.$$

Now we show that for each $v \in V$, $\pi^\gamma_v = \gamma\pi_v$. By contradiction, let $v \in V$ be an event such that $\pi^\gamma_v \neq \gamma\pi_v$ and π^γ_v is minimum. Clearly, v must be different from v_1 and hence there exists an activity $a = (u, v) \in A$ such that $\pi^\gamma_v = \pi^\gamma_u + \gamma L_a$. As

π_v^γ is minimum and $\pi_u^\gamma < \pi_v^\gamma$, then $\pi_u^\gamma = \gamma\pi_u$. It follows $\pi_v^\gamma = \gamma\pi_u + \gamma L_a = \gamma\pi_v$, a contradiction. Hence,

$$P_{rob}(\mathcal{P}_{\sigma=1}, \mathsf{Alg}_\gamma^*) = \max_{i=(G,L,w)\in I} \frac{\sum_{u\in V} w_u \pi_u^\gamma}{\sum_{u\in V} w_u \pi_u} = \max_{i=(G,L,w)\in I} \frac{\sum_{u\in V} w_u \gamma\pi_u}{\sum_{u\in V} w_u \pi_u} = \gamma.$$

□

Lemma 7. *For each instance $i \in I$, $f(\mathsf{Alg}_\alpha^+(i)) \leq f(\mathsf{Alg}_\gamma^*(i))$.*

Proof. Let $i = (G, L, w) \in I$. Let us denote by π^γ the solution provided by $\mathsf{Alg}_\gamma^*(i)$, by π^α the solution provided by $\mathsf{Alg}_\alpha^+(i)$, and by π the solution provided by $CPM(i)$. To prove the statement, it is sufficient to show that

$$\pi_u^\alpha \leq \pi_u^\gamma, \ \forall\ u \in V.$$

By contradiction, let us assume that there exists an event u such that $\pi_u^\alpha > \pi_u^\gamma$. In the proof of Lemma 6, it is shown that $\pi_u^\gamma = \gamma\pi_u$. Then,

$$\pi_u^\alpha > \pi_u^\gamma = \gamma\pi_u = (1 + \frac{\alpha}{L_{min}})\pi_u.$$

Since, by contradiction hypothesis $\pi_u^\alpha > \pi_u^\gamma$, then $u \neq v_1$. It follows that there exists a path $(v_1, \ldots, u)$ in G such that its length ℓ is greater than 0. By definition of Alg_α^+, then

$$\pi_u^\alpha = \pi_u + \alpha\ell.$$

In conclusion,

$$\pi_u^\alpha = \pi_u + \alpha\ell > (1 + \frac{\alpha}{L_{min}})\pi_u = \pi_u + \alpha\frac{\pi_u}{L_{min}}.$$

It follows that $\ell L_{min} > \pi_u$, a contradiction. □

Lemmata 6 and 7 imply the following results.

Corollary 2. *Let $\mathcal{P}_{\sigma=1}$ be defined with $\Delta = 0$. Then, $P_{rob}(\mathcal{P}_{\sigma=1}, \mathsf{Alg}_\alpha^+) \leq 1 + \alpha/L_{min}$.*

4.3 Robust algorithms for $\sigma = 1$ on a linear graph.

In this section, we present an algorithm that computes an optimal robust timetable for the problem (P2) for the case $\sigma = 1$ on a linear graph. The idea of the algorithm is to add each slack "as late as possible". Let $V = \{v_1, \ldots, v_{|V|}\}$ be ordered such that $A = \{a_1 = (v_1, v_2), \ldots, a_{|A|} = (v_{|V|-1}, v_{|V|})\}$. Define s^α by

$$s_{a_j}^\alpha := \begin{cases} \alpha & \text{if } (\Delta+1)|j \\ 0 & \text{else} \end{cases} \tag{11}$$

for all arcs $a_j \in A$. We then add s_a^α to L_a^0 for each $a \in A$ and calculate a solution of (P2) by applying CPM. We denote this algorithm by $\mathsf{Alg}_{s^\alpha}^+$. The following lemma states that a timetable π is robust if and only if the slack time of each $\Delta + 1$ consecutive arcs is large enough to let vanish the delay.

Lemma 8. *If $\sigma = 1$ and $\Delta \leq |V| - 2$, a timetable π for a linear graph G is robust if and only if*

$$\sum_{k=0}^{\Delta} s_{a_{j+k}} \geq \alpha \quad \text{for each } j = 1, \ldots, |A| - \Delta. \tag{12}$$

Theorem 2. *$\mathsf{Alg}^+_{s^\alpha}$ is an optimal robust algorithm for $\mathcal{P}_{\sigma=1}$ based on the initial problem (P2).*

Proof. Omitted. □

Corollary 3. *If G is a linear graph, there exists a linear time algorithm that computes $\mathcal{P}_{\sigma=1}$-optimal solutions.*

Proof. Running algorithm $\mathsf{Alg}^+_{s^\alpha}$ on a linear graph needs time $\mathcal{O}(|A|)$, checking whether the output satisfies $\sum_{u \in V} w_u \pi_u \leq K$ needs time $\mathcal{O}(|V|)$. As $\mathsf{Alg}^+_{s^\alpha}$ is an optimal robust algorithm, it finds a feasible timetable if and only if (MRS) is feasible. □

4.4 Robust algorithms for arbitrary σ on a linear graph.

We now present an algorithm for an arbitrary σ if G is a linear graph. It assigns the same slack

$$s^* = \min \left\{ \alpha, \frac{\sigma\alpha}{\Delta + 1} \right\} \tag{13}$$

to each arc. In the following, we will show that $\mathsf{Alg}^+_{s^*}$ is robust and that it is optimal compared to all robust algorithms that add an equal slack s to all arcs. We need the following two lemmata for the proof:

Lemma 9. *If $s_a < \alpha$ for all arcs $a \in A$, then the number of nodes affected by a single delay of $\sigma\alpha$ on arc $a_j = (v_j, v_{j+1})$ is equal to the number of nodes affected by σ single delays of α on the σ consecutive arcs $a_{j+k} = (v_{j+k}, v_{j+k+1})$, $k = 0, \ldots, \sigma - 1$.*

Proof. Let $s_a < \alpha$ for all arcs $a \in A$. If, on the one hand, a_j is delayed by $\sigma\alpha$, then v_{j+1} has a delay of $\sigma\alpha - s_{a_j}$, v_{j+2} has a delay of $\sigma\alpha - s_{a_j} - s_{a_{j+1}}$ and so on, and $v_{j+\sigma}$ has a delay of $\sigma\alpha - \sum_{k=0}^{\sigma-1} s_{a_{j+k}}$. If, on the other hand, $a_j, \ldots, a_{j+\sigma-1}$ are delayed by α, then v_{j+1} has a delay of $\alpha - s_{a_j}$, v_{j+2} has a delay of $2\alpha - s_{a_j} - s_{a_{j+1}}$ and so on, and $v_{j+\sigma}$ has a delay of $\sigma\alpha - \sum_{k=0}^{\sigma-1} s_{a_{j+k}}$. As $s_a < \alpha$ for all arcs $a \in A$, all these delays are positive.

So in both cases, the nodes $v_{j+1}, \ldots, v_{j+\sigma}$ are affected, and as the delay of $v_{j+\sigma}$ is the same in both cases, the total number of subsequent affected nodes is the same, too. □

Lemma 10. *If all arcs $a \in A$ have the same slack $s_a = s$, then the number of nodes affected by σ delays of α on σ consecutive arcs is always greater than or equal to the number of nodes affected by σ delays of α on σ non-consecutive arcs.*

Proof. Omitted. □

Now, we can prove that $\mathsf{Alg}_{s^*}^+$ is robust and that it is optimal for (P2):

Theorem 3. *Let G be a linear graph. Assume that we add the same slack time s to all arcs. Then Alg_s^+ is a robust algorithm for $\mathcal{P}_{\sigma\geq 1}$ if and only if $s \geq s^*$. If $s > s^*$ and $|V| > 1$, then $f(\mathsf{Alg}_s^+(i)) > f(\mathsf{Alg}_{s^*}^+(i))$ if $w_u \geq 0$ for all nodes $u \in V$ and $w_u > 0$ for at least one node $u \in V$.*

Proof. Omitted. □

Theorem 4. *Let s^* be defined as in Eq. (13), and let G be a linear graph. If, for each $(G, w, L^0) \in I$, $w_a > 0$ for at least one $a \in A$, then $P_{rob}(\mathcal{P}_{\sigma\geq 1}, \mathsf{Alg}_{s^*}^+) \leq 1 + s^*/L_{min}$.*

Proof. Omitted. □

5 Dynamic recovery with sum of delays as limitation

In this section we consider the second case of limitation: We have to find a timetable π such that each recovered solution x^k must not deviate too much from the initial timetable π, i.e. x^k must satisfy

$$d(x^k, \pi) := \left\|x^k - \pi\right\|_1 \leq \Delta \tag{14}$$

for some given $\Delta \in \mathbb{N}$ and for all $k = 1, 2, \ldots, \sigma$. Throughout this section, we consider the problem $\mathcal{P}_{\sigma\geq 1}$ based on the initial planning problem (P1) which implies that all weights w_a are nonnegative.

Again, our strategy to make a timetable robust against delays is to add an amount s of slack time to all the arcs, i.e. to use the algorithm Alg_s^+.

We first investigate how much we loose in the optimal solution if we use Alg_s^+ instead of an algorithm that computes the optimal (but not robust) solution of (P1), i.e. without the additional slack s. Again, let L_{min} be the minimum value assigned by the function L with respect to all the possible instances of $\mathcal{P}_{\sigma\geq 1}$.

Lemma 11. *Let G be a tree and let $w_a > 0$ for at least one $a \in A$. If Alg_s^+ is robust, its price of robustness is $P_{rob}(\mathcal{P}_{\sigma\geq 1}, \mathsf{Alg}_s^+) \leq 1 + s/L_{min}$.*

Proof.

$$\begin{aligned}
P_{rob}(\mathcal{P}_{\sigma\geq 1}, \mathsf{Alg}_s^+) &= \max_{i=(G,w,L^0)\in I} \frac{\sum_{a=(u,v)\in A} w_a(L_a^0 + s)}{\sum_{a=(u,v)\in A} w_a L_a^0} \\
&= 1 + s \cdot \max_{i=(G,w,L^0)\in I} \frac{\sum_{a=(u,v)\in A} w_a}{\sum_{a=(u,v)\in A} w_a L_a^0} \\
&\leq 1 + s \cdot \max_{i=(G,w,L^0)\in I} \frac{\sum_{a=(u,v)\in A} w_a}{\sum_{a=(u,v)\in A} w_a L_{min}} \\
&\leq 1 + s/L_{min}.
\end{aligned}$$

□

Now we discuss how much slack time s is needed to guarantee robustness of Alg_s^+. Our first result deals with strict robustness, i.e. if $\Delta = 0$. In this case we have to make sure that any delay can be compensated by the slack time on the corresponding edge. Since L_a^k never differs from L_a^0 by more than α in any scenario, it suffices to add an additional slack of α to each L_a^0 for all $a \in A$. Then the resulting disposition timetable in each step equals the original timetable π, i.e. a recovery step is in fact not necessary.

Lemma 12. *The algorithm Alg_α^+ is strictly robust (i.e. it is robust for the case $\Delta = 0$) for any graph G. Furthermore, if G is a tree, its price of robustness is $P_{rob}(\mathcal{P}_{\sigma\geq 1}, \mathsf{Alg}_\alpha^+) \leq 1 + \alpha/L_{min}$.*

Proof. The robustness of Alg_α^+ is clear. The price of robustness follows from Lemma 11. □

Now we turn our attention to the case $\Delta > 0$, but first only look at one recovery step (i.e. $\sigma = 1$). If $\Delta \leq \frac{\alpha}{2}$, a robust solution can be found as follows:

Lemma 13. *Let $\sigma = 1$ and $\Delta \leq \frac{\alpha}{2}$. Then $\mathsf{Alg}_{\alpha-\Delta}^+$ is robust. Furthermore, if G is a tree, its price of robustness is $P_{rob}(\mathcal{P}_{\sigma\geq 1}, \mathsf{Alg}_{\alpha-\Delta}^+) \leq 1 + (\alpha - \Delta)/L_{min}$.*

Proof. Let π be a solution computed by $\mathsf{Alg}_{\alpha-\Delta}^+$ and let x be the solution after the recovery phase. Assume that arc $(u, v) \in A$ is delayed by α. Let w_j, $j = 1, \ldots, l$, be the set of nodes directly connected to v by an arc $(v, w_j) \in A$, see Figure 2. We calculate the delays as

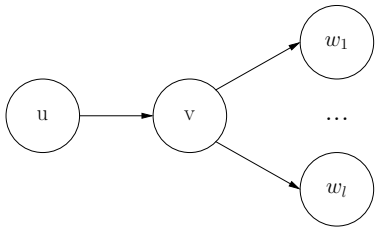


Fig. 2. The DAG for the proof of Lemma 13.

$$x_v - \pi_v = \alpha - (\alpha - \Delta) = \Delta$$
$$x_{w_j} - \pi_{w_j} \leq [\Delta - (\alpha - \Delta)]_+ = 0 \text{ for all } j = 1, \ldots, l,$$

the latter using $\Delta \leq \frac{\alpha}{2}$. Hence,

$$\sum_{u \in V} (x_u - \pi_u) = \Delta.$$

The price of robustness follows from Lemma 11. □

Next, we simplify the network and look at a linear graph. However, this simplification allows to drop the restrictions on Δ and σ from the previous lemmata.

Theorem 5. *Let G be a linear graph. Then Alg_s^+ is robust for $\mathcal{P}_{\sigma\geq 1}$ if and only if*

$$s \geq s^* := \frac{2\sigma\alpha\left(\left\lceil\frac{2\Delta}{\sigma\alpha}\right\rceil+\sigma\right)-\sigma\alpha(\sigma+1)-2\Delta}{\left(\left\lceil\frac{2\Delta}{\sigma\alpha}\right\rceil+\sigma\right)\left(\left\lceil\frac{2\Delta}{\sigma\alpha}\right\rceil+\sigma-1\right)}.$$

Proof. Omitted. □

Corollary 4. *It holds that $s^* \geq \frac{\sigma^2\alpha^2}{2\Delta+\sigma^2\alpha}$ where equality holds if $\frac{s\Delta}{\sigma\alpha}$ is integer.*

Proof. One can compute that $s^* \geq \frac{\sigma^2\alpha^2}{2\Delta+\sigma^2\alpha}$ if and only if

$$\underbrace{\left\lceil\frac{2\Delta}{\sigma\alpha}\right\rceil\sigma\alpha}_{:=A}\underbrace{\left(4\Delta+\sigma\alpha-\sigma\alpha\left\lceil\frac{2\Delta}{\sigma\alpha}\right\rceil\right)}_{:=B} \geq \underbrace{2\Delta}_{:=C}\underbrace{(2\Delta+\sigma\alpha)}_{:=D}.$$

For the latter expression note that $A, B, C, D \geq 0$ and that $A+B = C+D$ and that $A-B \leq D-C$. Hence $AB \geq CD$ and the lower bound is established.

Plugging in s^* in the case that $\frac{s\Delta}{\sigma\alpha}$ is integer shows (after some calculations) that equality holds. □

The price of robustness of algorithm $\mathsf{Alg}_{s^*}^+$ can finally be written down.

Corollary 5. *Let G be a linear graph. Then $P_{rob}(\mathcal{P}_{\sigma\geq 1}, \mathsf{Alg}_{s^*}^+) = 1 + s^*$ where s^* is the minimal slack time of Theorem 5.*

Note that for a concrete scenario, a slack smaller than s^* might also give a robust timetable. This might happen for example if no two source-delayed arcs follow each other or if the size of the network is limited such that at least one node $u \in V$ with a delay of $(x_u - \pi_u) > s^*$ has no outgoing arc. However, we are not interested in one special scenario, but in all possible scenarios from the set of admissible scenarios.

We also remark that this is a discussion of $P_{rob}(\mathcal{P}, \mathsf{Alg}_s^+)$ only. The question if there exists an approach which does better in the worst case is still open. But note that it need not be optimal to add the same slack s to all arcs when the weights w_a are different from each other. This can be seen in the following

Example 2. Consider $G = (V, A)$ with $V = \{v_1, v_2, v_3, v_4\}$, $A = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$, weights $w = (1, 100, 1)$ and lower bounds $L^0 = (1, 1, 1)$, see Fig. 3. Let $\alpha = 4$, $\Delta = 5$ and $\sigma = 1$. If we add the same slack to all arcs, we need at least a slack of 2. With $s = (2, 2, 2)$, we have

$$\sum_{a=(u,v)\in A} w_a(\pi_v - \pi_u) \;=\; \sum_{a=(u,v)\in A} w_a(L_a^0 + s) \;=\; 306$$

(if we schedule each node as early as possible, i.e. $\pi_j := \pi_{j-1} + L_{j-1} + s_{j-1}$, $j = 2, \ldots, 4$). If we allow different slacks on the arcs and set $s = (2, 0, 3)$, we get a robust timetable with

$$\sum_{a=(u,v)\in A} w_a(\pi_v - \pi_u) = 107.$$

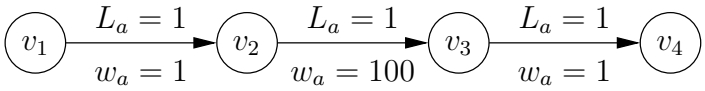


Fig. 3. The DAG for example 2.

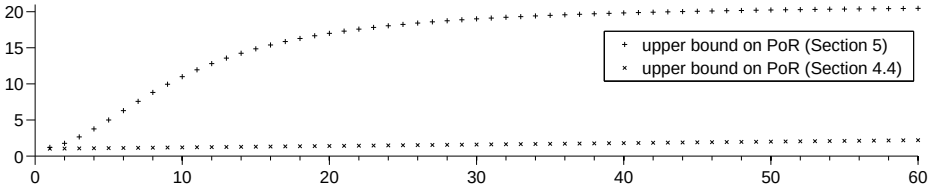


Fig. 4. The price of robustness, depending on the actual restrictions on the recovery algorithm, for $\alpha = 20$ and $\Delta = 1000$ as a function of σ.

6 Conclusions

In this paper, we showed how the concept of recoverable robustness from [15] can be extended to the concept of dynamic recoverable robustness. We showed how this concept can be applied to the delay management problem and suggested different concrete restrictions of the recovery algorithm.

Depending on the concrete restrictions on the recovery algorithms, the price of robustness is very different. In Figure 4, we give the price of robustness for a linear graph if we either restrict the number of nodes being affected by a delay or if we restrict the allowed deviation from the original timetable.

References

1. H. G. Bayer and B. Sendhoff. Robust Optimization - A Comprehensive Survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, 2007.

2. A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Mathematical Programming: Special Issue on Robust Optimization*, volume 107. Springer, Berlin, 2006.
3. D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.
4. J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming*. Springer Verlag, New York, 1997.
5. A. Borodin and R. El-Yaniv, editors. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
6. S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In *Proc. of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS07)*, 2007.
7. M. Fischetti and M. Monaci. Robust optimization through branch-and-price. In *Proceedings of the 37th Annual Conference of the Italian Operations Research Society (AIRO)*, 2006.
8. M. Fischetti and M. Monaci. Light robustness. Research Paper ARRIVAL-TR-0066, ARRIVAL project, 2008.
9. M. Gatto, B. Glaus, R. Jacob, L. Peeters, and P. Widmayer. Railway delay management: Exploring its algorithmic complexity. In *Algorithm Theory - Proceedings SWAT 2004*, volume 3111 of *LNCS*, pages 199–211. Springer, 2004.
10. M. Gatto, R. Jacob, L. Peeters, and A. Schöbel. The Computational Complexity of Delay Management. In *Proc. of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 3787 of *Lecture Notes in Computer Science*, 2005.
11. M. Gatto, R. Jacob, L. Peeters, and P. Widmayer. Online Delay Management on a Single Train Line. In *Proc. of the Algorithmic Methods for Railway Optimization (ATMOS04)*, volume 4359 of *Lecture Notes in Computer Science*, 2007.
12. A. Ginkel and A. Schöbel. The bicriteria delay management problem. *Transportation Science*, 41(4):527–538, 2007.
13. L. Giovanni, G. Heilporn, and M. Labbé. Optimization models for the delay management problem in public transportation. *European Journal of Operational Research*, 2006. to appear.
14. P. Kall and S.W. Wallace. *Stochastic Programming*. Wiley, Chichester, 1994.
15. C. Liebchen, M. Lüebbecke, R. H. Möhring, and S. Stiller. Recoverable robustness. Technical Report ARRIVAL-TR-0066, ARRIVAL Project, 2007.
16. A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming, Handbooks in Operations Research and Management Science Volume 10*. North-Holland, 2003.
17. A. Schöbel. A model for the delay management problem based on mixed integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.
18. A. Schöbel. Book Chapter: Integer programming approaches for solving the delay management problem. volume 4359 of *Lecture Notes in Computer Science*, pages 145–170, 2007.

Efficient On-Trip Timetable Information in the Presence of Delays

Lennart Frede[1], Matthias Müller–Hannemann[2] and Mathias Schnee[1]

[1]Darmstadt University of Technology, Computer Science,
Hochschulstraße 10, 64289 Darmstadt, Germany
{frede,schnee}@algo.informatik.tu-darmstadt.de
[2]Martin-Luther-University Halle, Computer Science,
Von-Seckendorff-Platz 1, 06120 Halle, Germany
muellerh@informatik.uni-halle.de

Abstract. The search for train connections in state-of-the-art commercial timetable information systems is based on a static schedule. Unfortunately, public transportation systems suffer from delays for various reasons. Thus, dynamic changes of the planned schedule have to be taken into account. A system that has access to delay information of trains (and uses this information within search queries) can provide valid alternatives in case a train change breaks. Additionally, it can be used to actively guide passengers as these alternatives may be presented before the passenger is already stranded at a station due to a broken transfer. In this work we present an approach which takes a stream of delay information and schedule changes on short notice (partial train cancellations, extra trains) into account. Primary delays of trains may cause a cascade of so-called secondary delays of other trains which have to wait according to certain waiting policies between connecting trains. We introduce the concept of a dependency graph to efficiently calculate and update all primary and secondary delays. This delay information is then incorporated into a time-expanded search graph which has to be updated dynamically. These update operations are quite complex, but turn out to be not time-critical in a fully realistic scenario. We finally present a case study with data provided by Deutsche Bahn AG showing that this approach has been successfully integrated into our multi-criteria timetable information system MOTIS and can handle massive delay data streams instantly.

Keywords: timetable information system, primary and secondary delays, dependency graph, dynamic graph update

1 Introduction and Motivation

In recent years the performance and quality of service of electronic timetable information systems has increased significantly. Unfortunately, not everything runs smoothly in scheduled traffic and the presence of delays is the norm rather than the exception.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1584

Delays can have various causes: Disruptions in the operations flow, accidents, malfunctioning or damaged equipment, construction work, repair work, and extreme weather conditions like snow and ice, floodings, and landslides to name just a few. A system that incorporates up-to-date train status information (most importantly information about future delays based on the current situation) can provide a user with valid timetable information in the presence of disturbances.

Such an on-line system can additionally be utilized to verify the current status of a journey:

- Journeys can either be still valid (i.e., they can be followed as planned),
- can be affected such that the arrival at the destination is delayed,
- or may no longer be possible.

In the latter case a connecting train will be missed, either because the connecting train cannot wait for a delayed train, or the connecting train may have been canceled. In a delay situation, such a status information is very helpful. In the positive case that all planned train changes are still possible, passengers can be reassured that they do not have to worry about potential train misses. To learn that one arrives x minutes late with the planned sequence of trains may allow a customer to make arrangements, e.g. inform someone to pick one up later accordingly. In the unfortunate case that a connecting train will be missed, this information can now be obtained well before the connection breaks and the passenger is stranded at some station. Therefore, valid alternatives may be presented while there are still more possibilities to act. This situation is clearly preferable over missing a connecting train and than going to a service point to request an alternative.

As up to now the commercial systems do not take the current situation into account (although estimated arrival times may be accessible for a given connection, these times are not used actively during the search), their recommendations may be impossible to use, as the proposed alternatives already suffer from delays and may even already be infeasible at the time they are delivered by the system.

Static timetable information systems. The standard approach to model static timetable information is as a shortest path problem in either a time-expanded or time-dependent graph. The recent survey [1] describes the models and suitable algorithms in detail. We developed our timetable information system MOTIS which performs a multi-criteria search for train connections in a realistic environment using a suitably constructed time-expanded graph. Our underlying model ensures that each proposed connection is indeed feasible, i.e. can be used in reality. The criteria considered are travel time, number of interchanges, ticket cost, and reliability of all interchanges of a connection. The system is able to present many attractive alternatives to customers [2].

Our contribution and overview. Previous research on timetable information systems has focused on the static case where the timetable is considered as fixed. Here we start out a new thread of research on dynamically changing timetable

data due to disruptions. We extended our timetable information system MOTIS to use current train status information. Modeling issues have been discussed on a theoretical level but no true to life system with real delay data has been studied in the literature and to our knowledge no such system that guarantees optimal results (with respect to even a single optimization criterion) exists.

We give first results of implementing such a system for a real world scenario with no simplifying assumptions at all. The architecture we propose is intended for a multi-server environment where the availability of search engines has to be guaranteed at all times. Our system consists of two main components, the *dependency graph* and the *search graph*. The dependency graph is used to efficiently propagate primary delay information according to waiting policies. The overall new status information is then incorporated into the search graph which is used for customer search queries. Our dependency graph is similar to a simple time-expanded graph model with distinct nodes for each departure and arrival event of the whole schedule for the current and following days. This is a natural and efficient model since every event has to store its own update information. For the search graph, however, we are free to use either the time-expanded or the time-dependent model. In this paper, we have chosen to use the time-expanded model for the search graph since MOTIS is based on this. Although update operations are quite complex in this model, it will turn out that they can be performed very efficiently, in less than a millisecond per update message on average.

We will also discuss the difference between searches in an *on-trip* scenario, where a passenger is either stranded at a station or in a train whose connecting train will be missed, to classical *pre-trip* searches.

The rest of this paper is organized as follows: In Section 2, we will discuss primary and secondary delays. We introduce our architecture in Section 3 and its two components, the update of the search graph (in Section 4) and the propagation algorithm on our dependency graph model (in Section 5). In Section 6, we present our approach to perform on-trip as opposed to pre-trip searches. Afterwards, we provide our experimental results in Section 7. Finally, we conclude and give an outlook.

Related work. Delling et al. [3] independently of us came up with ideas on how to regard delays in timetabling systems. In contrast to their work we do not primarily work on edge weights, but consider nodes with time stamps. The edge weight for time follows, whereas edge weights for transfers and cost do not change during the update procedures. This is important for the ability to do multi-criteria search.

A related field of current research is disposition and delay management. Gatto et al. [4,5] have studied the complexity of delay management for different scenarios and have developed efficient algorithms for certain special cases using dynamic programming and minimum cut computations. Various waiting policies have been discussed, for example by Ginkel and Schöbel [6]. Schöbel [7] also proposed integer programming models for delay management. Stochastic models

for the propagation of delays are studied, for example, by Meester and Muns [8]. Waiting policies in a stochastic context are treated in [9].

2 Up-To-Date Status Information

2.1 Primary Delay Information

First of all, the input stream of status messages consists of reports that a certain train departed or arrived at some station at time τ either on time or delayed by x minutes. In case of a delay, such a message is followed by further messages about predicted arrival and departure times for all upcoming stations on the train route.

Besides, there can be information about additional special trains (a list of departure and arrival times at stations plus category, attribute and name information). Furthermore, we have (partial) train cancellations, which include a list of departure and arrival times of the canceled stops (either all stops of the train or from some intermediate station to the last station).

Moreover, we have manual decisions by the transport management of the form: "Change from train t to t' will be possible" or "will not be possible". In the first case it is guaranteed that train t' will wait as long as necessary to receive passengers from train t. In the latter case the connection is definitively going to break although the current prediction might still indicate otherwise. This information may depend on local knowledge, e.g. that not enough tracks are available to wait or that additional delays are likely to occur, or may be based on global considerations about the overall traffic flow. We call messages of this type *connection status decisions*.

2.2 Secondary Delays

Secondary delays occur when trains have to wait for other delayed trains. Two simple, but extreme examples for waiting policies are:

- *never wait*
  In this policy, no secondary delays occur at all. This causes many broken connections and in the late evening it may imply that customers do not arrive at their destination on the same travel day. However, nobody will be delayed who is not in a delayed train.
- *always wait as long as necessary*
  In this strategy, there are no broken connections at all, but massive delays are caused for many people, especially for those whose trains wait and have no delay on their own.

Both of these policies seem to be unacceptable in practice. Therefore, train companies usually apply a more sophisticated rule system specifying which trains have to wait for others and for how long. For example, the German railways Deutsche Bahn employ a complex set of rules, dependent on train type and local specifics.

In essence, this works as follows: There is a set of rules describing the maximum amount of time a train t may be delayed to wait for passengers from a feeding train f. Basically, these rules depend on train categories and stations. But there are also more involved rules, like if t is the last train of the day in that direction, the maximum delay time is increased, or during peak hours, when trains operate more frequently, the maximum delay time may be decreased.

The *waiting time* $wt(t, s, f)$ is the maximum delay acceptable for train t at station s waiting for a feeding train f. Let $dep_{sched}(s, t)$ and $dep(s, t)$ be the departure time according to the schedule resp. the new departure time of train t at station s, $arr(s, t)$ the arrival time of a train and $minct(s, f, t)$ the minimum change time needed from train f to train t at station s. Note that in a delayed scenario the change time can be reduced, as guides may be available that show changing passengers the way to their connecting train. If the following equation holds

$$arr(s, f) + minct(s, f, t) - dep_{sched}(s, t) < wt(t, s, f)$$

train t will incur a secondary delay because it waits for f at station s. Its new departure time is determined by the following equation

$$dep(s, t) = \begin{cases} arr(s, f) + minct(s, f, t) & \text{if } t \text{ waits} \\ dep_{sched}(s, t) & \text{otherwise .} \end{cases}$$

In case of several delayed feeding trains, the new departure time will be determined as the maximum over these settings.

During day-to-day operations these rules are always applied automatically. If the required waiting time of a train lies within the bounds defined by the rule set, trains will wait. Otherwise they will not. All exceptions from these rules have to be given as connection status decisions.

3 System Architecture

Our system consists of two main components. One part is responsible for the propagation of delays from the status information and for the calculation of secondary delays, while the other component handles connection queries. The core of the first part is a *dependency graph* which models all the dependencies between different trains and between the stops of the same train (in Section 5 we give more details on that). The obtained information is then sent to the search graph which is updated accordingly. This decoupling of dependency and search graph allows us to use any graph model for the search graph.

In a distributed scenario this architecture can be realized with one server for the dependency graph that continuously receives new status information and broadcasts the update information to a number of servers on which the query algorithms run. Load balancing can schedule the update phases for each server. If this is done in a round robin fashion, the availability of service is guaranteed.

Our design decision to work with two separate components also gives us additional flexibility when to broadcast the update information. In this paper, we

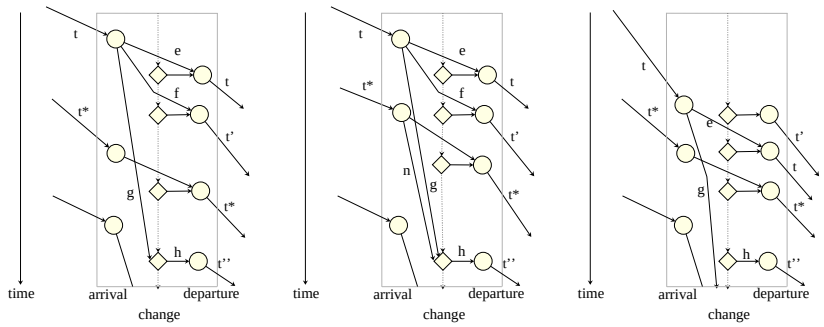


Fig. 1. The change level at a station (left) and changes if train t^* arrives earlier (middle picture) or train t arrives later (right).

broadcast the update information immediately when it becomes available. However, a reasonable alternative is to broadcast a consistent update state only every Δ minutes, for some small Δ. This option may save many update operations in the search graph which, in particular, result from small oscillations in forecasts of trains with frequent comparisons of actual and scheduled times.

4 Updating the Search Graph

Time-Expanded graph model. Let us briefly recall the time-expanded graph model. The basic idea is to introduce a directed search graph where every node corresponds to a specific event (departure, arrival, change of a train) at a station.

A connection served by a train from station A to station B is called *elementary*, if the train does not stop between A and B. Edges between nodes represent either elementary connections, waiting within a station, or changing between two trains. For each optimization criterion, a certain length is associated with each edge.

Traffic days, possible attribute requirements and train class restrictions with respect to a given query can be handled quite easily. We simply mark train edges as *invisible* for the search if they do not meet all requirements of the given query. With respect to this visibility of edges, there is a one-to-one correspondence between feasible connections and paths in the graph.

More details of the graph model can be found in [2].

Modeling interchanges in a time-expanded graph. To model non-constant change times between pairs of trains, additional nodes and edges are required besides the ones for arrival and departure events. In forward search (when the desired departure is specified), for every departure time at a station there is a change node connected via *entering edges* to all departure nodes at that time. The change nodes are interconnected with *waiting edges*. *Leaving edges* link to

the first change node which is reachable in the time needed for a transfer from this train to any other. All possible shorter change times (e.g. for trains at the same platform) are realized using *special transfer edges*. Additionally, we have *stay-in-train edges*. Only entering edges carry the cost of a train change.

In Figure 1 (left) it is possible to change from train t to all trains departing not earlier than t'' using leaving edge g, any number of waiting edges and an entering edge (e.g. h to enter t''). A change to train t' on the same platform is also feasible using special interchange edge f and, of course, to stay in train t via stay-in-train edge e. However, it is impossible to change to train t^* although it departs later than t', because it requires more time to reach it.

Updates The update in the search graph does not simply consist of setting new time stamps for nodes (primary and secondary delays), insertions (additional trains) and deletions (cancellations) of nodes and resorting lists of nodes afterwards. Furthermore, all the edges present to model the changing of trains at the affected stations have to be recomputed respecting the changed time stamps, additional and deleted nodes, and connection status information. The following adjustments are required on the change level (see Figure 1):

- Inserting change nodes or unhooking them from the waiting edges chain at times where a new event is the only one or the only event is moved away or canceled.
- Updating the leaving edges pointing to the first node reachable after a train change.
- Updating the nodes reachable from a change node via entering edges.
- Recalculating special interchange edges from resp. to arrival resp. departure nodes with a changed time stamp (either remove, adjust or insert special interchange edges).

The result of the update phase is a graph that looks and behaves exactly as if it was constructed from a schedule describing the current situation. Additionally it contains information about the original schedule and reasons for the delays.

Next we give two examples for updating the search graph[1]: Suppose train t^* manages to get rid of some previous delay and now arrives and departs earlier than previously predicted (see Figure 1, middle part). In the new situation it is now possible, to change to train t'' using the new leaving edge n and the existing entering edge h.

In our second example let train t arrive delayed as depicted in Figure 1 (right). As it now departs after t', it is not only impossible to change to t' (special interchange edge f is deleted), but also the change departure nodes for the departures of t' and t are in reverse order. Therefore, the waiting edges have to be relinked. Furthermore, a change to t'' is no longer possible, so the leaving edge h points to a node later than the departure of t''.

[1] Note that we increased the station dependent interchange time from the middle to the right extract to make this example work.

5 Dependency Graph

5.1 Graph Model

Our *dependency graph* (see Fig. 2) models the dependencies between different trains and between the stops of the same train. Its node set consists of four types of nodes:

- departure nodes,
- arrival nodes,
- forecast nodes, and
- schedule nodes.

Each node has a time stamp which can dynamically change. Departure and arrival nodes are in one-to-one correspondence with departure and arrival events. Their time-stamps reflect the current situation, i.e. the expected departure or arrival time subject to all delay information known up to this point.

Schedule nodes are marked with the planned time of an arrival or departure event, whereas the time stamps of forecast nodes is the current external prediction for their departure or arrival time.

The nodes are connected by five different types of edges. The purpose of an edge is to model a constraint on the time stamp of its head node. Each edge $e = (v, w)$ has two attributes. One attribute is a Boolean value, signifying whether this edge is currently active or not. The other attribute $\tau(e)$ denotes a point in time which basically can be interpreted as a lower bound on the time stamp of its head node w, provided that the edge is currently active.

- *Schedule edges* connect schedule nodes to departure or arrival nodes. They carry the planned time for the corresponding event of the head node (according to the published schedule). Edges leading to departure nodes are always active, since a train will never depart prior to the published schedule.
- *Forecast edges* connect forecast nodes to departure or arrival nodes. They represent the time stored in the associated forecast node. If no forecast for the node exists, the edge is inactive.
- *Standing edges* connect arrival events at a certain station to the following departure event of the same train.
  They model the condition that the arrival time of train t at station s plus its minimum standing time $stand(s, t)$ must be respected before the train can depart (to allow for boarding and deboarding of passengers). Thus, for a standing edge e, we set $\tau(e) = arr(s, t) + stand(s, t)$. Standing edges are always active.
- *Traveling edges* connect a departure node of some train t at a certain station s to the very next arrival node of this train at station s'. Let $dep(s, t)$ denote the departure time of train t at station s and $tt(s, s', t)$ the travel time for train t between these two stations. Then, for edge $e = (s, s')$, we set $\tau(e) = dep(s, t) + tt(s, s', t)$. These edges are only active if the train currently has a secondary delay (otherwise the schedule or forecast edges provide the necessary conditions for its head node).

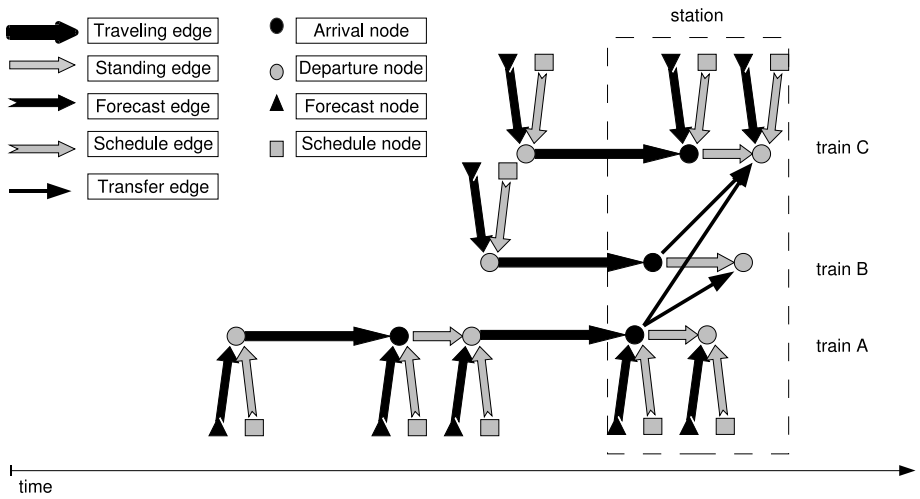


Fig. 2. Illustration of the dependency graph model.

Due to various, mostly unknown factors determining the speed of trains in a delayed scenario, e.g. speed of train, condition of the track, track usage (by other trains and freight trains that are not in the available schedule), used engines with acceleration/deceleration profiles, signals along the track etc. we assume for simplicity that $tt(s, s', t)$ is the time given in the planned schedule.

– *Transfer edges* connect arrival nodes to departure nodes of other trains at the same station, if there is a planned transfer between these trains. Thus, if f is a potential feeder train for train t at station s, we set $\tau(e) = wait(t, s, f)$, where

$$wait(t, s, f) = \begin{cases} arr(s, f) + minct(s, f, t) & \text{if } t \text{ waits for } f \\ 0 & \text{otherwise} \end{cases}$$

(cf. Section 2.2) if we respect the waiting rules. Recall that t waits for f only if the following equation holds

$$arr(s, f) + minct(s, f, t) - dep_{sched}(s, t) < wt(t, s, f)$$

or we have an explicit connection status decision that t will wait.
By default these edges are active. In case of an explicit connection status decision "will not wait" we mark the edge in the dependency graph as not active and ignore it in the computation.
For an "always wait" or "never wait" scenario we may simply always return the resulting delayed departure time or zero, respectively.

5.2 Computation on the Dependency Graph

The current time stamp for each departure or arrival node can now be defined recursively as the maximum over all deciding factors: For a departure of train t at station s with feeders $f_1, \dots f_n$ we have $dep(s,t) =$

$$\max\{dep_{sched}(s,t), dep_{for}(s,t), arr(s,t) + stand(s,t), max_{i=1}^{n}\{wait(t,s,f_i)\}\}.$$

For an arrival we have

$$arr(s,t) = \max\{arr_{sched}(s,t), arr_{for}(s,t), dep(s',t) + tt(s',s,t)\}$$

with the previous stop of train t at station s'. Inactive edges do not contribute to the maximum in the preceeding two equations.

If we have a status message that a train has finally departed or arrived at some given time dep_{fin} resp. arr_{fin}, we do not longer compute the maximum as described above. Instead we use this value for future computations involving this node.

We maintain a priority queue (ordered by increasing time stamps) of all nodes whose time stamps have changed since the last computation was finished. Whenever we have new forecast messages, we update the time stamps of the forecast nodes and, if they have changed, insert them into the queue. As long as the queue is not empty we extract a node from the queue and update the time stamps of the dependent nodes (which have an incoming edge from this node). If the time stamp of a node has changed in this process, we add it to the queue as well.

For each node we keep track of the edge e_{max} which currently determines the maximum so that we do not need to recompute our maxima over all incoming edges every time a time stamp changes. Only if $\tau(e_{max})$ was decreased or $\tau(e)$ for some $e \neq e_{max}$ increases above $\tau(e_{max})$ the maximum has to be recomputed.

- If $\tau(e)$ decreases and $e \neq e_{max}$ nothing needs to be done.
- If $\tau(e)$ increases and $e \neq e_{max}$ but $\tau(e) < \tau(e_{max})$ nothing needs to be done.
- If $\tau(e)$ increases and $e = e_{max}$ the new maximum is again determined by e_{max} and the new value is given by the new $\tau(e_{max})$.

When the queue is empty, all new time stamps have been computed and the nodes with changed time stamps can be sent to the search graph update routine.

6 Search Types

Most timetable information systems consider a pre-trip scenario: The user is at home and requests a connection from station s_1 to s_2 departing or arriving around some time τ or inside an interval $[\tau_1, \tau_2]$. In such a scenario, it is important that the search delivers all attractive connections with respect to several criteria which suit the query. Even if you use information systems at a station or click "Right-now" in an online system you will usually be offered several alternatives.

In an *on-trip* scenario one is much closer to an earliest arrival problem. We differentiate two cases of the on-trip search:

1. A customer is at a certain station and wants to travel right now. Either he comes without a travel plan (for example, he was unable to plan the end of some meeting) or he may have just missed a connecting train.
2. The customer sits already in a train and wants to search for alternatives, for example, because he has been informed that a connecting train will be missed.

In both cases travelers want to reach their destination as fast and convenient as possible. In case of delays many railways even remove restrictions on train-bound tickets, so it might be possible to completely forget about ticket costs, since the ticket is already paid and the passenger may use any means of transportation available. If there is a restriction like "no high speed train" (like the German ICE or French TGV) which is not revoked, an on-trip search with train category restrictions should be supported.

On-trip search at a station. In the example above one would not want to spend too much time at a station to shorten the traveling time measured from the departure with the first used train to the arrival at the destination (as calculated in the pre-trip scenario), instead the total travel time counting from "now" is one of the optimization goals. However, in the presence of delays it may become more important to search for reliable connections.

On-trip search in a train. In case the user currently travels in a train the on-trip search is different from the scenario at a station. Instead of leaving the train and standing at a station with the connecting train long gone (or canceled), we can do much better if we know of this problem in advance. Interesting alternatives may either leave the train before arriving at the station where the connection breaks, or stay longer in the train to change trains at a subsequent station.

Realization. Both on-trip searches can be realized in our timetable information system using different starting events. Instead of creating start labels for all departures in the departure interval (for forward search), we either

- create only a single start label at the change level of the source station and count time including the waiting time before taking any train (on-trip station), or
- create only a single start label at the arrival station of the train edge the traveler uses when receiving the information about a connecting train that will be missed (on-trip train).

Note that in the on-trip train case, using the arrival node of the train instead of any of the departure nodes, the modeling of interchanges in the time expanded graph guarantees that only valid train changes at the first stop after receiving the information are used. It would not be feasible to solve the on-trip train case with

a single departure at that station, because we need to ensure that the departure of the train with which one arrives, all departures below the station dependent change time (through special interchange rules) and all later departures are considered and all but the first case are counted as an additional interchange. Thus quite a lot of different departures with differing values for elapsed time at start and number of interchanges used so far would have to be considered.

7 Evaluation of the Prototype

We implemented the dependency graph and the update algorithm described in Section 5 and extended our time table information system MOTIS to support updating the search graph (cf. Section 4). Although these update operations are quite costly, we give a proof of concept and show that they can be performed sufficiently fast for a system with real-time capabilities.

Our computational study uses the German train schedule of 2008. During each operating day all trains that pass various trigger points (stations and important points on tracks) generate status messages. There are roughly 5000 stations and 1500 additional trigger points. Whenever a train generates a status message on its way, new predictions for the departure and arrival times of all its future stops are computed and fed into a data base. German railways Deutsche Bahn AG provided delay and forecast data from this data base for a number of operation days. The simulation results for these days look rather similar without too much fluctuation neither in the properties of the messages nor in the resulting computational effort. In the following, we present results for a standard operating day with an average delay profile.

To test our system, we used five sets of waiting profiles. Basically, the train categories were divided into five classes: high speed trains, night trains, regional trains, urban trains, and class "all others." Waiting times are then defined between the different classes as follows:

- *standard* High speed trains wait for each other 3 minutes,
  other trains wait for high speed trains, night trains, and trains of class "all others" 5 minutes,
  night trains wait for high speed and other night trains 10 minutes, and 5 minutes for class "all others."
- *small* All times of scenario standard are halved, but night trains do not wait for train class "all others."
- *double* All times of scenario standard are doubled.
- *all5* All times of scenario standard are set to five minutes, in addition regional trains wait 5 minutes for all but urban trains.
- *extreme* All times of the previous scenario are doubled.

It is important to keep in mind that the last two policies are far from reality and are intended to strain the system beyond the limits it was designed to handle. For each of these different waiting profiles we tested different maximum distances of feeding and connecting trains $\delta \in \{5, 15, 30, 45, 60\}$, with one hour

search graph	
event nodes	1.0 mil
change nodes	0.8 mil
edges	2.2 mil

dependency graph	
events	977,324
standing edges	449,575
driving edges	488,662

Table 1. Properties of our search graph (left) and dependency graph (right).

being the periodicity for most types of trains, and compare them to a variant without waiting for different trains (policy *no wait*). In this reference scenario it is still necessary to propagate delays in the dependency graph to correctly update the train runs. Thus the same computations as with waiting policies is carried out, only the terms for feeding trains are always zero.

We constructed a search and dependency graphs from the real schedule consisting of 37,000 trains operating on the selected day. The number of nodes and edges in both graphs are given in Table 1. There is one event node, one schedule node and one forecast node per train event in the dependency graph, the number of forecast and schedule edges equals the number of events, too. The number of standing and traveling edges are in one to one correspondence to the stay-in-train and train edges of the search graph. The number of feeding edges depends on the waiting policy and δ and can be found in the eighth column of Table 2. There is a monotonous growth in the number of transfer edges depending on the parameter δ. Additionally, the number of these edges increase as more trains wait for other trains because of the additional rules for scenarios with more rules.

For the chosen simulation day we have a large stream of real forecast messages. Whenever a complete sequence of messages for a train has arrived, we send them to the dependency graph for processing. 340,495 sequences containing a total of 6,211,207 forecast messages are handled. Of all messages 2,471,582 forecasts are identical to the last forecasts already processed for their nodes. The remaining 3,739,625 messages either trigger computations in the dependency graph or match the current time stamp of the node. The latter require neither shifting of nodes nor a propagation in the dependency graph. The resulting number of node shifts is given in the seventh column of Table 2.

At the end of the day 596,496 nodes have received at least one forecast. For 265,544 nodes the forecast differs from the scheduled time although there are 3,287,834 forecasts differing from the scheduled time for the event. Note that the last number is much higher as trains whose prediction changes produce new messages each time. A train with a large number of stops and a long travel time thus can generate a large number of messages.

In Table 2 we give the results for our test runs for the different policies and values of δ. All experiments were run on a standard PC (AMD Athlon 64 X2 4600+ 2.4 GHz with 4GB of RAM). The key figures for required computations, stations with a delayed event and node shifts increase when changing to policies for which trains wait longer or more trains have to wait. Increasing δ yields a higher effect the more trains wait. The overall small impact of changing δ is due to the majority of delays being rather small. Only for the less realistic scenarios

we notice a significant growth in all key criteria when increasing δ from 5 to 15, whereas all policies behave rather similarly for $\delta = 5$.

Amongst the plausible policies there is only a 11% difference in the number of moved nodes. It nearly doubles going to policy *all5* and even increases by a factor of 3.4 towards policy *extreme*. The increase in running time spent in the search graph is equivalent. Of our simulation time roughly 3 minutes are spent extracting and preprocessing the messages from the forecast stream. This time is obviously independent of the test scenario. Interestingly, the time spent in the dependency graph seems to be only minimally affected by exchanging the profiles against those that incur more computations and node shifts. As the

Instance		computation time for				Number of		Delayed at	
policy	δ	SG	DG	IO	total	shifts	feeding	end of day	
	in min	in s	in s	in s	in s	executed	edges	nodes	stations
no wait	-	807	133	177	1118	3,165,614	0	357,972	5,467
	5	819	136	162	1118	3,253,980	8,792	359,105	5,511
	15	860	137	167	1164	3,416,718	55,218	364,961	5,664
small	30	871	137	163	1171	3,430,189	124,141	365,179	5,664
	45	875	139	157	1171	3,432,189	207,855	365,206	5,664
	60	869	137	161	1167	3,434,041	267,638	365,231	5,664
	5	817	135	170	1122	3,254,013	8,792	359,105	5,511
	15	876	136	169	1180	3,426,272	55,284	365,110	5,711
standard	30	880	139	170	1189	3,445,019	124,305	365,353	5,723
	45	878	138	158	1175	3,454,623	208,127	365,395	5,733
	60	917	148	169	1234	3,460,210	268,002	365,452	5,738
	5	813	133	164	1110	3,265,175	8,792	359,254	5,511
	15	917	137	162	1216	3,557,572	55,284	367,171	5,731
double	30	931	136	171	1238	3,617,603	124,305	367,590	5,770
	45	959	136	160	1255	3,646,080	208,127	367,863	5,782
	60	979	137	161	1277	3,661,137	268,002	367,995	5,787
	5	830	137	178	1,145	3,419,161	16,261	366,372	5,815
	15	1,761	141	174	2,076	6,994,379	168,849	404,336	6,541
all5	30	1,776	146	157	2,078	7,095,897	400,114	405,827	6,557
	45	1,796	148	166	2,110	7,112,681	665,811	406,214	6,561
	60	1,793	150	170	2,113	7,121,351	874,649	406,433	6,561
	5	815	137	173	1,124	3,446,965	16,261	367,303	5,818
	15	3,090	159	175	3,424	12,090,373	168,849	422,119	6,648
extreme	30	3,111	164	178	3,453	12,155,547	400,114	434,040	6,676
	45	3,134	170	177	3,480	12,257,936	665,811	438,645	6,684
	60	3,306	178	179	3,663	12,285,623	874,649	440,233	6,684

Table 2. Computation time (propagation in the dependency graph (DG) and update of the search graph (SG), IO and total) and key figures for the number of feeding edges, node shifts in the search graph and the number of nodes and stations with delay at the end of the day with respect to different waiting policies.

overall running time is by far dominated by the reconstruction work in the search graph we would rather try to improve the performance there, if necessary.

Even for the most extreme scenario a whole day can be simulated in one hour. The overall simulation time for realistic policies lies around 20 minutes. For the policy *standard* with $\delta = 45$ we are below 1/5ms ($189\mu s$) per message, at a rate of less than 75 messages arriving per second. This clearly qualifies for live performance.

8 Conclusions and Future Work

We have built a first prototypal system which can be used for efficient off-line simulation with massive streams of delay and forecast messages for typical days of operation within Germany.

It remains an interesting task to implement a live feed of delay messages for our timetable information system and actually test real-time performance of the resulting system. Since update operations in the time-dependent graph model are somewhat easier than in the time-expanded graph model, we also plan to integrate the update information from our dependency graph into a multi-criteria time-dependent search approach developed in our group (Disser et al. [10]).

Additionally, we would be interested in looking into speed-up techniques for dynamic scenarios in the multi-criteria case. Since most of the existing speed-up techniques focus on single criterion search, time-less edges, and usually require bidirectional search, this is not at all easy in the multi-criteria static scenario without delays. The recent SHARC algorithm [11,12] is a powerful speed-up technique for uni-directional search which seems to be a promising candidate to generalize to a dynamic scenario and multiple search criteria.

Acknowledgments

This work was partially supported by the DFG Focus Program Algorithm Engineering, grant Mu 1482/4-1. We wish to thank Deutsche Bahn AG for providing us timetable data and up-to-date status information for scientific use, and Christoph Blendinger and Wolfgang Sprick for many fruitful discussions.

References

1. Müller-Hannemann, M., Schulz, F., Wagner, D., Zaroliagis, C.: Timetable information: Models and algorithms. In: Algorithmic Methods for Railway Optimization. Volume 4395 of Lecture Notes in Computer Science, Springer Verlag (2007) 67–89
2. Müller-Hannemann, M., Schnee, M.: Finding all attractive train connections by multi-criteria Pareto search. In: Proceedings of the 4th ATMOS workshop. Volume 4359 of Lecture Notes in Computer Science, Springer Verlag (2007) 246–263
3. Delling, D., Giannakopoulou, K., Wagner, D., Zaroliagis, C.: Timetable Information Updating in Case of Delays: Modeling Issues. Technical report, ARRIVAL (2008)

4. Gatto, M., Glaus, B., Jacob, R., Peeters, L., Widmayer, P.: Railway delay management: Exploring its algorithmic complexity. In: Algorithm Theory — SWAT 2004. Volume 3111 of Lecture Notes in Computer Science, Springer (2004) 199–211
5. Gatto, M., Jacob, R., Peeters, L., Schöbel, A.: The computational complexity of delay management. In: Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 05). Volume 3787 of Lecture Notes in Computer Science, Springer (2005) 227–238
6. Ginkel, A., Schöbel, A.: The bicriteria delay management problem. Transportation Science **41** (2007) pp. 527–538
7. Schöbel, A.: Integer programming approaches for solving the delay management problem. In: Algorithmic Methods for Railway Optimization. Volume 4359 of Lecture Notes in Computer Science, Springer (2007) 145–170
8. Meester, L.E., Muns, S.: Stochastic delay propagation in railway networks and phase-type distributions. Transportation Research Part B **41** (2007) 218–230
9. Anderegg, L., Penna, P., Widmayer, P.: Online train disposition: to wait or not to wait? ATMOS'02, ICALP 2002 Satellite Workshop on Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science **66** (2002)
10. Disser, Y., Müller-Hannemann, M., Schnee, M.: Multi-criteria shortest paths in time-dependent train networks. In: WEA 2008. Volume 5038 of Lecture Notes in Computer Science, Springer (2008) 347–361
11. Bauer, R., Delling, D.: SHARC: Fast and Robust Unidirectional Routing. In: Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX'08), SIAM (2008) 13–26
12. Delling, D.: Time-Dependent SHARC-routing. In: ESA 2008. Volume 5193 of Lecture Notes in Computer Science, Springer (2008) 332–343

Engineering Time-Expanded Graphs for Faster Timetable Information*

Daniel Delling, Thomas Pajor, and Dorothea Wagner

Department of Computer Science, University of Karlsruhe, P.O. Box 6980, 76128 Karlsruhe, Germany. {delling,pajor,wagner}@informatik.uni-karlsruhe.de

Abstract. We present an extension of the well-known time-expanded approach for timetable information. By remodeling unimportant stations, we are able to obtain faster query times with less space consumption than the original model. Moreover, we show that our extensions harmonize well with speed-up techniques whose adaption to timetable networks is more challenging than one might expect.

1 Introduction

During the last years, many speed-up techniques for computing a shortest path between a given source s and target t have been developed. The main motivation is that computing shortest paths in graphs is used in many real-world applications like route planning in road networks or timetable information for railways. Although DIJKSTRA's algorithm [5] can solve this problem, it is far too slow to be used on huge datasets. Thus, several speed-up techniques have been developed (see [4] for an overview) yielding faster query times for typical instances. However, recent research focused on developing speed-up techniques for road networks, while only few work has been done on adapting techniques to graphs deriving from timetable information systems. In general, two approaches exist for modeling timetable information: The time-dependent and time-expanded approach. While the former yields smaller inputs (and hence, smaller query times), the latter allows a more flexible modeling of additional constraints. It turns out that adaption of speed-up techniques to each of these models is more challenging than one might expect.

In this work, we use a different approach for obtaining faster query times. Instead of applying a routing algorithm, e.g., plain DIJKSTRA, on the original model, we improve the *time-expanded* model itself in such a way that a routing algorithm does not exploit parts of the graph not necessary for solving the earliest arrival problem (EAP). Interestingly, it turns out that those optimizations are included in the time-dependent approach implicitely. By introducing those techniques to the time-expanded approach, query times for the time-expanded approach are comparable to the time-dependent approach.

* Partially supported by the DFG (project WAG54/16-1) and the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1582

1.1 Related Work

The simple, i.e., without realistic transfers, time-expanded model has been introduced in [21]. The model has been generalized in [18] in order to deal with realistic transfers. Since then, this realistic model has been used for many experimental studies, e.g., [14, 19, 2]; most of them focusing on faster speed-up techniques or multi-criteria optimization for timetable information. However, [21] enriched the simple time-expanded graph by shortcuts and [19] introduced minor changes to the time-expanded model itself by removing unnecessary nodes with outgoing degree 1.

1.2 Our Contributions

This paper is organized as follows. Section 2 includes formal definitions and a review of the time-expanded model for timetable information. Our main contribution is Section 3. We show how the main ingredient for high-performance speed-up techniques in road networks, i.e., *contraction*, can be adapted to time-expanded graphs. Unfortunately, it turned out that this contraction yields a tremendous growth in number of edges (unlike in road networks). However, by changing the modeling of unimportant stations, a DIJKSTRA does not exploit unnecessary parts of the network. The key observation is the following. Assume T is a station with only one line stopping. A passenger traveling via T only leaves the train if T is her target station, otherwise it never pays off to leave the train. Moreover, we are able to generalize this approach to stations with more lines stopping at that station. In Section 4 we introduce a new speed-up technique tailored to time-expanded graphs based on blocking certain connections. Furthermore, we show how existing techniques have to be adapted to timetable graphs. It turns out that certain pitfalls exist that one might not expect. However, those adapted techniques harmonize well with our new approaches, which we confirm by an experimental evaluation in Section 5. We conclude our work in Section 6 with a summary and future work.

2 Preliminaries

Throughout the whole work, we restrict ourselves to the earliest arrival problem (EAP), i.e., find a connection in a timetable network with lowest travel time. In the following we often call this single-criteria search in contrast to multi-criteria search that also minimizes number of transfers and further criteria [14, 19].

Moreover, we restrict ourselves to simple, directed graphs $G = (V, E, \text{length})$ with positive length function $\text{length} : E \to \mathbb{R}^+$. The reverse graph $\overline{G} = (V, \overline{E})$ is the graph obtained from G by substituting each $(u,v) \in E$ by (v,u). A *partition* of V is a family $\mathscr{P} = \{P_0, P_1, \ldots, P_k\}$ of sets $P_i \subseteq V$ such that each node $v \in V$ is contained in exactly one set P_i. An element of a partition is called a *cell*. The *boundary nodes* B_P of a cell P are all nodes $u \in P$ for which at least one node $v \in V \setminus P$ exists such that $(v,u) \in E$ or $(u,v) \in E$.

The Condensed Model is the easiest approach for modeling timetable information. Here, a node is introduced for each station and an edge is inserted iff a direct connection

between two stations exists. The edge weight is set to be the minimum travel time over all possible connections between these two stations. Unfortunately, several drawbacks exist. First of all, this model does not incorporate the actual departure time from a given station. Even worse, travel times highly depend on the time of the day and the time needed for changing trains is also not covered by this approach. As a result, the calculated travel time between two arbitrary stations in such a graph is only a *lower bound* of the real travel time. However, in Section 4 we show that the condensed model is helpful for certain speed-up techniques.

The (Realistic) Time-Expanded Model. Throughout this work, we use the realistic time-expanded model allowing realistic queries. Therefore, three types of nodes are used to represent certain events in the timetable. *Departure* and *arrival nodes* are used to model elementary connections in the timetable. Thus, for each elementary connection $c \in \mathscr{C}$ one arrival and departure node is created and an edge is inserted between them. To model transfers, *transfer nodes* are introduced. For each departure event one transfer node is created which connects to the respective departure node having weight 0. To ensure a minimum transfer time $\textsc{transfer}(S)$ at a specific station S, an edge from each arrival node u is inserted to the smallest (considering time) transfer node v where $\Delta(\textsc{time}(u), \textsc{time}(v)) \geq \textsc{transfer}(S)$. Here $\Delta(\cdot,\cdot)$ denotes the time difference between two points in time and $\textsc{time} : V \to \mathscr{T}$ maps each node to its timestamp with respect to the timetable. Due to the periodic nature of our timetables Δ is defined by

$$\Delta(t_1,t_2) := \begin{cases} t_2 - t_1 & \text{if } t_2 \geq t_1, \\ t_2 + 1440 - t_1 & \text{otherwise.} \end{cases}$$

To ensure the possibility to stay in the same train when passing through a station, an additional edge is created which connects the arrival node with the appropriate departure node belonging to this same train. Further to allow transfers to an arbitrary train, transfer nodes are ordered non-decreasing. Two adjacent nodes (w.r.t. the order) are connected by an edge from the smaller to the bigger node. Furthermore, to allow transfers over midnight, an overnight-edge from the biggest to the smallest node is created. For further details, see [19].

For each edge $e = (u,v)$ in the expanded graph the weight $w(e)$ is defined as the time difference $\Delta(\textsc{time}(u), \textsc{time}(v))$ of the nodes the edge connects. Hence, we call the graph consistent in time, meaning for each path from u to v in the graph, the sum of the edge weights along the paths is equal to the time difference $\Delta(\textsc{time}(u), \textsc{time}(v))$.

For future considerations the following notation will be helpful. Let $\prec \subseteq V \times V$ be a relation which compares two events in time. Since in the expanded model nodes correspond to events with a certain timestamp, our relation is defined on the set of nodes of the graph. We say for two nodes $u, v \in V$ that $u \prec v$ if the event of u is happening *before* the event of v. Please note that it cannot be determined for u and v if $u \prec v$ just by comparing $\textsc{time}(u)$ and $\textsc{time}(v)$ due to the periodic nature of the timetable and the fact that times are always expressed in minutes after midnight. If for example $\textsc{time}(u) = 400$ and $\textsc{time}(v) = 600$ there are two possibilities. Either $u \prec v$ with $\Delta(u,v) = 200$ or $v \prec u$ with $\Delta(v,u) = 1640$. As a consequence, the Δ function applied to a tuple (u,v) is *only* valid if $u \prec v$.

3 Engineering the Time-Expanded Model

In this section, we present approaches how to enhance the classical time-expanded model. Our first attempt applies a technique deriving from road networks, i.e., contraction, to railway graphs. However, it turns out that this approach yields a too high number of edges. Hence, we also introduce the *Route-Model* which changes the modeling of "unimportant" stations.

3.1 Basic Contraction

All speed-up techniques developed during the last years have one thing in common. During preprocessing they apply a contraction routine, i.e., a process that removes unimportant nodes from the graph and adds shortcuts to the graph to keep the distances between the remaining nodes correct. Interestingly, the fastest hierarchical technique for routing in road networks, Contraction Hierarchies [6], relies *only* on such a routine. The key observation is that in road networks, the average degree of remaining nodes does *not* explode.

At a glance, one could be optimistic that contraction also works well in railway networks. Like in road networks, some nodes in time-expanded graphs are more important than others. However, contraction does not exploit the special structure of time-expanded timetable graphs. For example, departure nodes have an outgoing degree of 1. Thus, we can safely remove such nodes and add a shortcut between the corresponding transfer and arrival node. More precisely, we propose a new contraction routine consisting of three steps. In the following we explain each step separately.

Omitting Departure Nodes The first step of our contraction routing bypasses *all* departure nodes. In [19], the authors state that departure nodes can be omitted in time-expanded graphs which can be interpreted as bypassing those nodes.

Omitting Arrival Nodes In a second step, we bypass *all* arrival nodes within the network. As a consequence, the degree of transfer nodes highly increases. By these two steps we reduce the number of nodes by approximately a factor of 3. However, the graph still contains all original transfer nodes of which some are more important than others.

Bypass Transfer Nodes The final step of our contraction bypasses nodes according to their degree. We bypass nodes with low degree first yielding changes in the degree of its neighbors. Our contraction ends if all transfer nodes have a total degree at least of δ, which is a tuning parameter. We suggest to use a min-heap to determine the next node to be bypassed. The key of a node x shall be $\deg_{in}(x) + \deg_{out}(x)$.

Note that we need not apply all three steps. While the first step reduces both number of nodes and edges, the following two steps yield higher edge counts. In the following, we call a time-expanded model with shortcut departure nodes, the *phase 1* model. The *phase 2* model has neither arrival nor departure nodes. If we also remove (some) transfer nodes, we call the resulting graph a *phase 3* graph. For an experimental evaluation of this contraction routine, see Section 5.

3.2 Route-Model

In our experimental studies, it turned out that our contraction routine from the last section suffers from a dramatic growth in number of edges. Already our phase 2 model has up to 3.6 times more edges than the original graph (cf. Section 5). Hence, we here introduce a different approach, called the *route model*. In contrast to contraction, we exploit certain semantic properties of the time expanded graph regarding transferring which eventually leads to a reduction of the number of shortest paths. The classic time-expanded model allows transfers at a station from each arriving train to *all* subsequent departing trains. However, when planning an itinerary by hand, we would probably do the following intuitive pruning: During the way from the source to the target station assume we find a route which leads to some station S on the way, arriving there at time t_S. Then, we would not need to examine paths toward station S with an arrival time $t'_S > t_S$, since computing these paths is redundant as we already arrived at S earlier, and we could achieve the same result by taking the earlier computed path arriving at S at t_S and then waiting at S until t'_S. This observation is the basic idea behind the route model.

Remodeling of Stations. The modifications to the (original realistic) time-expanded graph are done locally and independently for each station S, and involve the following three steps:

1. Remove all outgoing edges from all arrival nodes. This includes edges to transfer nodes as well as edges to the departure node of the same train.
2. Insert a minimal number of new transfer-edges directly from the arrival nodes to departure nodes. This allows us to model transfers more specific without losing any optimal shortest paths in comparison to the original time expanded model.
3. Keep the transfer nodes and their interconnecting edges as well as departure-edges from transfer to departure nodes. Although, there are no more edges in the graph to get from an arrival node to a transfer node, the transfer nodes are still used as source nodes for the actual DIJKSTRA query.

The only non-trivial modification is the second one, where for each arrival node we need to find a minimal set of departure nodes which shall become reachable from the particular arrival node. For that reason let S be the currently considered station and $\mathcal{N}_S$ all *neighbors* of S. A station $T \in \mathcal{N}_S$ is called a neighbor of S if at least one elementary connection from S to T exists. Thus, we can speak of *routes* between S and each neighbor from $\mathcal{N}_S$. We now use the following notation. u denotes an arbitrary but fixed arrival node of S from which outgoing edges are inserted. v denotes the departure node toward which the edges (u,v) are inserted. Furthermore, w denotes the arrival node corresponding to the elementary connection to which the departure node v belongs. The basic idea is to insert (at least) one edge per route toward a departure node belonging to the the particular route. So, let us consider some fixed station $T \in \mathcal{N}_S$ with $T \neq R$ where R is the station where we just came from through u. Of all departure nodes v belonging to an elementary connection (v,w) from S to T we insert an edge (u,v) in S according to the following criteria.

1. The node w is the smallest (regarding time) possible (meaning it is not in violation with the second criterion) arrival node at T that is after u, i.e. $w \succ u$.

2. The node v respects the transfer time criterion at S. For that reason it has to hold that $v \succ u + \text{TRANSFER}(S)$ if u and v belong to different trains, or $v \succ u$ if they share the same train.

Obviously, by this strategy we select the edge (u,v) according to the earliest possible arrival event at the *target station* T. This yields a transfer to a train which arrives at T by the earliest possible time. Note that if we instead would have chosen v according to the earliest possible departure node at S, we could have missed a different train that departs at S later, but arrives at T earlier. Such a scenario is called overtaking of trains. Also note, that if the train belonging to u utilizes the route toward station T, it does not necessarily have to be the case, that the inserted edge (u,v) corresponds to the departure event of that specific train. It simply corresponds to the train arriving at T first, which may well be a different train.

Transfer Times at Neighboring Stations. While we did respect the transfer time criterion of S, we also have to respect the transfer time criterion at T. Figure 1 shows why this is important.

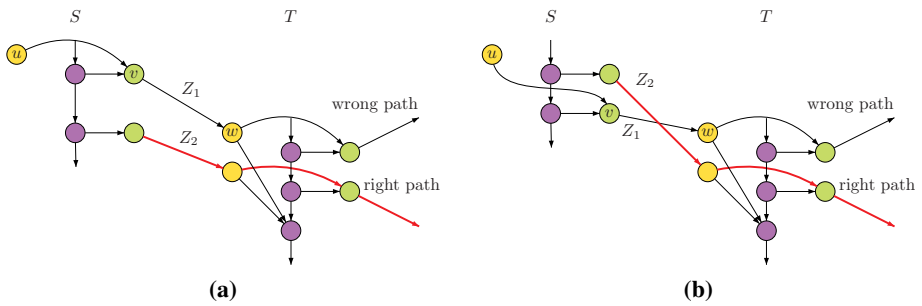


Fig. 1: Two problems concerning the transfer time criterion at station T.

On the left side the train Z_2 arriving at T just slightly after Z_1 is the optimal path, but it can not be transferred to, because at S we only chose Z_1 and at T the transfer time is too big to reach it from Z_1. On the right picture the scenario is even worse. While the train Z_1 is the earliest train regarding the arrival time at T, the optimal route again contains Z_2 which departs at S earlier than Z_2, but it is not reachable because it arrives at T slightly after Z_1. Again the transfer time at T is too big to enter Z_2 at T. In both cases we have to ensure that Z_2 can be entered somewhere. Since our modifications should remain local in the sense that modifications at S should not involve modifications at some other stations, we ensure that Z_2 can be reached at S.

By adding some more edges to the graph, we are able to allow those connections as well. Let w_{earl} denote the earliest arrival node at T as computed before. Then, we insert edges (u,v) (belonging to connections (v,w)) satisfying the following properties.

1. Consider all trains arriving after w_{earl} but no later than the transfer time at T, meaning $w \succ w_{\text{earl}}$ and $w \prec w_{\text{earl}} + \text{TRANSFER}(T)$.

2. Still respect the transfer time criterion at S, i.e. $v \succ u + \text{TRANSFER}(S)$ if u and v belong to different trains and $v \succ u$ otherwise.

This routine ensures that (a) it is possible to arrive at T as early as possible and (b) all trains that go through T within the margin between the earliest arrival time and the transfer time at T can be reached by entering them at S.

Uncommon Routes. Despite these modifications, we additionally have to deal with another phenomenom in railway networks. In very few cases, it might pay off to use an itinerary with a sequence of stations $R \to S \to T \to S \to R'$ instead of $R \to T \to R'$. This odd situation may arise if T and S are close to each other, a train runs from R to T, another from T to R', and $\text{TRANSFER}(S) < \text{TRANSFER}(T)$ holds. Figure 2 gives an example.

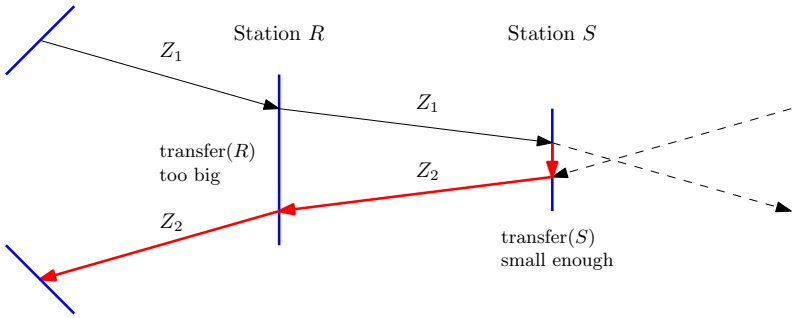


Fig. 2: Situation where it is necessary to go forth and back along the same route in order to transfer to train Z_2.

Our Route-Model does not allow such connections. However, we may overcome this problem by introducing edges at arrival nodes u of S toward departure nodes leading back to R if and only if the following inequation holds:

$$\kappa_{R,S} + \kappa_{S,R} + \text{TRANSFER}(S) < \text{TRANSFER}(R).$$

Here $\kappa_{R,S}$ denotes the best lower bound regarding travel time from R to S. By this we ensure that no shortest paths get lost while in most cases we still get the advantage of prohibiting cycles along the same route. Please note, that we can not rule out cycles such as $\cdots \to R \to S \to T \to R \to \cdots$, however cycles of this type occur less often in general timetable networks.

Leaving Big Stations Untouched. It turns out that remodeling of stations with many neighbors, e.g., major train hubs, lead to a disproportionately high increase in additional edges, since for each neighbor (route) at least one edge must be inserted for each arriving train. In the original time expanded model, however, at most two edges existed for each arrival node (arrival-transfer and arrival-departure). Since our modifications

are only local we can choose for each station individually whether we want to convert it to the Route-Model or not. For that reason we introduce a tuning parameter γ indicating that stations with more neighbors than γ should be left untouched. Hence, changing γ yields a trade-off between a speed-up regarding the number of touched nodes against an increasing size of the edge set of the graph.

A problem that arises when mixing Route-Model stations with classic stations is that the main advantage of the Route-Model—subsequent connections on the same route are not visited during the DIJKSTRA search—may fade. Analyzing the example in Figure 3, we observe a big station which has not been converted followed by a route containing a few small stations. While at the small stations no connections exist between connections of the same route, they are nevertheless visited, because they are all accessible through the big station. Hence, we developed *Node-Blocking* which adopts the idea behind the Route-Model as a speed-up technique, and blocks redundant connections of the same route, so they are not visited. This technique is explained in Section 4.

Theorem 1. *Applying* DIJKSTRA *on the Route-Model yields correct solutions to the earliest arrival problem.*

The proof of Theorem 1 can be found in the full paper.

4 Speedup Techniques

In principle, we could use DIJKSTRA's algorithm for solving EAP. However, plain DIJKSTRA visits unnecessary parts of the graph, even if we use our Route-Model. Hence, we introduce two approaches for obtaining faster query times. We adapt existing techniques—developed for road networks—to timetable graphs and introduce a new speed-up technique following the ideas from our Route-Model.

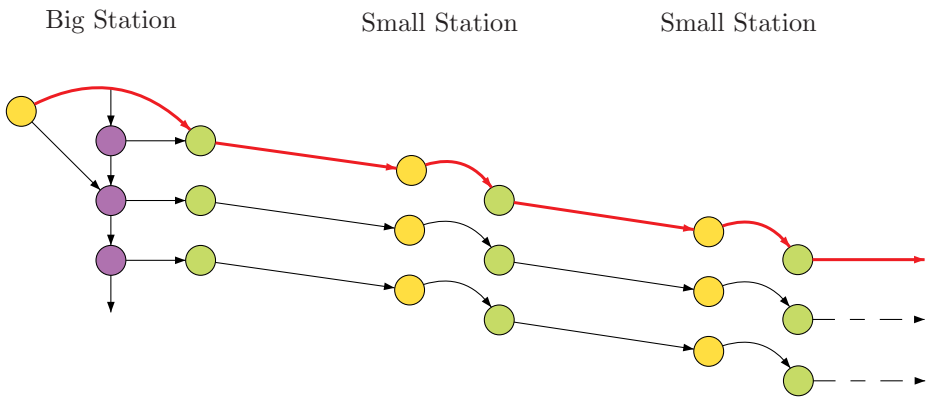


Fig. 3: When a big station which is not converted is visited during a DIJKSTRA query, all subsequent connections are visited as well, while only the red path should be relevant. Unimportant nodes are omitted in the figure.

4.1 Tailored Speed-Up Techniques

Node-Blocking is a speed-up technique tailored to time-expanded networks. It basically incorporates the ideas behind the Route-Model as described in Section 3.2: if we can reach a station S at some time t_S we try to prune paths reaching S at a later time $t'_S > t_S$. Recall that the Route-Model prunes the search by removing certain edges from the graph. Node-Blocking, on the contrary, achieves a similar result by dynamically blocking departure nodes during the DIJKSTRA query. The idea is as follows. If we visit a departure node v belonging to an elementary connection targeting some station T, we can *prune* all future departure nodes b targeting T.

Preprocessing. Formally, each departure node v of an elementary connection between two stations S and T induces a set B_v of blocked nodes. A node b is contained in B_v if and only if the following conditions hold.

1. b is a departure node at S belonging to an elementary connection targeting the same station T as v.
2. $b \succ v$ holds.
3. If w and c are the arrival nodes at T of the connections associated with v and b, respectively, then $w + \text{TRANSFER}(T) \prec c$ must hold, i.e., we respect the transfer time criterion at T.

Although the "blocked state" of each node is dynamic in the sense that it depends on the shortest path query, and therefore must be computed during the query, the set B_v of inducing blocked nodes can be precomputed for each node v by iterating through all departure nodes of the station and checking whether the above criteria apply to them.

Note that in contrast to the Route-Model, we do not have to deal with the transfer time criterion at S, since we only *block* nodes, and hence never allow a path to be taken which was forbidden by the transfer time criterion at S. In worst case, we block departure nodes which cannot be reached anyway due to the transfer time criterion of S. Moreover, all special cases are covered by our third condition.

Query. The modifications to standard DIJKSTRA algorithm are simple. We introduce an additional flag blocked(v) to all nodes of the graph, which is initialized to false. Then, whenever we try to insert a node v into the queue, we mark all nodes B_v as blocked. If v is marked as blocked, we prune the search.

Combination with Route Model. Although our Route-Model and Node-Blocking follow the same ideas, the advantage of the Route-Model is the lower computation-overhead during the query. However, as discussed in Section 3.2, it does not pay off to remodel major hubs. Hence, Node-Blocking harmonizes well with the Route-Model as we use Node-Blocking for pruning paths at such hubs.

Combination with Phase 1+ Models. Since from the Phase 1 model onwards departure nodes are removed, Node-Blocking has to be altered slightly to conform with these models. Instead of departure nodes blocking future departure nodes, we simply let the corresponding arrival nodes (belonging to the respective departure nodes) block each other. In this case, the arrival nodes assume the role of the previous departure nodes regarding blocking, which allows us to continue using the same query algorithm.

Theorem 2. *Applying Node-Blocking to* DIJKSTRA*'s algorithm yields correct solutions to the earliest arrival problem.*

The proof of Theorem 2 can be found in the full paper.

4.2 Adapting Speed-Up Techniques

Although the adaption of many techniques may be promising, we choose basic goal-directed techniques for adaption. It turned out that adaption of more sophisticated techniques, e.g., Highway Hierarchies [20], Contration Hierarchies [6], REAL [8], SHARC [1], is much more challenging than expected. The main reason are either the need of a bidirectional query algorithm or the bad performance of the contraction routine.

Arc-Flags. The classic Arc-Flag approach, introduced in [13, 12], first computes a partition $\mathscr{P}$ of the graph and then attaches a *label* to each edge e. A label contains, for each cell $P_i \in \mathscr{P}$, a flag $AF_{P_i}(e)$ which is $true$ if a shortest path to at least one node in P_i starts with e. A modified DIJKSTRA—from now on called Arc-Flags DIJKSTRA—then only considers those edges for which the flag of the target node's cell is $true$. The big advantage of this approach is its easy query algorithm. However, preprocessing is very extensive. The original approach grows a full shortest path tree from each boundary node yielding preprocessing times of several weeks for instances like the Western European road network. Recently, a new *centralized* approach has been introduced [11]. However, it turns out that this centralized cannot be used in time-expanded transportation networks due to memory consumption. Hence, we use the original approach of growing full shortest path trees from each node.

Adaption. The query algorithm can be adapted to time expanded railway graphs very easily. We only have to consider that the exact target node is unknown (just the target station is known). For that reason we simply abort the DIJKSTRA algorithm as soon as a node belonging to the target station is settled. The preprocessing of Arc-Flags, however, needs some extra attention. Since we do not know the exact target node in advance, we have to ensure that all nodes belonging to the same station also get the same cell-id of the partition assigned. For that reason, we simply compute the partition on the condensed graph and map it to the expanded graph by assigning for each node $v \in V$ the cell-id due to $\text{cell}(v) := \text{cell}(\text{STATION}(v))$.

Computing the backwards-shortest path trees from each boundary node of each cell can then be done as described in [13]. However, this approach yields a problem specific on time expanded graphs. Since the length of any path in the graph always corresponds to the time needed to travel between the beginning and ending event (node) of that particular path, any two different paths between the same nodes *always* have the same length. Therefore, the number of shortest paths (in fact, there are *only* shortest paths in time expanded graphs) is tremendous. Unfortunately, if we set flags to true for every path, we do not observe any speed-up (cf. Section 5). In order to achieve a speed-up we have to prefer some paths over others. We examine the following four reasonable strategies for prefering paths:

Hop Minimization. For two paths of equal length, choose the one that has less hops (nodes) on it. This approach is often used in road networks [1].

Transfer Minimization. Choose the path that has less transfers between trains. While this is a good strategy for querying, it sets too many arc-flags to true, since for different boundary nodes too many different paths lead a transfer-minimal connection.

Distance Minimization. Choose the path that is shorter (geographically).

Direct Geographical Distance. Choose the path whose direct geographical distance is closer to the source node of the shortest path tree, formally for some node v that is reached from u we choose the new predecessor according to

$$\text{pre}(v)_{\text{new}} := \underset{w \in \{u, \text{pre}(v)\}}{\text{argmin}} \{ \sqrt{\big(\text{coord}_x(w) - \text{coord}_x(s)\big)^2 + \big(\text{coord}_y(w) - \text{coord}_y(s)\big)^2} \},$$

where s is the source node of the shortest path tree. This optimization is very aggressive, as it leads to the same result for different boundary nodes of the same cell as often as possible.

Section 5 shows the huge difference in the query performance when the arc-flags are computed with different strategies. Note that we can optimize query times by setting as many flags as possible to false. However, we also loose the ability to choose the "best" path during the query (e.g. due to a minimal number of transfers, costs, etc.). This yields a trade-off between query time and the quality of the computed itineraries.

Arc-Flags and Node-Blocking. Unfortunately, Node-Blocking does not harmonize with Arc-Flags. This is due to the fact of Node-Blocking being a very aggressive technique, leaving only very few connection arcs per station and route accessible. The optimization criterion hereby, namely arriving as early as possible at the next station does not necessarily match with our path selection during Arc-Flags preprocessing. As a result, both techniques prune different shortest paths. A possible solution would be to adapt the path selection for Arc-Flags according to Node-Blocking. However, this turns out to be complicated as we have to grow shortest path trees on the reverse graph. Hence, this path selection strategy is not implemented yet.

ALT. Goal directed search, also called A^* [10], pushes the search towards a target by adding a *potential* to the priority of each node. The ALT algorithm, introduced in [7], uses a small number of nodes—so called *landmarks*—and the triangle inequality to compute such feasible potentials. Given a set $L \subseteq V$ of landmarks and distances $d(\ell, v), d(v, \ell)$ for all nodes $v \in V$ and landmarks $\ell \in L$, the following triangle inequations hold: $d(u,v) + d(v,\ell) \geq d(u,\ell)$ and $d(\ell,u) + d(u,v) \geq d(\ell,v)$. Therefore, $\pi(u,t) := \max_{\ell \in L} \max\{d(u,\ell) - d(t,\ell), d(\ell,t) - d(\ell,u)\}$ provides a lower bound for the distance $d(u,t)$ and, thus, can be used as a potential for u.

Adaption. The query algorithm is, again, straight forward to adapt to time-expanded railway graphs. Since the only difference to the standard DIJKSTRA algorithm is the key which is inserted into the priority queue, we can still simply abort the search as soon as a node of the target station gets settled. However, we cannot compute the landmarks

on the expanded graph directly since then we would have to know the target node t in advance. Hence, we compute the landmarks on the much smaller condensed graph which still yields feasible potentials because the edge weights in the condensed graph are defined as the lower bounds regarding travel time. The potential function π during the query is then computed as follows:

$$\pi(v) = \max_{\ell \in L} \max\{\text{dist}(\text{STATION}(v), \ell) - \text{dist}(T, \ell), \text{dist}(\ell, T) - \text{dist}(\ell, \text{STATION}(v))\},$$

where T is the target station of the query. We can think of this as using a "lower bound of a lower bound" of the shortest path.

Former studies revealed that the selection of landmark nodes is crucial to the performance of ALT. The quality of the lower bounds highly depends on the quality of the selected landmarks. Thus, several selection strategies exist. To this point, no technique is known how to pick landmarks yielding the smallest search space for random queries. Thus, several heuristics exist. The best are *avoid* and *maxCover*. The first tries to identify regions that are not well covered by landmarks while the latter is basically the avoid routine followed by a local optimization. For details, we refer to [9].

Due to the small size of the condensed networks, another strategy for obtaining potentials seems promising. For each query, we use the target station T as landmark and compute the distances of all stations to T on-the-fly. The advantage of this *dynamic-landmark-selection* is a tighter lower bound. However, we have to run a complete DIJKSTRA in the condensed graph for each query which can take more time than using worse lower bounds from landmarks during the query. Note that this approach for obtaining lower bounds for A* was already proposed in [14].

Combining Arc-Flags and ALT. In [16], we observed that Arc-Flags (with the direct geographical distance strategy) and ALT optimize in two different ways. While Arc-Flags prunes paths that lead to the wrong direction geographically, ALT optimizes in time in the sense that fast trains are preferred over slow trains. Fast trains (having less stops in between) tend to get near the target station faster, yielding a lower key in the priority queue regarding the lower bound function. For that reason, it is suggestive to examine the combination of the two speed-up techniques. The implementation is straight-forward, since Arc-Flags does not interfere with ALT—Arc-Flags simply ignores edges that do not have their appropriate flag set, and ALT just alters the key in the priority queue.

5 Experiments

In this section, we present our experimental evaluation. Our implementation is written in C++ using solely the STL. As priority queue we use a binary heap. Our tests were executed on one core of an AMD Opteron 2218 running SUSE Linux 10.3. The machine is clocked at 2.6 GHz, has 16 GB of RAM and 2 x 1 MB of L2 cache. The program was compiled with GCC 4.2, using optimization level 4.

Inputs. We use two inputs for our evaluation. The *railway* network of Central Europe and a local *bus* network of greater Berlin. Both networks have been provided by HAFAS for scientific use; the former network consists of 30,517 stations and 1,775,552 elementary connections. The corresponding figures for the latter are 2,874 and 744,005, respectively. While the network of Europe provides a good average structure for a railway network mixed of long-distance trains supported by short-distance trains, the bus network of Berlin consists of a very homogeneous structure, since there are almost no "long-distance" buses. Because of this and the very dense operations of buses with their short travel times between stations, it has already been shown [16] that this network seems to be a very hard instance for timetable information queries.

It should be noted that, while our timetable data is realistic, the transfer times at the stations were not available to us. Hence, we generated them at random and chose between 5 and 10 minutes for the railway and between 3 and 5 minutes for the bus network.

Default Settings. In the following, we report preprocessing times and the overhead of the preprocessed data in terms of *additional* bytes per node. We evaluate query performance by running 1 000 random *s–t* queries with source and target station picked uniformly at random. We *fix* the departure time to 7:00 am. We report the average number of settled nodes during the query as well as the average query time. The speed-up refers to the query time and is computed in reference to the classic time expanded model without any speed-up technique applied.

5.1 Models

Parameters. We start our experimental evaluation with parameter tests for our Route-Model. Recall that in the Route-Model we may affect the conversion process by the selection of γ which controls the maximum number of neighbors a station may have in order to become a Route-Model station. In the following we use values between 2 and 10 for γ. Table 1 reports for both our inputs: the resulting size (in terms of number of edges) and query performance. Note that we do not report number of nodes, as the remodeling routine does not add or remove any nodes. We also enabled Node-Blocking (see Section 4.1).

We observe that for both instances the Route-Model yields a speed-up. Increasing γ up to 5 increases performance, while values > 5 do not pay off. This is mostly due to the fact that for both graphs the majority of stations has less or equal than 5 neighbors (91% for the Europe and even 99% for the Berlin network).

Concerning Europe with $\gamma < 5$, we observe that the resulting graph has *less* edges than originally. Recall in the original graph the number of outgoing edges per arrival node is at most 2 (one toward the nearest transfer node and one toward the departure node of the same train). Hence, a decrease in number of the edges can only result from merely one edge being inserted for many arrival nodes at stations of degree 2. Interestingly, this observation of decreasing edges does not hold for our bus network which is due to the high density of the network: Because the stations are very close to each other, it often holds that the travel time to go forth and back between some stations S_1 and S_2 is less than $\text{TRANSFER}(S_1)$, which results in back-edges being inserted for

Table 1: The effect of γ on the performance of the Route-Model with Node-Blocking enabled.

	europe				*bvb*			
	SIZE	QUERY			SIZE	QUERY		
γ-value	#edges	#settled	[ms]	speed-up	#edges	#settled	[ms]	speed-up
reference	8,505,951	1,161,696	534.7	1.00	3,694,253	151,379	37.6	1.00
2	7,912,584	411,836	202.4	2.64	3,785,680	91,591	27.4	1.37
3	8,035,324	359,294	171.7	3.11	4,292,849	74,963	25.2	1.49
4	8,332,816	329,413	158.3	3.38	5,059,228	63,438	25.1	1.50
5	8,729,619	313,046	154.1	3.47	5,437,647	59,670	25.4	1.48
6	9,071,974	303,460	153.9	3.47	5,625,277	57,990	25.6	1.47
7	9,396,276	297,831	155.1	3.45	5,768,926	56,994	25.8	1.46
8	9,712,940	292,482	156.4	3.42	5,782,375	56,921	25.7	1.46
9	9,936,119	289,036	158.7	3.37	5,782,375	56,921	25.8	1.46
10	10,195,050	285,103	159.3	3.36	5,782,375	56,921	25.8	1.46

arrival nodes at S_2 (coming from S_1). Second, the operation frequency of the buses is very high, such that it may occur that edges toward more than the first bus of the route are inserted, when they arrive at the next station within the margin of its transfer time.

Summarizing, a value of $\gamma = 5$ yields the best results for railway input. The corresponding figure for the bus networks is 4.

Comparison to the Classic Time-Expanded Model. Next, we compare different contraction steps (Section 3) and our route model with the classic time expanded model. Table 2 shows the differences in graph size and query performance. While the overall graph size decreases when switching from the classic expanded to the phase 1 model, the number of edges significantely increases if applying our phase 2 model. Although the number of nodes decreases about 50%, this increase in number of edges leads to an *worse* query performance, since more edges are relaxed during the query. We hence conclude that the phase 2 model—and therefore the phase 3 model as well—is not the preferred choice for fast timetable queries.

Regarding the Route-Model, the increase in graph size is still reasonable while the query time decreases. However, we see, that the query performance benefits from Node-Blocking as the speed-up more than doubles in the Europe network with Node-Blocking enabled. The reason for the weak performance without Node-Blocking is that paths through the graph, that should be pruned by the Route-Model approach, are still relaxed when they are not blocked in non-converted big traffic hubs. In the bus network the general performance gain is not as big as with the railway network. Even Node-Blocking does not have such a great impact, which is mostly due to the dense structure of this network.

Because the Route-Model can be combined well with the phase 1 model (departure nodes are simply removed after the conversion to the Route-Model), this gives us a gain in graph size while still keeping the advantages of the Route-Model. The query performance behaves as expected and increases by approximately one third compared to the Route-Model alone. If we then additionally apply Node-Blocking on the route

Table 2: Comparison of the different models. The Route-Model is computed with $\gamma = 5$ for *europe* and $\gamma = 4$ for *bvb*.

		SIZE		QUERY		
input	Model	#nodes	#edges	#settled	[ms]	spd-up
	Classic expanded	5,207,980	8,505,951	1,161,696	534.7	1.00
	Phase 1	3,472,022	6,769,991	768,181	426.5	1.25
	Phase 2	1,736,064	15,571,190	431,274	631.1	0.85
europe	Route	5,207,980	8,729,619	793,462	360.6	1.48
	Route w/ blocking	5,207,980	8,729,619	313,046	154.1	3.47
	Route + Phase 1	3,472,018	6,821,337	439,024	256.3	2.09
	Route + Phase 1 w/ blocking	3,472,018	6,821,337	200,213	122.8	4.35
	Classic expanded	2,232,016	3,694,253	151,379	37.6	1.00
	Phase 1	1,488,011	2,950,248	99,253	29.1	1.29
	Phase 2	744,006	13,229,482	60,218	56.8	0.66
bvb	Route	2,232,016	5,059,228	97,978	32.6	1.15
	Route w/ blocking	2,232,016	5,059,228	63,438	25.1	1.50
	Route + Phase 1	1,488,011	3,918,788	51,210	22.7	1.66
	Route + Phase 1 w/ blocking	1,488,011	3,918,788	34,032	18.6	2.02

+ phase1 model, we get the best query performance of all the models which yields a speed-up of 4.35 in the railway network of Europe and 2.02 in the Berlin bus network.

5.2 Speedup Techniques

Up to now, we showed that by remodeling stations and using additional pruning techniques, we already achieve a speed-up of 4.35 over plain DIJKSTRA. Here, we now show that this approach harmonizes well with other speed-up techniques deriving from road networks.

Path-Selection during Arc-Flags Preprocessing. We already mentioned in Section 4.2 that in expanded timetable networks the number of shortest paths between two nodes is enormously high. It turns out that setting arc-flags for all paths yields a bad query performance. Hence, we have to favor some paths over the others. We proposed four different reasonable strategies: Minimize hops, minimize transfers, minimize accumulated geographic distance along the path and finally minimize the direct geographic distance from the preceding node to the source of the shortest path tree (see Section 4.2). Table 3 shows the impact of each strategy on the performance of Arc-Flags. Note that due to the long preprocessing times of Arc-Flags, we use a subnetwork of our European instance, namely the German railway network called *de_fern* (6822 stations and 554996 connections).

While minimizing hops is useful in road networks [1] (which can be interpreted there as preferring a route that has less road crossings) this results in a poor performance in railway network. Almost all flags are opened during preprocessing, thus the overhead of the Arc-Flags query algorithm outweighs the benefit from the few remaining pruned arcs. Interestingly, using minimal transfer or minimal distance strategies as

Table 3: Arc-Flags. Evaluation of different path-sepection strategies. For each strategy we apply a partition with 64 cells.

	PREPRO		QUERY		
Strategy	[h:m]	[B/n]	#settled	[ms]	speed-up
reference	—	0	152,998	58.1	1.00
hops	17:00	26.2	149,931	70.3	0.83
transfers	16:26	26.2	152,307	71.7	0.81
distance	20:53	26.2	134,462	61.8	0.94
geo. dist. to target	16:08	26.2	38,511	15.0	3.87

path selection yields a poor query performance as well. This is mostly due to too many different paths of boundary nodes of the same cell being optimal, thus too many flags are set to $true$. Recall that the partition is computed on the condensed graph, hence for one station that is at the border of a cell, nodes belonging to all times of day are boundary nodes which may lead to very different transfer or distance minimal routes in the graph.

The minimal direct geographic distance strategy overcomes this issue by *always* choosing the same preceding node for *all* times of the day. For that reason, as many arc-flags as possible are kept $false$, which eventually yields a speed-up of 3.87 on the German railway network. Since all other strategies actually worsen the query performance, we choose the direct geographic distance strategy for further experiments involving Arc-Flags on time expanded railway networks.

Speed-Up Techniques on our Models. In the next experiment we compare the performance of the adapted speed-up techniques on the different models from Section 3. Because of the bad performance of the phase 2 model, we only compare the classic expanded model, the phase 1 model, the Route-Model and the combination of the route and phase 1 models.

Furthermore, we tested the effect of dynamic-landmark-selection against a precomputed set of landmarks. Table 4 shows our results. We show the query performance as well as preprocessing-costs by preprocessing time and additionally bytes per node required to store the preprocessed data. For each model we tested the following speed-up techniques:

- **BA**: Node-Blocking with ALT.
- **BdA**: Node-Blocking with ALT and dynamic-landmark-selection.
- **uFA**: Unidirectional Arc-Flags with ALT.
- **uFdA**: Unidirectional Arc-Flags with ALT and dynamic-landmark-selection.

Regarding classic ALT we always used a set of 8 precomputed landmarks by the *max-Cover* [9] method. Arc-Flags were computed using a partition of 128 cells obtained from *SCOTCH* [17]. The strategy for path-selection was *geographic distance to target*. Note that for Arc-Flags, we turn off Node-Blocking (cf. Section 4.2).

We observe, that for all speed-up technique our modifications to the classic expanded model yield improvements regarding both query performance and preprocessing time. While the transition from the classic to the phase 1 model is more beneficial for

Arc-Flags than ALT with Node-Blocking, the latter performs better on the Route-Model where Node-Blocking fits the model considerably better. The combination "Route + Phase 1" unifies the advantages of each model yielding the best speed-ups.

In general, Arc-Flags has a higher impact on query time than ALT together with Node-Blocking (about 5.5 times faster on both networks) which is being paid for with very high preprocessing time and roughly 30 times more required space per node. Note, that the dynamic ALT comes for free, as it does not require any preprocessing at all. With our modified models we can, however, still achieve a speed-up of 10.13 in Europe and 2.54 in Berlin with dynamic ALT and Node-Blocking, which is useful in a scenario where preprocessing is limited or not allowed.

Table 4: Comparing different models in conjunction with the classic speed-up techniques. The parameter set used throughout: 128 cells, *geographic distance to target* path-selection-strategy for Arc-Flags and 8 landmarks using *maxCover* for the classic ALT.

	europe					*bvb*				
	PREPRO		QUERY			PREPRO		QUERY		
Model/Algorithm	[h:m]	[B/n]	#settled	[ms]	spd	[h:m]	[B/n]	#settled	[ms]	spd
Reference	—	0	1,161,696	534.7	1.00	—	0	151,379	37.6	1.00
Classic Exp. (BA)	$\approx$ 4 s	4.0	261,151	162.7	3.29	$\approx$ 2 s	4.1	96,533	33.6	1.12
Classic Exp. (BdA)	$\approx$ 1 s	4.0	233,280	130.8	4.09	$\approx$ 1 s	4.0	94,345	29.1	1.29
Classic Exp. (uFA)	106:11	106.5	71,937	32.7	16.35	45:30	108.0	49,921	17.0	2.21
Classic Exp. (uFdA)	106:11	106.5	65,143	33.9	15.77	45:30	107.9	49,014	15.2	2.47
Phase 1 (BA)	$\approx$ 5 s	4.5	208,579	145.5	3.67	$\approx$ 2 s	4.1	67,019	26.1	1.44
Phase 1 (BdA)	$\approx$ 1 s	4.0	185,996	116.4	4.59	$\approx$ 1 s	4.0	65,488	22.8	1.65
Phase 1 (uFA)	77:52	127.2	30,583	14.0	38.19	31:59	129.0	15,004	5.4	6.96
Phase 1 (uFdA)	77:52	126.7	27,310	18.5	29.06	31:59	128.9	14,713	5.1	7.37
Route (BA)	$<$ 4 s	4.4	140,826	73.2	7.30	$\approx$ 2 s	4.1	49,591	22.3	1.69
Route (BdA)	$\approx$ 1 s	4.0	127,444	65.4	8.18	$\approx$ 1 s	4.0	48,390	19.8	1.90
Route (uFA)	85:49	109.7	50,050	22.1	24.19	50:58	147.1	25,289	10.2	3.69
Route (uFdA)	85:49	109.3	45,180	25.3	21.13	50:58	147.0	24,785	9.3	4.04
Route + Ph. 1 (BA)	$\approx$ 4 s	4.5	89,524	58.7	9.11	$<$ 2 s	4.1	26,653	16.0	2.35
Route + Ph. 1 (BdA)	$\approx$ 1 s	4.0	80,665	52.8	10.13	$\approx$ 1 s	4.0	26,007	14.8	2.54
Route + Ph. 1 (uFA)	83:58	128.2	20,044	9.5	56.28	34:56	170.6	6,195	2.6	14.46
Route + Ph. 1 (uFdA)	83:58	127.7	17,805	15.2	35.18	34:56	170.5	6,053	2.8	13.43

Comparing the standard ALT against ALT with dynamic landmarks, we observe, that regarding query time dynamic ALT only pays off as long as the general speed-up (achieved through some other speed-up technique or model) does not exceed the cost we pay for computing the distance table on-the-fly. Since the condensed graph of Europe has about 11 times more stations than the Berlin graph, the cost for computing the dynamic distance table carries much more weight there—A one-to-all DIJKSTRA takes about 7 ms on the condensed graph of Europe. Hence, it never pays off using dynamic landmarks together with Arc-Flags here. The same effect can be observed in the Berlin network, however, only with the combination of the route and phase 1 models due to the much smaller condensed graph.

Summarizing, our modifications yield a speed-up of 3.5 if we apply ALT and Arc-Flags to both time-expanded graphs. The corresponding figure for our bus network is 5.5. This yields an overall speed-up of 56.28 for Europe and 14.46 for Berlin when compared to the classic model without any speed-up technique applied.

5.3 Comparison to the Time-Dependent Model

Table 5 compares the performance of DIJKSTRA's algorithm and ALT applied to our route+phase 1 time-expanded model and the time-dependent model. We observe that by the introduction of our Route-Model (and Node-Blocking) query performance of time-expanded queries are faster than for the time-dependent approach. Hence, we are able to close the performance-gap between both models. Analyzing the time-dependent approach, we notice that Node-Blocking is included implicitly: During a query we do not relax an edge more than once although it represents several connections running from one station to another. Hence, early connections *block* later ones. Our remodeling and Node-Blocking technique introduces these optimizations to the time-expanded approach. As a result the performance advantage of the time-dependent approach fades.

Table 5: Performance of DIJKSTRAand uni-directional ALT using a *time-dependent* variant of our European input. For comparison, the corresponding figure for the time-expanded approach (route-model with phase 1) are given as well.

	time-dependent					time-expanded				
	PREPRO	QUERIES				PREPRO	QUERIES			
	time	#settled	speed	time	speed	time	#settled	speed	time	speed
technique	[h:m]	nodes	up	[ms]	up	[h:m]	nodes	up	[ms]	up
Dijkstra	0:00	260 095	1.0	125.2	1.0	0:00	200 213	1.0	122.8	1.0
uni-ALT	0:02	127 103	2.0	75.3	1.7	0:01	89 524	2.2	58.7	2.1

6 Conclusion

In this work, we introduced a local remodeling routine for the time-expanded approach based on the intuition that at many stations in a network, the number of reasonable choices is little. It turns out that this approach leads to a closely related speed-up technique harmonizing well with our remodeling. Moreover, we adapted speed-up techniques to the time-expanded model and show that they harmonize well with our new approach. Altogether, our approach yields query times up to 56.28 times faster than pure DIJKSTRA.

Regarding future work, we are optimistic that our approach would also work well for multi-criteria optimization. Although our pruning techniques may not work as strict as for single-criteria search, the number of reasonable choices is little in this scenario as well. Another very important problem is dynamization. It seems as if updating a time-expanded graph is rather expensive, though possible [3, 15].

References

1. R. Bauer and D. Delling. SHARC: Fast and Robust Unidirectional Routing. In I. Munro and D. Wagner, editors, *Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX'08)*, pages 13–26. SIAM, 2008.
2. R. Bauer, D. Delling, and D. Wagner. Experimental Study on Speed-Up Techniques for Timetable Information Systems. In C. Liebchen, R. K. Ahuja, and J. A. Mesa, editors, *Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07)*, pages 209–225. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
3. D. Delling, K. Giannakopoulou, D. Wagner, and C. Zaroliagis. Timetable Information Updating in Case of Delays: Modeling Issues. Technical Report 133, Arrival Technical Report, 2008.
4. D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering Route Planning Algorithms. Submitted for publication, 2008.
5. E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.
6. R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In C. C. McGeoch, editor, *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, June 2008.
7. A. V. Goldberg and C. Harrelson. Computing the Shortest Path: A* Search Meets Graph Theory. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 156–165, 2005.
8. A. V. Goldberg, H. Kaplan, and R. F. Werneck. Better Landmarks Within Reach. In C. Demetrescu, editor, *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*, pages 38–51. Springer, June 2007.
9. A. V. Goldberg and R. F. Werneck. Computing Point-to-Point Shortest Paths from External Memory. In *Proceedings of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX'05)*, pages 26–40. SIAM, 2005.
10. P. E. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.
11. M. Hilger. Accelerating Point-to-Point Shortest Path Computations in Large Scale Networks. Master's thesis, Technische Universität Berlin, 2007.
12. E. Köhler, R. H. Möhring, and H. Schilling. Acceleration of Shortest Path and Constrained Shortest Path Computation. In *Proceedings of the 4th Workshop on Experimental Algorithms (WEA'05)*, Lecture Notes in Computer Science, pages 126–138. Springer, 2005.
13. U. Lauther. An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background. In *Geoinformation und Mobilität - von der Forschung zur praktischen Anwendung*, volume 22, pages 219–230. IfGI prints, 2004.
14. M. Müller-Hannemann and M. Schnee. Finding All Attractive Train Connections by Multi-Criteria Pareto Search. In *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2007.
15. M. Müller-Hannemann, M. Schnee, and L. Frede. Efficient On-Trip Timetable Information in the Presence of Delays. In *Proceedings of the 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, September 2008.

16. T. Pajor. Goal Directed Speed-Up Techniques for Shortest Path Queries in Timetable Networks, January 2008. Student Research Project.
17. F. Pellegrini. SCOTCH: Static Mapping, Graph, Mesh and Hypergraph Partitioning, and Parallel and Sequential Sparse Matrix Ordering Package, 2007.
18. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Experimental Comparison of Shortest Path Approaches for Timetable Information. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)*, pages 88–99. SIAM, 2004.
19. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Efficient Models for Timetable Information in Public Transportation Systems. *ACM Journal of Experimental Algorithmics*, 12:Article 2.4, 2007.
20. P. Sanders and D. Schultes. Engineering Highway Hierarchies. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA'06)*, volume 4168 of *Lecture Notes in Computer Science*, pages 804–816. Springer, 2006.
21. F. Schulz, D. Wagner, and K. Weihe. Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. In *Proceedings of the 3rd International Workshop on Algorithm Engineering (WAE'99)*, volume 1668 of *Lecture Notes in Computer Science*, pages 110–123. Springer, 1999.

Integrated Gate and Bus Assignment at Amsterdam Airport Schiphol

G. Diepen*, J.M. van den Akker* J.A. Hoogeveen*

Department for Information and Computing Sciences
Utrecht University
P.O. Box 80089, 3508 TB Utrecht, The Netherlands
{diepen,marjan,slam}@cs.uu.nl

Extended abstract

Abstract. At an airport a series of assignment problems need to be solved before aircraft can arrive and depart and passengers can embark and disembark. A lot of different parties are involved with this, each of which having to plan their own schedule. Two of the assignment problems that the 'Regie' at Amsterdam Airport Schiphol (AAS) is responsible for, are the gate assignment problem (i.e. where to place which aircraft) and the bus assignment problem (i.e. which bus will transport which passengers to or from the aircraft). Currently these two problems are solved in a sequential fashion, the output of the gate assignment problem is used as input for the bus assignment problem. We look at integrating these two sequential problems into one larger problem that considers both problems at the same time. This creates the possibility of using information regarding the bus assignment problem while solving the gate assignment problem. We developed a column generation algorithm for this problem and have implemented a prototype. To make the algorithm efficient we used a special technique called stabilized column generation and also column deletion. Computational experiments with real-life data from AAS indicate that our algorithm is able to compute a planning for one day at Schiphol in a reasonable time.
Keywords: gate assigment, integrated planning, airports, column generation, integer linear programming

1 Introduction

Between the time an aircraft lands at an airport and the time it departs again many things must happen. One of the most obvious things is that passengers needs to disembark the aircraft. Moreover, the aircraft needs to be refueled, new passengers need to board, new supplies have to be put on board, the aircraft has to get cleaned. All of the actions take place while the aircraft is standing at a *gate*. We will consider the arrival of an aircraft until the following departure of

* Supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society)

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1591

the same aircraft as one *flight*. The *gate assignment problem* deals with assigning a given set of flights to a set of gates such that certain criteria are met.

In this paper, we consider the gate assignment at Amsterdam Airport Schiphol (AAS). We investigate the daily planning, i.e. the creation of a planning for the upcomimg day on the basis of the available information about the flights of that day. In Diepen et al. [4], we have presented a column generation algorithm to create an assignment for aircraft to gates that is as robust as possible, meaning that any small deviation from the scheduled arrival and departure times should not result in lots of rescheduling.

Some flights are not assigned to a gate with an airbridge but to a so-called remote stand. This implies that passengers have to be transported to and from the aircraft by buses. We have shown how we can create a robust schedule for these platform buses by a similar type of column generation algorithm (see Diepen [3]) in case the gate assignment is given.

This approach resembles the way AAS is actually solving these two problems currently. First the gate assignment problem is solved, the solution of which is then used as input for the bus planning problem. Although the bus planner have the possibility to influence the gate planning by providing preferences, in general the two problems are solved in a sequential way.

Observe that this could imply that a schedule for the gate assignment results in a very bad schedule for the bus planning. In many cases minor changes to the original solution for the gate assignment problem would allow better assignments for the buses. So although this would mean a sub-optimal solution for the gate assignment problem to be used, the solution for both the gate and bus planning as a whole would improve.

In this paper, we focus on *the integration of gate assignment and bus planning*. Our goal is to achieve better overall robustness and a more efficient bus planning without too much negative effects on the gate assignment. The airport authorities at AAS have indicated that robustness is very important for them, in order to limit the amount of gate changes during the day of operations.

During the last years, a signficant amount of research has been performed on the integration of real-life scheduling problems. For example Freling, Huisman, and Wagelmans [6] look into the integration of solving the combination of the vehicle and crew scheduling problems that arise in the public transport scheduling. They present two different models and algorithms for solving the integrated version of the two problems, and compare the results to the results obtained by using the standard sequential approach.

One of the areas where the integration of real-life scheduling problems is investigated a lot, is in the *airline* industry. Cordeau et al. [2] investigate the integration of the aircraft routing problem with the crew scheduling problem. They propose a solution approach based on Benders decomposition and show that solving these two problems as one integrated problem yields significant cost savings. Other integrations that have been considered are schedule assignment and the fleet assignment problems (see Rexing et al. [9] and Lohatepanont and

Barnhart [8]) and the integration of the fleet assignment and the crew scheduling problems (see Gao [7], Clarke et al. [1], and Sandhu and Klabjan [10]).

At Amsterdam Airport Schiphol, the software package currently in use for solving the gate assignment problem, uses a rule based approach for optimizing the assignment. It includes many aspect, however, it does not support the main thing we aim for, robustness. Furthermore, it is also capable of scheduling additional processes besides the assignment of aircraft to gates. For instance, in Vancouver the same program is used and there the scheduling of the push back trucks is also handled by the program.

The purpose of the research described in this paper is to enable the use information of regarding the bus planning problem while solving the gate assignment problem. Instead of an iterative method in which the separate problems are solved in turns and are allowed to send constraints or preferences to each other, our approach is to combine the two assignment problems into *one big problem* and solving this one big problem as a whole, where the objective is to maximize overal robustness.

The outline for the remainder of the paper is as follows: In Section 2 we will give the problem formulation and our model and in Section 3 we present solution method. Furthermore, in Section 4 we will report on the results of the experiments that we performed and finally, in Section 5 we give our conclusions.

2 Problem formulation

In this section, we describe the problem and present an integer linear programming formulation. For the upcoming day we want to create a gate assignment for a given set of flights and a planning for the platform buses transporting passengers to and from flights at a remote stand.

For the gate asssignment several properties of the flights are important:

- Arrival and departure time
- Region of origin and destination (Shengen/EU/Non-EU)
- Size category
- Ground handler

At AAS the ground handlers are divided into two groups: KLM Ground Services and other companies. Clearly, two flights cannot be assigned to the same gate at the same time. At AAS the minimum amount of idle time between two consecutive flights at a gate is 20 minute. For each gate it is known which regions (because of safety regulation), size categories, and grounds handlers it can serve. This results in constraints to ensure that at a gate there are only flights maching the properties of the gate with respect to region, size of the aircraft and ground handler.

Moreover, certain preferences might be taken into account. For example, some airlines such as KLM have their own gates or want their flights to be grouped as much as possibe on certain gates, for example we could require that at least 5 out of 7 Swiss flights are on a specific gate.

Flights that stay on the ground for a longer period, eg. 3 hours, may have to be sprit. This means that after some time the flight is removed from the gate and later is moved back to some (possibly other) gate. We included this in our model, but omit the description for reasons of brevity.

Our objective is to create an assignment schedule that is as robust as possible, meaning that the resulting schedule is able to cope with minor disturbances during the actual day as well as possible. The following picture shows an example of a schedule that is typically non-robust and can be improved by interchanging flights 3 and 4.

Fig. 1. Example of a non-robust schedule.

Observe that a schedule is best able to cope with disturbances if all idle times between each pair of consecutive flights on a gate are as large as possible. We model this with a cost function that greatly penalizes short idle times, while giving very low cost to large, and thus favorable idle times.

For the cost of the idle time t between two consecutive flights v and w on a gate we use the same cost function presented in [4]

$$c^G(t) = conv(v, w)1000(\arctan(0.21(-t)) + \frac{\pi}{2}),$$

where $conv(v, w)$ denotes the convenience multiplier expressing the preference of flight w directly succeeding flight v on a gate. For example, this multiplier is large is v and w belong to the same airline since in this case the airline has a clear incentive to make v depart on time.

If a flight is handled at a remote stand, the passengers are moved to and from the terminal by bus. The number of buses needed depends on the number of passengers. In this way, each flight assigned to a remote stand, results in a number of bus trips. At arrival all trips takes place at the same time, and for the departure there by rule have to be at least two trips and the first trip starts already some time before the departure of the flight. When ordering buses and drivers, AAS can specify the amount buses required per 15 minutes. As a results the bus drivers (about 60) on a day work in about 20 types of shifts, where shifts longer than 4.5 hours contain a mandatory break.

To maximize robustness, we make use of a similar cost function of the idle times t between consecutive trips of the same bus. The exception is that at each given time we have significantly lowered the total cost, this to resemble the fact that the gate assignment is still the more important problems of the two:

$$c^B(t) = 50(\arctan(0.21(-t)) + \frac{\pi}{2}),$$

By taking the sum of the total cost of both sub problems, we now have a representation for the quality of the robustness of a solution as a whole.

The ILP formulation. The model is obtained by combination and extension of the separate models presented in [4] and [3] to solve the gate assignment and the bus planning problems respectively.

Our model is based on so-called *gate plans*, which consist of a set of flights assigned to one gate. We aggregate gates with the same properties into groups of gates and each such group we refer to as a *gate type*. These properties contain at least the origin/destination, size and ground handler. However, a trivial aggregation in which each separate gate (except for the platform stands) is considered a single type is also possible.

We define the decision variable

$$x_i = \begin{cases} 1 \text{ if gate plan } i \text{ is selected} \\ 0 \text{ otherwise,} \end{cases}$$

Since it might be non-trivial to assign all flights to a gate, allow a flight to be unassigned at high cost. This is modelled by the binary variable UAF_v. Let V denote the number of flights, A the number of gate types, S_a the number of gates of type a, and K the number of preferences. Now the robustness cost of the gate assignment are given by:

$$\text{Min} \quad \sum_{i=1}^{N} c_i^G x_i + \sum_{v=1}^{V} Q_v \mathrm{UAF}_v$$

and the gate plan have to satisfy the following constraints:

$$\mathrm{UAF}_v + \sum_{i=1}^{N} g_{vi} x_i = 1 \qquad v = 1 \ldots V \tag{1}$$

$$\sum_{i=1}^{N} e_{ia} x_i \leq S_a \qquad a = 1 \ldots A \tag{2}$$

$$\sum_{i=1}^{N} \sum_{v=1}^{V} \sum_{a=1}^{A} p_{vak} e_{ia} g_{vi} x_i \geq P_k \qquad k = 1, \ldots, K \tag{3}$$

where

$$g_{vi} = \begin{cases} 1 \text{ if flight } v \text{ is in gate plan } i \\ 0 \text{ otherwise,} \end{cases}$$

$$e_{ia} = \begin{cases} 1 \text{ if gate plan } i \text{ is for gate type } a \\ 0 \text{ otherwise,} \end{cases}$$

$$p_{vak} = \begin{cases} 1 \text{ if flight } v \text{ has preference for gate of type } a \text{ in preference } k \\ 0 \text{ otherwise,} \end{cases}$$

Constraint (1) ensures that all flights are either present in one of the selected gate plans, or the unassignment variable for the flight will have the value 1, resulting in a penalty in the objective function.

Constraint (2) ensures that we select as many gate plans of a certain type as there are gates of the type and Constraint (3) ensures that we fullfill all of the preferences that are given with regards to the gate assignment. Here P_k is the minimum required number of flights with a preference for gate type a that we have to assign to a gate of type a to meet the preference constraints, e.g. the constraint can be that at least 7 out of the 10 flights of a certain airline are at a given gate.

In case we only need to solve the bus planning problem, we are given a set of flights of which it is known where exactly they are standing. With this information we can create the trips needed to transport all the passengers and in the model we must ensure that each of these trips is either driven by a bus, or it is left unassigned with a penalty cost.

For the combination of the two problems, we do not yet know which flights will be placed on the platform (and also, on which platform) and therefore we have to find a way to determine which trips we actually need to assign to buses.

To handle this problem, we generate all possible trips for flights that could be assigned to the remote stands. This means that for each of these flights we create the trips for each of the platforms that it can be assigned to. For example, if an arriving flight requires two trips because of the number of passengers and it can be assigned to the D/E platform, as well as the B platform it means that we will create two trips from the D/E platform and two trips from the B platform to the terminal building. Similarly, not only different platforms, but also different destinations in the terminal building must be considered. For each possible combination we would have to create the trips also. To allow for this coupling we will work with all possible trips and determine which of these are needed in a solution and which are not. For this purpose we will use the variables NNT_t for each trip t to denote whether the trip t needs to be assigned to a bus or that the trip is irrelevant for the assignment problem.

Similar to the gate assignment, we define bus plans as the set of trips performed by one bus. We define

$$y_j = \begin{cases} 1 \text{ if bus plan } j \text{ is selected} \\ 0 \text{ otherwise,} \end{cases}$$

and the binary variable UAT_t to signal if trip t is unassigned. Let T be the number of trips, B be the number of shift types and T_b be the number of buses

with drivers available is shift b. We now obtain the following model:

$$\text{Min} \quad \sum_{i=1}^{N} c_i^G x_i + \sum_{v=1}^{V} Q_v \text{UAF}_v + \sum_{j=1}^{M} c_j^B y_j + \sum_{t=1}^{T} R_t \text{UAT}_t$$

subject to

$$(1) - (3)$$

$$\sum_{j=1}^{M} f_{jb} y_j \leq T_b \qquad b = 1 \ldots B \tag{4}$$

$$\text{NNT}_t + \text{UAT}_t + \sum_{j=1}^{M} h_{tj} y_j = 1 \qquad t = 1 \ldots T \tag{5}$$

$$\text{NNT}_t + \sum_{i=1}^{N} \sum_{v=1}^{V} t_{tvi} g_{vi} r_i x_i = 1 \qquad t = 1 \ldots T \tag{6}$$

$$x_i \in \{0, 1\} \qquad i = 1 \ldots N \tag{7}$$

$$y_j \in \{0, 1\} \qquad j = 1 \ldots M \tag{8}$$

where

$$f_{jb} = \begin{cases} 1 \text{ if bus plan } j \text{ for a shift of type } b \\ 0 \text{ otherwise,} \end{cases}$$

$$h_{tj} = \begin{cases} 1 \text{ if trip } t \text{ is in bus plan } j \\ 0 \text{ otherwise,} \end{cases}$$

$$t_{tvi} = \begin{cases} 1 \text{ if assigning flight } v \text{ to gate plan } i \text{ implies trip } t \text{ must be driven} \\ 0 \text{ otherwise,} \end{cases}$$

$$r_i = \begin{cases} 1 \text{ if gate plan } i \text{ is for a remote stand} \\ 0 \text{ otherwise,} \end{cases}$$

Constraint (4) ensures that for each bus shift, we select at most the number of buses present in that shift.

Constraint (5) states that either not needed, or, in case it is needed, must either be assigned to a bus plan or it must be explicitly become unassigned at high cost.

Without any further constraints on the NNT_t variables, the easiest solution would be to set the value of all of these variables to 1 and all of the trip constraints would be satisfied right away. Constraint (6) ensures that this cannot happen for trips that are defined for flights assigned to the remote stands. It is also this constraint that actually links the gate and bus model into one large model.

3 Solving the problem

3.1 Assigning flights to gate plans and trips to bus plans

Observe that the above model determines for each group of gates and each group of shifts an equal sized set of gate plans and bus plans respectively. To

approximate the optimal solution of the above ILP-formulation, we will first relax the integrality constraints (7) and (8). After that we will solve the resulting LP relaxation to optimality by making use of column generation.

The pricing problem. After each iteration of the column generation process, we need to determine whether other columns exist that might improve the value of the objective function, the so-called *pricing problem*. In our case we have to solve two types of pricing problems, one for finding gate plans and one for finding bus plans.

The pricing problem for the gate assignment part boils down to a set of shortest path problems. For each gate type a we define a graph G^a, the nodes of which are the flights that are allowed to be assigned to gate type a and there is an arc between each pair of flights (v, w) such that w can directly succeed v on that gate, i.e., the difference between the arrival time T_w^{arr} of flight w and the departure time T_v^{dep} is at least 20 minutes. Furthermore we add a source vertex s with an arc to every node v and a sink t and an arc from every node to t. Now every path in G_a corresponds to a feasible gate plan and vice versa. To be able to solve the pricing problem as a shortest path problem, we set the cost of arc (v, w) equal to the contribution of flight v to the reduced cost as follows:

$$c^G(T_w^{\text{arr}} - T_v^{\text{dep}}) - \pi_v - \sum_{k=1}^{K} p_{vak}\psi_k - \sum_{t=1}^{T} t_{tv}\rho_t.$$

where the dual multipliers π_v for flight v and ψ_k for preference k follow from Constraint (1) and Constraint (3) respectively. Moreover, ρ_t is the dual multiplier of Constraint (6), which only applies to gate plans that are for remote stands (because only then $r_i = 1$). The last term which incorporates the 'coupling' constraint is the only difference with the pricing problem for the gate assignment problem. We may assume that the flights are sorted by their arrival times, which implies a topological order on the vertices of the graph. Because we now have a DAG with a topological order it is possible to find the shortest path in $O(|V| + |E|)$ time.

The pricing problem with regards to the bus problem boils down to a similar type of shortest path problem and is the same as the pricing problem for solving only the bus planning problem separately. The only difference is that the size of the individual graphs is larger due to the increased number of trips.

Because solving all of the pricing problems in each iteration may be rather time consuming, we have tried out different strategies with regards to which of the pricing problems we solve during each iteration. One possible approach is to interleave the solving of the pricing problems; one iteration we solve the pricing problems for the buses and the other iteration we solve the pricing problems for the gates.

Although, after some initial tests we found that searching for both gate and bus plans with negative reduced cost from the beginning on turned out to work better than the other possibilities.

In [4] and [3], we generated a pool of additional columns that can be added to the ILP and enable us to solve the ILP in a reasonable amount of time. For gate assignment these column are obtained by after the pricing problem has been solved, forbidding one flight in the gate plan and resolve the shortest path problem. We perform this step for every flight in the optimal solution of the pricing problem. For bus planning we generate additional columns in the same way. When solving the problems separately, the columns are added when we start solving the ILP. However, when solving the integrated problem all additional columns with negative reduced cost are already added during the column generation process.

Improvements in solving the LP. During our first experiments, it turned out that the LP problem tends to require a long solution time and be a very degenerate. This degeneracy appears in two ways during the column generation process. First, resolving the restricted master problem after new columns have been added takes quite many iterations and second, new columns that are generated with negative reduced cost do not improve the objective function after they have been added to the restricted master problem.

We have applied two different approaches to improve the solving of the LP. The first approach we used is *column deletion* and consists of the removal of columns with too large positive reduced cost after every given number of iterations. The effect of this removal is not only that the model is simplified and some degeneracy is removed, but also that the resulting model is smaller and therefore it can be solved more quickly. For solving the problems separately, this approach showed promising results for decreasing the computation required time when .

The second approach we implemented is so-called **stabilized column generation**. This technique was introduced in du Merle et al. [5] and consists of a combination of two techniques. One technique is the addition of bounded surplus and slack variables to the original primal problem to overcome degeneracy. The second technique consists of adding surplus and slack variables that have a positive coefficient in the objective function. Combining these two techniques both stabilizes and accelerates the column generation procedure. It decreases the amount of degeneracy a.o. because the slack and surplus variables give more possibilities for assigning a positive value to a newly added column. Moreover, it has a positive effect on the tailing-off effect, i.e. slow convergence. A more elaborate explanation is omitted for reasons of brevity.

Solving the ILP. After the LP is solved to optimality by means of column generation, we are not finished yet because this solution might be fractional. In case it is integral, we are finished since we have an integral solution that is optimal. If we do not have an integral solution, we proceed as follows:

1. first we add all unique extra gate and bus plans that were generated as extra columns while solving the pricing problems.
2. we then add all the unique variables that were taken out during the column generation

3. we reinstate the integrality constraints (7) and (8)

Solving the resulting ILP turned out to be still quite difficult. In order to speed up this solving, we added additional constraints to the problem. These constraints act as a rounding-heuristic. For each flight and for each bus these additional constraints were created in the following way:

1. Determine if a flight or trip that is only present in selected gate plans and bus plans respectively that are all of the same type, meaning that in the fractional solution a flight or a bus trip is always assigned to one particular gate type or one particular bus shift.
2. Create a constraint that ensures the flight or the trip has to be assigned to that particular gate type or bus shift in an integral solution.

Although the above constraints might cause the optimal solution of our initial ILP to be cut off, our experiments did not show any noticeable negative side effects with regards to the cost of the integral solution compared to the optimal fractional solution.

3.2 Assigning gate and bus plans to the actual gates and buses

After solving the ILP from the previous section, we have determined the set of gate and bus plans that provide a (near) optimal solution. For each group of gates and each group of shifts we have an equal size set of gate plans and bus plans respectively. The one thing still left to do is to assign each gate plan and each bus plan to each unique gate and bus respectively.

In case of the bus planning problem, this part is trivial since the buses within one shift do not have any differences at all; it really does not matter to which of these buses a particular bus plan is assigned to.

However, for the gate assignment problem it depends on the definition of the gate types. If each single gate is a separate type, we already have an assignment of flights to physical gates and this step is also trivial.

If we have grouped the gates with certain equal properties into types, the individual gates within such a type still might be different on some other, less important properties. These additional properties can be used for determining to which physical gate a particular gate plan needs to be assigned.

Since the size of these problems is relatively small (in the order of 5 to 10 gates within one group) it is probably most effective to leave this up to the gate planner to do this manually.

4 Experimental results

For testing our model, we wrote a prototype in C++ and ran numerous experiments. All experiments were ran on on Pentium 4 2.8 GHz computer equipped with 1GB of RAM. The solver we used for solving all (I)LP problems is Cplex 9.1.3 via the Concert Technology interface.

AAS provided us with both data regarding the gate assignment problem, which consisted of all flight information for three high-season (HS) days and three low-season (LS) days and data regarding the bus planning problem with all information regarding buses for one complete month.

From the supplied gate data we created two types of instances. In one type of instances we aggregate all gates with identical properties (e.g. size, region, ground handler, pier) into groups of gates. We refer to this type of instances as Grouped Gates (GG). Furthermore, we constructed instances where every gate is considered as a group with size one except for the platform gates. Recall that for these instances our algorithm directly assigns flights to physical gates. We refer to this type of instances as Single Gates (SG). This deaggregation results in over twice the number of gate types, as can be seen in Table 1. This way we created 12 instances with regards to the gate and flight information. The high-season instances contain about 600 flights and about 1000 arival/departure events for the bus planning. For the low-season instances these numbers are 500 and 900 respectively.

To create a sufficiently large number of experiments, we combined each of the 12 gate assignment instances with the buses and shifts of all 30 of the bus planning problem instances. These instances contain about 60 buses and about 20 type of shifts (of which about 70 percent is long enough to contain a mandatory break). We may expect the set of buses available at each given time of the day should roughly be enough for driving all trips.

Instance	Gates	Gate types	Remote stands
Grouped	128	40	34
Single	128	94	34

Table 1. Sizes of the provided instances with regard to gates

In Table 2 we present the general results with regards to solving the LP part of the problem. We combined each instance of the gate assignment problem with the 30 available instances of the bus planning problem and we present the average time over these 30 instances needed for solving each combination, the minimum time, and the maximum time. We also present the average number of iterations needed to solve the LP relaxation and finally, we also present the average time needed in each iteration of the column generation process to solve the pricing problem and the time needed for resolving the Restricted Master Problem (RMP) after we have added the columns found when solving the pricing problem.

Our experiments indicate that the LP can be solved within a reasonable amount of time. From Table 2 we can see that a significant amount of the time needed for solving the LP-relaxation is spent in solving all the separate pricing problems. Since all parts of the pricing problem that need to be solved can be solved completely independent from each other, we could easily bring down

	Total time LP (s)				Avg time (s)/iter	
Instance	Average	Min	Max	Avg iter	RMP	Pricing
02-08-GG	1129.6	967.8	1472.0	161.67	2.8	3.9
02-08-SG	2070.1	1752.1	2657.7	171.90	4.8	6.8
03-08-GG	973.9	864.7	1213.2	148.27	2.6	3.7
03-08-SG	1847.4	1627.4	2337.8	163.07	4.4	6.5
04-08-GG	1142.6	1010.4	1641.3	157.50	3.2	4.0
04-08-SG	2575.2	2189.9	3970.3	212.77	4.6	7.2
15-03-GG	658.5	560.3	769.3	165.17	1.1	2.7
15-03-SG	1235.8	1094.6	1472.0	175.17	1.9	4.8
16-03-GG	710.0	623.8	850.4	161.90	1.3	2.8
16-03-SG	1383.4	1144.0	1661.5	175.87	2.5	5.0
17-03-GG	595.0	474.6	775.1	141.37	1.2	2.8
17-03-SG	1125.1	991.3	1422.4	151.70	2.2	4.9

Table 2. General LP results

the influence of the pricing problems on the total time needed for solving the LP-relaxation by making use of parallel programming.

To investigate the effect of the column deletion and the stabilized column generation, we also ran part of the instances without these enhancements. It could be clearly seen that the time needed to solve the LP relaxation to optimality explodes without the use of column deletion and stabilization. One part responsible for this huge increase in time needed is the large increase in the average time needed for solving one iteration of the RMP. This can be explained by the fact that after a couple of iterations, the model quickly becomes very large due to the fact that all columns stay in the model.

It turns out that without column deletion and stabilized column generation, the average number of iterations needed to solve the LP-relaxation to optimality is higher than when both are enabled, while the average time needed for solving the pricing problems is lower. The increase in number of iterations needed is an example of the so-called tailing-off effect. In the beginning there are big improvements in each iteration, while more and more iterations are needed when closer by the optimum. Using the stabilized column generation has a positive effect on this tailing-off effect, as can be seen by the number of iterations needed.

It turns out that the combination of column deletion and stabilized column generation are responsible for a huge improvement, in our experiments by a factor 2.5 up 19, in the time needed for solving the LP-relaxation to optimality with column generation. Interesting is the fact that the improvement seems larger when the instances are larger (see HS versus LS)

The results for solving the ILP are given in Table 3. The table shows that we were able to solve the very large ILP within a few minutes. In our experiments the integrality gap turned out to be very small.

As mentioned in Section 3 we added additional constraints to the model before solving the actual ILP. These additional constraints can be considered as a kind of rounding-heuristic in the way that if in the optimal solution for the

LP-relaxation a flight is always assigned to a certain type of gate in all selected gate plans, we add a constraint that enforces the flight to be assigned to a gate plan of that type.

The average number of constraints that were added for flights as well as for buses is shown in Table 3. These constraints result in ILP models that are a lot smaller and hence in a much smaller solution time. From our experiments we found that the additional constraints did not have a significant impact on the value of the final ILP solution and did not result in infeasibility of the ILP.

	Average additional constraints		Average solving
Instance	Flight constraints	Trip constraints	time ILP (s)
02-08-GG	121.4	57.6	43.5
02-08-SG	103.4	57.9	54.1
03-08-GG	117.8	57.1	42.0
03-08-SG	105.4	57.7	103.3
04-08-GG	119.3	57.2	82.7
04-08-SG	108.7	57.5	95.2
15-03-GG	108.9	58.4	86.5
15-03-SG	91.0	59.0	271.0
16-03-GG	107.0	59.1	45.8
16-03-SG	84.2	59.3	170.6
17-03-GG	118.5	59.9	20.6
17-03-SG	105.6	59.6	29.5

Table 3. General results ILP

One other way to speed up the process of solving the ILP we used is to first only solve the root node relaxation. We then add a so called cut up limit to the model that is 0.5% above the value of the root node. This cut up limit acts for the ILP solver as if an integral solution with that particular value has already been found, meaning that any node with a relaxation value greater than this cut up value is automatically pruned. Strictly speaking this might result in infeasibility of the ILP (when the optimal ILP solution exceeds the thhreshold), but this never occured in our experiments.

Furthermore, when looking at the time needed for solving the various final ILP models, we can see that these times are still within very acceptable ranges, also for the Single Gate Problems. This indicates that it is feasible to assign flights and trip directly to physical gates and buses respectively.

5 Conclusion and further research

We have investigated the combination of two assignment problems that in practice are solved in a sequential fashion. We formulated the combined problem in one large model for which we approximate the optimal solution by means of an approach based on column generation.

We implemented our algorithm and tested it with real-life data provided by AAS. The results show that our approach is capable of solving these real-life instances within acceptable time, especially given the fact that this approach solves two problems within about the same time that currently is available at AAS for the computer to present a solution for only the gate assignment problem.

We also showed that our approach is still capable of solving the instances within acceptable running times if we create a single gate type for each separate gate, except for the remote stands. This different model lead to over twice the number of gate types which significantly increased the size of the instances.

We are currently performing a simulation study of the platform buses, to evaluate the robustness of the column generation planning compared to a kind of first-come-first-served method as used at AAS. We can clearly see that the column generation schedule is more smooth, in the sense that the idle time is spread more evenly. Currently, the gate assignment at AAS needs a lot of replanning during the day of operation. However, comparing the quality of our resulting schedules to the actual schedules in use at AAS is difficult for a variety of reasons, the main one being the fact it is not possible to retrieve the schedule we would like to compare to, namely the initial schedule as produced by the computer for the upcoming day.

An interesting possibility of further investigation is to start looking at a more operational type of planning. It would be interesting to see how our suggested approach performs if we do not let it create a schedule from scratch but we supply it with a schedule and some disturbances and let the program try to resolve this updated problem.

One of the main things that would have to be considered for this approach is the fact that any new solution should not deviate too much from the currently existing solution. So when solving the problems after some parts are fixed (since they already happened) and other events have changed properties (e.g. earlier or later Estimated Time of Arrivals and Departures) the cost function would not only have to consider the robustness of the schedule, but also the similarity to the original-day-ahead schedule, since too many changes in a schedule will result in a lot of confusion for the different parties dependent on the schedule.

References

1. L. Clarke, C. Hane, E. Johnson, and G. Nemhauser. Maintenance and crew considerations in fleet assignment. *Transportation Science*, 30:249–260, 1996.
2. J.-F. Cordeau, G. Stojkovic, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.
3. G. Diepen. *Column Generation Algorithms for Machine Scheduling and Integrated Airport Planning*. PhD thesis, Utrecht University, (In preparation), 2008.
4. G. Diepen, J. M. v. d. Akker, J. A. Hoogeveen, and J. W. Smeltink. Using column generation for gate planning at amsterdam airport schiphol. Technical Report UU-CS-2007-018, Institute of Information and Computing Sciences, Utrecht, the Netherlands, 2007.

5. O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Math.*, 194(1-3):229–237, 1999.
6. R. Freling, D. Huisman, and A. P. M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85, 2003.
7. C. Gao. *Airline Integrated Planning and Operations*. PhD thesis, Georgia Institute of Technology, August 2007.
8. M. Lohatepanont and C. Barnhart. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science*, 38(1):19–32, 2004.
9. B. Rexing, C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy. Airline fleet assignment with time windows. *Transportation Science*, 34(1):1–20, 2000.
10. R. Sandhu and D. Klabjan. Integrated airline planning. In *AGIFORS Symposium 2004, Singapore*, 2004.

IP-based Techniques for Delay Management with Priority Decisions *

Michael Schachtebeck and Anita Schöbel

Institute for Numerical and Applied Mathematics,
Georg-August University, Göttingen, Germany.
{schachte,schoebel}@math.uni-goettingen.de

Abstract. *Delay management* is an important issue in the daily operations of any railway company. The task is to update the planned timetable to a *disposition timetable* in such a way that the inconvenience for the passengers is as small as possible. The two main decisions that have to be made in this respect are the *wait-depart decisions* to decide which connections should be maintained in case of delays and the *priority decisions* that determine the order in which trains are allowed to pass a specific piece of track. They later are necessary in the capacitated case due to the limited capacity of the track system and are crucial to ensure that the headways between different trains are respected and that single-track traffic is routed correctly. While the wait-depart decisions have been intensively studied in literature (e.g. [Sch06,Gat07]), the priority decisions in the capacitated case have been neglected so far in delay management optimization models.

In the current paper, we add the priority decisions to the integer programming formulation of the delay management problem and are hence able to deal with the capacitated case. Unfortunately, these constraints are disjunctive constraints that make the resulting event-activity network more dense and destroy the property that it does not contain any directed cycle. Nevertheless, we are able to derive reduction techniques for the network which enable us to extend the formulation of the never-meet property from the uncapacitated delay management problem to the capacitated case. We then use our results to derive exact and heuristic solution procedures for solving the delay management problem.

The results of the algorithms are evaluated both from a theoretical and a numerical point of view. The latter has been done within a case study using the railway network in the region of Harz, Germany.

1 Introduction

The delay management problem deals with (small) source delays of a railway system as they occur in the daily operational business of any public transportation company. In case of such delays, the scheduled timetable is not feasible any more and has to be updated to a *disposition timetable*. The main question which has been treated in the literature so far is to decide which trains should wait for delayed feeder trains and which trains better depart on time (*wait-depart decisions*).

A first integer programming formulation for the uncapacitated delay management problem has been given in [Sch01] and has been further developed in [GHL08,Sch07b], see also [Sch06] for an overview about various models. The complexity of the problem has been investigated in [GJPS05,GGJ+04] where it turns out that the problem is NP-hard even in very special cases. The online version of the problem has been studied in [GJPW07,Gat07]. In [BHLS07], it was shown that the online version of the uncapacitated delay management problem is PSPACE-hard. Further publications about delay management include a model in the context of max-plus-algebra ([RdVM98,Gov98]), a formulation as discrete time-cost tradeoff problem ([GS07]) and simulation approaches ([SM99,SMBG01])

* This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

However, these studies neglect the limited capacity of the track system while dealing with delay management. Adding these constraints, the problem becomes significantly harder to solve. Some first ideas on how to model these constraints in the context of delay management have been presented in [Sch07a]. Capacity constraints are also taken into account in a real-world application studied within the project *DisKon* supported by Deutsche Bahn (see [BGJ+05]). Here, the following setting to apply delay management in practice is suggested: In a first step, a macroscopic approach deals with the wait-depart decisions, while a second step ensures feasibility within a microscopic model. This is done by postponing departures until the track to be used is available. It may however yield rather bad solutions.

In the following, we will for the first time analyze the integer programming formulation of the delay management problem for the capacitated case.

The remainder of the paper is structured as follows. In Section 2 we present an integrated integer programming model including the priority decisions and hence respecting the limited capacity of the track system. We analyze the formulation, present reduction techniques and extend the never-meet property in Section 3. We then discuss in which cases the problem can be solved exactly using the software Xpress. Four heuristic approaches are described and analyzed in Section 4. A numerical evaluation of these approaches also is presented in Section 4. We finally conclude the paper mentioning ideas for further research.

2 Integer Programming Formulation

The uncapacitated delay management problem is defined as follows: Given the public transportation network $PTN = (V, E)$ (consisting of the set V of stations and the set E of direct links between stations), the set $\mathcal{F}$ of trains, a set of connections and some source delays, decide which connections should be maintained and which connections should be dropped such that the average delay of a passenger at his final destination is minimal. This problem was first introduced in [Sch01].

In this paper, we take the limited capacity of the tracks into account to obtain the *delay management problem with capacity constraints*. This means we also have to decide which train should drive first if two or more trains use the same piece of infrastructure (for example on single-track lines or when two consecutive trains use the same track in the same direction). Note that it can be better to change the originally scheduled order of the trains to reduce the delay. A first model of this problem has been introduced in [Sch07a]. Here we present and analyze its formulation as integer program.

To this end, we first introduce the corresponding *event-activity network* which is a directed graph $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ (see [Nac98,Sch07b]). $\mathcal{E}$ 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 a time π_i to each event $i \in \mathcal{E}$. If a delay occurs, we need to update the given timetable π to a so called *disposition timetable* $x \in \mathbb{N}^{|\mathcal{E}|}$. To present the constraints that have to be satisfied by a (disposition) timetable, we need the following four different types of activities, $\mathcal{A} = \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{change}} \cup \mathcal{A}_{\text{head}}$:

- driving activities $\mathcal{A}_{\text{drive}} \subset \mathcal{E}_{\text{dep}} \times \mathcal{E}_{\text{arr}}$ modeling the driving of a train between two consecutive stations (including turn-around edges),
- waiting activities $\mathcal{A}_{\text{wait}} \subset \mathcal{E}_{\text{arr}} \times \mathcal{E}_{\text{dep}}$,
- changing activities $\mathcal{A}_{\text{change}} \subset \mathcal{E}_{\text{arr}} \times \mathcal{E}_{\text{dep}}$ which are used to model transfers from one train to another one, and
- headway activities $\mathcal{A}_{\text{head}} \subset \mathcal{E}_{\text{dep}} \times \mathcal{E}_{\text{dep}}$ which model the limited capacity of the track system.

If two events $i, j \in \mathcal{E}$ are connected by an activity $(i, j) \in \mathcal{A}$, then event i has to be performed before event j can take place. In particular, each activity $a = (i, j) \in \mathcal{A}$ has assigned a duration. If a is a driving, waiting, or changing activity, this duration is denoted by $L_a = L_{ij}$, and we require

$$x_j - x_i \geq L_{ij}.$$

The headway activities, on the other hand, appear in pairs: if $(i,j) \in \mathcal{A}_{\text{head}}$, then $(j,i) \in \mathcal{A}_{\text{head}}$, too. This is used to model the disjunctive constraints $x_j - x_i \geq L_{ij}$ or $x_i - x_j \geq L_{ji}$. The goal is to choose exactly one activity of each such pair and to respect the resulting constraint. This means that a priority decision has to be made.

In order to present the integer programming formulation, we need some more parameters:

First, we allow two types of source delays: The first is a delay d_i at an event $i \in \mathcal{E}$ (e.g. a driver coming too late to his duty), which refers to a fixed point of time, such that $x_i \geq \pi_i + d_i$ is required. The second is a delay $d_a = d_{ij}$ which increases the duration of an activity $a = (i,j) \in \mathcal{A}$, 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 duration L_a of activity a. If an activity has no source delay (this is, for example, the case for all headway activities as we do not allow activity delays on headway activities), we assume $d_a = 0$ to simplify the notation.

Moreover, let w_i be the number of passengers getting off at event $i \in \mathcal{E}$ and w_a be the number of passengers who want to use a connection $a \in \mathcal{A}_{\text{change}}$. Throughout this paper, we assume $w_a > 0$ for all $a \in \mathcal{A}_{\text{change}}$ (otherwise, nobody uses the connection, so it can be removed from the network). We further assume that all lines have a common period T (this assumption can easily be relaxed by introducing periods T_a for all changing activities $a \in \mathcal{A}_{\text{change}}$).

To model the wait-depart decisions, we introduce binary variables

$$z_a = \begin{cases} 0 & \text{if changing activity } a \text{ is maintained} \\ 1 & \text{otherwise} \end{cases} \qquad g_{ij} = \begin{cases} 0 & \text{if event } i \text{ takes place before event } j \\ 1 & \text{otherwise} \end{cases}$$

for all changing activities $a \in \mathcal{A}_{\text{change}}$ and for all headway activities $(i,j) \in \mathcal{A}_{\text{head}}$. The integer programming formulation reads as follows:

$$(\mathbf{DM}) \ \min f(x,z,g) = \sum_{i \in \mathcal{E}_{\text{arr}}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} z_a w_a T \tag{1}$$

such that

$$x_i \geq \pi_i + d_i \quad \forall i \in \mathcal{E} \tag{2}$$
$$x_j - x_i \geq L_a + d_a \quad \forall a = (i,j) \in \mathcal{A}_{\text{nice}} := \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}} \tag{3}$$
$$M z_a + x_j - x_i \geq L_a \quad \forall a = (i,j) \in \mathcal{A}_{\text{change}} \tag{4}$$
$$M g_{ij} + x_j - x_i \geq L_{ij} \quad \forall (i,j) \in \mathcal{A}_{\text{head}} \tag{5}$$
$$x_i \in \mathbb{N} \quad \forall i \in \mathcal{E} \tag{6}$$
$$z_a \in \{0,1\} \quad \forall a \in \mathcal{A}_{\text{change}} \tag{7}$$
$$g_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}_{\text{head}} \tag{8}$$
$$g_{ij} + g_{ji} = 1 \quad \forall (i,j) \in \mathcal{A}_{\text{head}} \tag{9}$$

where M is a constant which is "large enough". We will show in Corollary 2 that M can indeed be chosen finitely beforehand. But let us first explain the meaning of the objective function and of the constraints:

In the objective function, we minimize the sum of all delays passengers have when starting their trips or at their final destinations plus the sum of all missed connections. It approximates the sum of all delays over all customers. Furthermore, note that any optimal solution of this program is a Pareto solution with respect to the two objective functions *minimize the delay over all vehicles* and *minimize the number of missed connections.*

Constraints (3) make sure that the delay is passed on correctly along waiting and driving activities. (4) and (5) do the same for changing activities that are maintained and for the headway activities which should be respected. Constraint (9) ensures that exactly one of each pair of headway constraints is respected.

Relaxing all constraints modeling the limited capacity of the tracks yields the uncapacitated delay management problem:

$$(\mathbf{UDM})\ \min f(x,z) = \sum_{i\in\mathcal{E}_{\text{arr}}} w_i(x_i-\pi_i) + \sum_{a\in\mathcal{A}_{\text{change}}} z_a w_a T$$

$$\text{such that (2), (3), (4), (6), (7) are satisfied.}$$

Let us denote the objective value of the optimal solution of (DM) by F^{DM} and let F^{UDM} be the objective value of the corresponding instance of (UDM). Since (UDM) is a relaxation of (DM), we obtain $F^{\text{UDM}} \leq F^{\text{DM}}$.

Later on, we will fix the priority variables heuristically and treat the resulting headway constraints as the constraints in (3). We hence define for some set $\mathcal{A}_{\text{fix}} \subseteq \mathcal{A}_{\text{head}}$ the problem

$$(\mathbf{UDM}(\mathcal{A}_{\text{fix}}))\ \min f(x,z) = \sum_{i\in\mathcal{E}_{\text{arr}}} w_i(x_i-\pi_i) + \sum_{a\in\mathcal{A}_{\text{change}}} z_a w_a T$$

such that

$$x_j - x_i \geq L_a + d_a \quad \forall a=(i,j)\in\mathcal{A}_{\text{fix}} \tag{10}$$

$$\text{and such that (2), (3), (4), (6), (7) are satisfied.}$$

Note that UDM($\mathcal{A}_{\text{fix}}$) yields a *feasible* solution of (DM) if $\mathcal{A}_{\text{fix}} = \{(i,j)\in\mathcal{A}_{\text{head}} : g_{ij}=0\}$ for some $g\in\{0,1\}^{|\mathcal{A}_{\text{head}}|}$ which satisfies (9) (and provided that UDM($\mathcal{A}_{\text{fix}}$) is feasible). In this case we obtain $F^{\text{UDM}} \leq F^{\text{DM}} \leq f^{\text{UDM}(\mathcal{A}_{\text{fix}})}$.

We may also fix the variables z_a and g_{ij} and obtain $\mathcal{A}_{\text{fix}} := \{a\in\mathcal{A}_{\text{change}} : z_a = 0\ \} \cup \{(i,j)\in\mathcal{A}_{\text{head}} : g_{ij}=0\ \}$. Determining the remaining variables x_i in (DM) then reduces to a simple project planning problem:

$$(\mathbf{PP}(\mathcal{A}_{\text{fix}}))\ \min f(x) = \sum_{i\in\mathcal{E}_{\text{arr}}} w_i(x_i-\pi_i)$$

$$\text{such that (2), (3), (10), and (6) are satisfied.}$$

The version of (DM) in which $\mathcal{A}_{\text{change}} = \emptyset$ has been shown to be NP-complete in [CS07]. This yields NP-completeness of (DM). However, (PP($\mathcal{A}_{\text{fix}}$)) can be solved in polynomial time, e.g. by applying the forward phase of the critical path method (CPM) of project planning (see [Elm77]) as follows: We first sort $\mathcal{E} = \{i_1,\dots,i_{|\mathcal{E}|}\}$ topologically and obtain an order $\prec$. Then we set

$$\tilde{x}_{i_1} := \pi_{i_1} + d_{i_1} \tag{11}$$

$$\text{for all } k\in\{i_2,\dots,i_{|\mathcal{E}|}\} : \tilde{x}_k := \max\Big\{\pi_k + d_k, \max_{a=(i,k)\in\mathcal{A}_{\text{fix}}\cup\mathcal{A}_{\text{nice}}} \tilde{x}_i + L_a + d_a\Big\}. \tag{12}$$

We now come back to the integer programming formulation of (DM) and show that M is finite and can be chosen beforehand for any instance of (DM). To this end, we first give an upper bound on the maximum time of each single event in an optimal disposition timetable. We denote the slack time of an activity $a=(i,j)$ as

$$s_a = \pi_j - \pi_i - L_a,$$

i.e. s_a gives the buffer time included in the scheduled duration of the activity.

Theorem 1. *Let an instance of (DM) be given and let*

$$D := \max_{i\in\mathcal{E}} d_i + \sum_{a\in\mathcal{A}_{\text{nice}}} (d_a - s_a)_+ + \sum_{(i,j)\in\mathcal{A}_{\text{head}}:\pi_i>\pi_j} \pi_i - \pi_j + L_{ij}. \tag{13}$$

Then there exists an optimal solution (x,z,g) of (DM) such that $x_k \leq \pi_k + D$ for all $k\in\mathcal{E}$.

Proof. We show the following stronger statement: For any feasible solution $(\bar{x}, z, g)$ of (DM), there exists a feasible solution $(\tilde{x}, z, g)$ with $\tilde{x}_k \leq \bar{x}_k$ (hence $f(\tilde{x}, z, g) \leq f(\bar{x}, z, g)$) that fulfills $\tilde{x}_k \leq \pi_k + D$ for each $k \in \mathcal{E}$.

Given $(\bar{x}, z, g)$, let $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : g_{ij} = 0\}$. As $(\bar{x}, z, g)$ is a feasible solution, $\mathcal{N}' := (\mathcal{E}, \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{fix}})$ does not contain any directed circle, so we sort $\mathcal{E} = \{i_1, \ldots, i_{|\mathcal{E}|}\}$ topologically. We now solve $\text{TT}(\mathcal{A}_{\text{fix}})$ optimally and denote the solution obtained by $\tilde{x}$ (see (11) and (12)). Then $(\tilde{x}, z, g)$ is a feasible timetable satisfying $\tilde{x}_k \leq \bar{x}_k$.

Claim: For each $k \in \mathcal{E}$ we have $\tilde{x}_k \leq \pi_k + U_k$ where

$$U_k = \max_{\substack{i \in \mathcal{E}: \\ i \preceq k}} d_i + \sum_{\substack{a=(i,j) \in \mathcal{A}_{\text{nice}}: \\ j \preceq k}} (d_a - s_a)_+ + \sum_{\substack{(i,j) \in \mathcal{A}_{\text{head}}: \\ g_{ij}=0, \pi_i > \pi_j, j \preceq k}} \pi_i - \pi_j + L_{ij}. \tag{14}$$

We prove the claim by induction. For the first event, we have $\tilde{x}_1 = \pi_1 + d_1 \leq \pi_1 + U_1$. Now consider $\tilde{x}_k$. We distinguish the following three cases depending on which term in the definition (12) of $\tilde{x}$ is maximal:

- $\tilde{x}_k = \pi_k + d_k$. Since $d_k \leq U_k$, the claim is true.
- $\tilde{x}_k = \tilde{x}_i + L_a + d_a$ for $(i,k) \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{fix}}$. We assume that $(i,k) \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}$ or that $(i,k) \in \mathcal{A}_{\text{head}}$ with $\pi_i < \pi_k$. Then

$$\begin{aligned}
\tilde{x}_k &= \tilde{x}_i + L_a + d_a \leq \pi_i + U_i + L_a + d_a \\
&\leq \underbrace{\pi_i + L_a}_{\leq \pi_k} + \max_{\substack{i' \in \mathcal{E}: \\ i' \preceq i}} d_{i'} + d_a + \sum_{\substack{a'=(i',j) \in \mathcal{A}_{\text{nice}}: \\ j \preceq i}} (d_{a'} - s_{a'})_+ + \sum_{\substack{(i',j) \in \mathcal{A}_{\text{head}} \\ g_{i'j}=0, \pi_{i'} > \pi_j, j \preceq i}} \pi_{i'} - \pi_j + L_{i'j} \\
&\leq \pi_k + U_k, \text{ hence the claim is true.}
\end{aligned}$$

- $\tilde{x}_k = \tilde{x}_i + L_a + d_a$ for $(i,k) \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{fix}}$ where $(i,k) \in \mathcal{A}_{\text{head}}$ with $\pi_i > \pi_k$ and $g_{ik} = 0$. Then

$$\begin{aligned}
\tilde{x}_k &= \tilde{x}_i + L_{ik} \leq \pi_i + U_i + L_{ik} \\
&\leq \max_{\substack{i' \in \mathcal{E}: \\ i' \preceq i}} d_{i'} + \sum_{\substack{a'=(i',j) \in \mathcal{A}_{\text{nice}}: \\ j \preceq i}} (d_{a'} - s_{a'})_+ \\
&\quad + \pi_k + (\pi_i - \pi_k + L_{ik}) + \sum_{\substack{(i',j) \in \mathcal{A}_{\text{head}} \\ g_{i'j}=0, \pi_{i'} > \pi_j, j \preceq i}} \pi_{i'} - \pi_j + L_{i'j} d \leq \pi_k + U_k,
\end{aligned}$$

  and again the claim follows.

□

Using this result, we can give an upper bound on the minimal size needed for M:

Corollary 2. *$M \geq D$ is "large enough".*

Proof. We show that each timetable that satisfies (2)-(3), (6)-(9), (4) for all $a \in \mathcal{A}_{\text{change}}$ with $z_a = 0$ and (5) for all $(i,j) \in \mathcal{A}_{\text{head}}$ with $g_{ij} = 0$ also satisfies (4) for all $a \in \mathcal{A}_{\text{change}}$ with $z_a = 1$ and (5) for all $(i,j) \in \mathcal{A}_{\text{head}}$ with $g_{ij} = 1$ if $M \geq D$.

For each changing activity $a = (i,j) \in \mathcal{A}_{\text{change}}$, $\pi_j - \pi_i \geq L_a$, hence (using Theorem 1)

$$M \geq D \geq x_i - \pi_i \geq x_i - \pi_j + L_a \overset{(2)}{\geq} x_i - x_j + L_a,$$

so (4) indeed is satisfied for all $a \in \mathcal{A}_{\text{change}}$.

Now, let $(i,j) \in \mathcal{A}_{\text{head}}$ with $g_{ij} = 0$. We have to show that $M + x_i - x_j \geq L_{ji}$

Case 1: $\pi_i < \pi_j$. We use the proof of Theorem 1: For each j, each term that is added to U_j also is added to D. As $\pi_i < \pi_j$ and $g_{ij} = 0$, the term $\pi_j - \pi_i + L_{ji}$ is added to D, but not to U_j, hence $D - U_j \geq \pi_j - \pi_i + L_{ji}$ and we obtain

$$x_j \leq \pi_j + U_j \leq \pi_j + D - (\pi_j - \pi_i + L_{ji}) = D + \pi_i - L_{ji} \overset{(2)}{\leq} D + x_i - L_{ji}.$$

Case 2: $\pi_i > \pi_j$. As π is a feasible timetable, $\pi_i > \pi_j$ implies $\pi_i - \pi_j \geq L_{ji}$. Using Theorem 1,

$$M \geq D \geq x_j - \pi_j \geq x_j - \pi_i + L_{ji} \overset{(2)}{\geq} x_j - x_i + L_{ji}.$$

Both cases show that (5) is indeed satisfied for all $(i,j) \in \mathcal{A}_{\text{head}}$. □

3 Reducing the Complexity of the Integer Program

Headway constraints make the delay management problem hard to solve; due to headway constraints, a delay from a subsequent train might be carried over even to a train which has been scheduled earlier if the order of both trains is switched. At a first glance, it seems that all of the headway activities can carry over a delay to a previous train in an optimal solution. However, there indeed are some headway activities that can be neglected beforehand as we will show in this section. This reduction helps to solve (DM) more efficiently. We will introduce an algorithm to reduce the network in time $\mathcal{O}(|\mathcal{A}|)$ before solving the remaining NP-hard problem.

Definition 3. *Let $\mathcal{A}' \subseteq \mathcal{A}$. For $i \in \mathcal{E}$, we define the* successors *of i in $(\mathcal{E}, \mathcal{A}')$ and the* predecessors *of i in $(\mathcal{E}, \mathcal{A})$ as*

$$suc(i, \mathcal{A}') := \{j \in \mathcal{E} \setminus \{i\} : \text{ there exists a directed path from } i \text{ to } j \text{ in } (\mathcal{E}, \mathcal{A}')\},$$
$$pre(i) := \{j \in \mathcal{E} \setminus \{i\} : \text{ there exists a directed path from } j \text{ to } i \text{ in } (\mathcal{E}, \mathcal{A})\}.$$

Using this notation, we introduce the following algorithm:

Algorithm `mark`:
Input: The event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ and source delays $d_j > 0$ ($d_a > 0$) for some events $j \in \mathcal{E}$ (and for some activities $a = (i,j) \in \mathcal{A}_{\text{nice}}$, respectively).
Step 1: Set $\mathcal{A}_\pi := \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : \pi_i < \pi_j\}$.
Step 2: For each source delay $d_j > 0$, $j \in \mathcal{E}$ ($d_a > 0$, $a = (i,j) \in \mathcal{A}_{\text{nice}}$): mark j and all $k \in suc(j, \mathcal{A}_\pi)$.

Note that we do not use all headway activities in the algorithm. The next theorem shows that this is in fact correct. In its proof and throughout this paper, we use $\mathcal{A}_\pi$ as defined in Step 1 of algorithm `mark`.

Theorem 4. *Let $L_a > 0$ $\forall a \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}$, $L_{ij} > 0$ $\forall (i,j) \in \mathcal{A}_{\text{head}}$ and $w_i > 0$ $\forall i \in \mathcal{E}$. Let (x, z, g) be an optimal solution of (DM). Then, the following holds:*

$$i \in \mathcal{E} \text{ is not marked by algorithm } \texttt{mark} \;\Rightarrow\; x_i = \pi_i.$$

Proof. By contradiction. Assume $\exists i \in \mathcal{E}$ such that i does not get marked by algorithm `mark` and $x_i > \pi_i$. Assume that i is an event with minimal x among all such events.

First, by the construction of algorithm `mark`, we have $d_i = 0$, and for each $a = (k,i) \in \mathcal{A}_\pi$, $d_a = 0$ and k isn't marked as well (if i is not marked, also $pre(i)$ is not marked). As k is not marked and we assumed i to be the event with minimal x among all events j with $x_j > \pi_j$ that are not marked, we have $x_k = \pi_k$.

Now, we show that reducing x_i to $\tilde{x}_i := \pi_i < x_i$ also yields a feasible solution for (DM):

– As $d_i = 0$, $\tilde{x}_i = \pi_i$ satisfies (2).
– For all incoming activities $a = (k, i) \in \mathcal{A}_\pi$ we use $x_k = \pi_k$ and $d_a = 0$ to derive $\tilde{x}_i - x_k = \pi_i - \pi_k \geq L_a = L_a + d_a$, hence (3)-(5) hold for each $a = (k, i) \in \mathcal{A}_\pi$.
– For all outgoing activities $(i, k) \in \mathcal{A}$ we obtain $x_k - \tilde{x}_i > x_k - x_i \geq L_a + d_a$ where the last step holds since x is a feasible timetable. Consequently (3)-(5) hold for *all* $a = (i, k) \in \mathcal{A}$.
– Now let $(k, i) \in \mathcal{A}_{\text{head}} \setminus \mathcal{A}_\pi$. If $g_{ki} = 1$, (5) is satisfied due to Corollary 2, so assume $g_{ki} = 0$. We define $\tilde{g}_{ki} := 1$ and $\tilde{g}_{ik} := 0$; then

$$x_k - \tilde{x}_i = x_k - \pi_i \geq \pi_k - \pi_i \geq L_{ki}$$

since $(k, i) \notin \mathcal{A}_\pi$, i.e. $\pi_k > \pi_i$, so (5) holds for (i, k). Due to Corollary 2, (5) also holds for (k, i).

So $(\tilde{x}, z, \tilde{g})$ is a feasible solution with strictly better objective value than (x, z, g) (as $w_i > 0$), a contradiction to the optimality of (x, z, g). □

If we allow $w_i = 0$, then the following modification of Theorem 4 holds:

Theorem 5. *Let $L_a > 0$ $\forall a \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}$, $L_{ij} > 0$ $\forall (i, j) \in \mathcal{A}_{\text{head}}$ and $w_i \geq 0$ $\forall i \in \mathcal{E}$. Then there exists an optimal solution (x, z, g) of (DM) with:*

$$i \in \mathcal{E} \text{ is not marked by algorithm } \texttt{mark} \;\Rightarrow\; x_i = \pi_i.$$

Due to these results, we can use algorithm `mark` to reduce the size of the MIP formulation:

– Run algorithm `mark` on an instance of (DM).
– Delete events that are not marked (unless they have a source-delayed outgoing activity) and activities whose start or end event is not marked (unless they are source-delayed).
– Solve (DM) for this reduced instance.
– Set $x_i = \pi_i$ for all events $i \in \mathcal{E}$ that have been deleted in the second step.

Our numerical results show that reducing the network by algorithm `mark` as a preprocessing step significantly decreases the time needed to solve the IP, see page 8.

The results from Theorem 4 and Theorem 5 also can be used to tighten the upper bound on the minimal size needed for M from Corollary 2: It is sufficient to use the reduced network from above to calculate M.

There is another advantage of our results: Using algorithm `mark`, we can extend the *never-meet property* (see [Sch06,Sch07b]) to capacitated delay management problems, and show that – if this property holds for a given instance of (DM) – our objective (1) coincides with the sum of all delays that each passenger has at his or her final destination. The definition of this property is given next.

Definition 6. *An instance of (DM) has the* never-meet property *if*

– *for each source delay $d_j > 0$ with $j \in \mathcal{E}$ (or $d_a > 0$ with $a = (i, j) \in \mathcal{A}$), $suc(j, \mathcal{A}_\pi)$ is an out-tree, and if*
– *for each pair of different source delays $d_j > 0$ ($d_a > 0$, $a = (i, j)$) and $d_{\tilde{j}} > 0$ ($d_{\tilde{a}} > 0$, $\tilde{a} = (\tilde{i}, \tilde{j})$), we have $suc(j, \mathcal{A}_\pi) \cap suc(\tilde{j}, \mathcal{A}_\pi) = \emptyset$.*

This means that the never-meet property is satisfied if no event can be influenced (directly or indirectly) by more than one source delay. Note that one could define the never-meet property in any network (e.g. also in $\mathcal{N}$) but due to the circles of the headway constraints it would never be satisfied. Hence it is crucial to use the results of Theorem 4.

Lemma 7. *Given an instance of (DM) that satisfies the never-meet property and an optimal solution (x, z, g), assume that $z_a = 1$ for some $a = (i, j) \in \mathcal{A}_{\text{change}}$. Then, for each $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$, k is not marked by algorithm `mark`, applied to $(\mathcal{E}, \mathcal{A} \setminus \{a\})$.*

Proof. By contradiction. Let $a = (i, j) \in \mathcal{A}_{\text{change}}$ and $z_a = 1$ in an optimal solution (x, z, g). Assume that there exists $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$ that is marked by algorithm `mark`, applied to $(\mathcal{E}, \mathcal{A} \setminus \{a\})$. Then, by construction of algorithm `mark`, there exists a directed path p_1 from $\tilde{j}_1$ to k in $(\mathcal{E}, \mathcal{A}_\pi \setminus \{a\})$ with either $d_{\tilde{j}_1} > 0$ or $d_{\tilde{a}_1} > 0$ for some $\tilde{a}_1 = (\tilde{i}_1, \tilde{j}_1)$.

As $z_a = 1$ and $w_a > 0$, there has to be a reason why a is not maintained, namely because of a delay of i. Hence, according to Theorem 4, i is marked by algorithm `mark`, so there exists a directed path p_2 from $\tilde{j}_2$ to i in $(\mathcal{E}, \mathcal{A}_\pi \setminus \{a\})$ with either $d_{\tilde{j}_2} > 0$ or $d_{\tilde{a}_2} > 0$ for some $\tilde{a}_2 = (\tilde{i}_2, \tilde{j}_2)$. As $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$, p_2 can be extended to a path p_3 from $\tilde{j}_2$ to k in $(\mathcal{E}, \mathcal{A}_\pi)$ that contains a.

This is a contradiction to the never-meet property: either $\tilde{j}_1 = \tilde{j}_2$, then $suc(\tilde{j}_1, \mathcal{A}_\pi)$ is not an out-tree as we have two different paths p_1 (not containing a) and p_3 (containing a) from $\tilde{j}_1$ to k, or $\tilde{j}_1 \neq \tilde{j}_2$, then $suc(\tilde{j}_1, \mathcal{A}_\pi) \cap suc(\tilde{j}_2, \mathcal{A}_\pi) \supseteq \{k\} \neq \emptyset$. □

Corollary 8. *Given an instance of (DM) that satisfies the never-meet property and an optimal solution (x, z, g), assume that $z_a = 1$ for some $a = (i, j) \in \mathcal{A}_{\text{change}}$. Then, for each $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$, $x_k = \pi_k$.*

Proof. If, in an optimal solution, $z_a = 1$ for some $a \in \mathcal{A}_{\text{change}}$, this solution is also an optimal solution for the same instance with event-activity network $(\mathcal{E}, \mathcal{A} \setminus \{a\})$. From Lemma 7 we know that k is not marked for all $k \in suc(j, \mathcal{A}_\pi)$. Theorem 4 hence completes the proof. □

Theorem 9. *If the never-meet property holds, (DM) is equivalent to minimizing the sum of all delays of all passengers at their final destinations.*

Proof. The proof can be done analogously to the result in [Sch07b] since the ingredient needed is provided in Corollary 8. □

Numerical Results

To test the efficiency of our reduction, we used railway data from the Harz region in the center of Germany (in form of a periodic timetable, including headway and turnover activities), originally used in [LSS+07]. We consider all passenger railway lines within this region as well as 9 long-distance lines. The dataset contains 598 stations, 92 trains (vehicles) and 30 lines, each line with two directions. We take into account all events and all activities that take place in an 8-hours time window. Details on how the periodic timetable is expanded to an aperiodic event-activity network are described in [LSS+07]. See Figure 1 for a sketch of the part of the German railway network which we consider.

The resulting event-activity network contains 21 269 events and 39 985 activities. We generated 1 000 different delay scenarios. In each delay scenario, 25 randomly chosen driving or waiting activities have been delayed by a random delay between 60 and 1 200 seconds. In both cases, (DM) was solved using Xpress-MP 2007B on an AMD Opteron 275 system with 4 GB RAM.

The results clearly show the benefit of a preprocessing step based on algorithm `mark`: In 13 of 1 000 test cases, the original problem could not be solved due to an "out of memory" error – in all those cases, we got an optimal solution for the reduced problem. In average, the time needed to solve the reduced problems was only 19.1% of the time needed to solve the original problems. Especially, in 929 cases, the computation time was reduced to less than 50%, and in 332 cases (nearly one third of all cases), it was even reduced to less than 10%.

4 Heuristics

Although the results from Theorem 4 and Theorem 5 can be used for introducing a preprocessing step that speeds up the computation of an optimal solution, very large instances of (DM) cannot be solved exactly in a reasonable amount of time. To be able to provide at least some solution

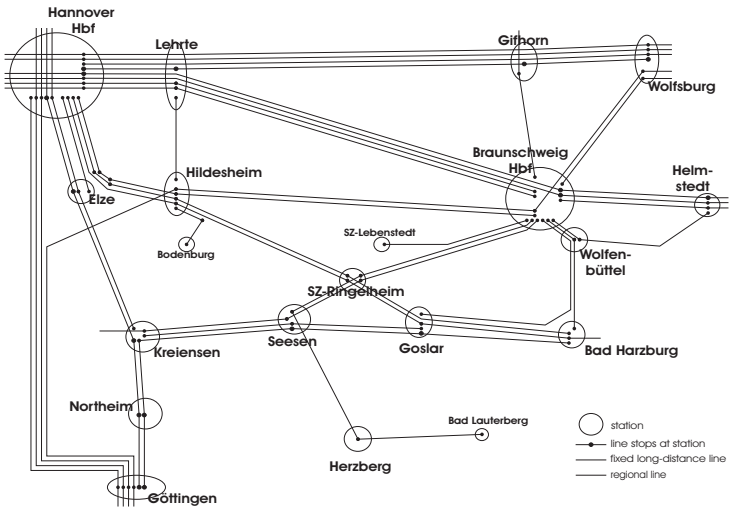


Fig. 1. The part of the German railway network which is considered for our numerical results. (The Figure is taken from [LSS+07].)

for these cases, we introduce heuristics that can solve even large instances by approximation. The idea of our heuristics is to fix the order of trains (i.e. the variables g_{ij}) in advance and solve the remaining uncapacitated delay management problem (UDM). Such a fixing can be done in many ways; four of them will be discussed next. In the first approach we fix the order of the trains according to the original timetable.

Algorithm `First scheduled, first served` **(FSFS):**

1. Set $\mathcal{A}_{\text{fix}} := \{(i,j) \in \mathcal{A}_{\text{head}} : \pi_i \leq \pi_j\}$.
2. Compute the exact solution of (UDM($\mathcal{A}_{\text{fix}}$)).

In the next heuristic, we first neglect all capacity constraints and solve the uncapacitated delay management problem (UDM). We then fix the order of the trains according to the optimal disposition timetable x of (UDM).

Algorithm `First rescheduled, first served` **(FRFS):**

1. Solve the corresponding problem (UDM) and obtain disposition timetable x.
2. Set $\mathcal{A}_{\text{fix}} := \{(i,j) \in \mathcal{A}_{\text{head}} : x_i \leq x_j\}$.
3. Compute the exact solution of the corresponding instance of (UDM($\mathcal{A}_{\text{fix}}$)).

Solving an additional instance of (UDM) in the first step increases the running time of FRFS compared to FSFS. In the third heuristic, we again start by solving the corresponding instance of (UDM) and fix the order of trains according to an optimal disposition timetable, but here we additionally fix the wait-depart decisions obtained from the solution of (UDM) such that we can compute a feasible disposition timetable for (DM) by applying the critical path method. In contrast to FRFS, Step 3 can now be solved very efficiently.

Algorithm `FRFS with early connection fixing` **(EARLYFIX):**

1. Solve the corresponding problem (UDM) and obtain a disposition timetable x and values for z.
2. Set $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : x_i \leq x_j\}$.
3. Compute the solution of (PP($\mathcal{A}_{\text{fix}}$)).

Since the first step in both FRFS and in EARLYFIX is to solve (UDM), we obtain from these heuristics not only an approximation of the optimal solution, but also a lower bound on its objective value, and hence their absolute errors can be bounded a posteriori by

$$F^{\text{FRFS}} - F^{\text{UDM}} \text{ and } F^{\text{EARLYFIX}} - F^{\text{UDM}},$$

respectively, where F^{FRFS} and F^{EARLYFIX} are the objective values of FRFS and EARLYFIX. Comparing these heuristics we obtain the following result.

Lemma 10. $F^{\text{DM}} \leq F^{\text{FRFS}} \leq F^{\text{EARLYFIX}}$.

The last heuristic we tested is the only one with polynomial runtime. It is a modification of the FSFS heuristic we presented first. As we do in FSFS, we fix the order of trains according to the original timetable π, but instead of solving the remaining problem exactly, we use a heuristic approach to fix the wait-depart decisions. The idea is to maintain the "most important" connections and do not care about the less important ones.

Algorithm `FSFS with priority-based fixing` **(PRIORITY):**

1. Maintain the "most important" connections:
   - Sort the changing edges in descending order according to their weights w_a.
   - Set $z_a = 0$ for the first k% of the connections.
2. Set $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : \pi_i \leq \pi_j\}$.
3. Compute the exact solution of (PP($\mathcal{A}_{\text{fix}}$)).

Comparing PRIORITY to FSFS, we obtain the following relation.

Lemma 11. *Let F^{FSFS} and F^{PRIORITY} denote the objective values of the solutions computed by FSFS and PRIORITY, respectively. Then $F^{\text{DM}} \leq F^{\text{FSFS}} \leq F^{\text{PRIORITY}}$*

Finding bounds on the relative error of these heuristics is in general not possible: the results of all heuristics might get arbitrarily bad compared to the optimal solution. In our first result we prove that fixing the priority decisions according to the original timetable might become arbitrarily bad, while Lemma 13 shows that fixing the priority decisions according to the optimal solution of the corresponding (UDM) might also become arbitrarily bad. Hence, both groups of heuristics FSFS and PRIORITY as well as FRFS and EARLYFIX can become arbitrarily bad. However, we are able to bound the relative error of EARLYFIX using the input data of the special instance in Theorem 15.

Lemma 12. *Let* HEU *be a heuristic that solves (DM), fixing the g_{ij} variables as they are set in the original timetable, and let F^{HEU} its objective value. Then for each $k \in \mathbb{N}$, there exists an instance of (DM) such that*

$$\frac{F^{\text{HEU}} - F^{\text{DM}}}{F^{\text{DM}}} > k.$$

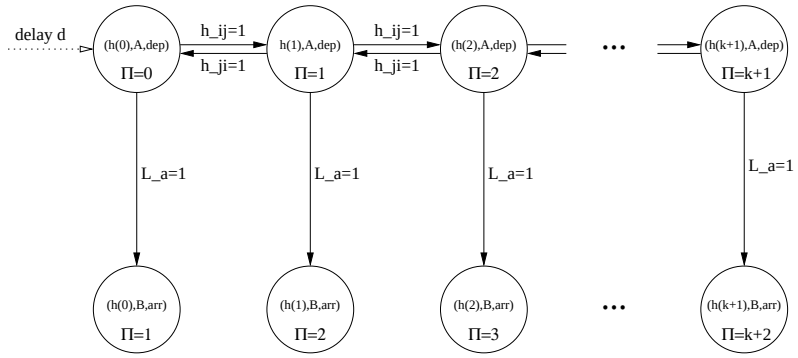


Fig. 2. Event-activity network for the proof of Lemma 12.

Proof. Let $k \in \mathbb{N}$. Assume that we have two stations A and B and $k+2$ trains $h_0, h_1, \ldots, h_{k+1}$. All trains drive from station A to station B. By (t, s, arr) and (t, s, dep), we denote the arrival of train t at station s and its departure from that station. $\pi(t, s, \text{arr})$ denotes the time of such an event in the original timetable. In the original timetable, the trains in our instance leave station A in the order $h_0, h_1, \ldots, h_{k+1}$ at the times $\pi(h_i, A, \text{dep}) = i$ and arrive at station B at the times $\pi(h_i, B, \text{arr}) = i + 1$, $i \in \{0, \ldots, k+1\}$. For each $i \in \{0, \ldots, k\}$, the departure of train h_i and the departure of train h_{i+1} are connected by a pair of headway edges. All weights and all lower bounds are set to 1. The resulting event-activity network is shown in Figure 2.

Now, assume that (h_0, A, dep) is delayed by $d \geq k+2$. In the optimal solution, the trains $h_1, \ldots, h_{k+1}$ leave and arrive on time, while train h_0 has a delay of d, so $F^{\text{DM}} = d$. If we solve the problem by a heuristic that sets the g_{ij} variables as they are set in the original timetable without delays, all trains get a delay of at least d, so $F^{\text{HEU}} \geq (k+2) \cdot d$, hence

$$\frac{F^{\text{HEU}} - F^{\text{DM}}}{F^{\text{DM}}} \;\geq\; \frac{(k+2) \cdot d - d}{d} \;=\; k+1 \;>\; k.$$

□

Similarly, one can show the following result concerning FSFS and PRIORITY.

Lemma 13. *Let* HEU *be a heuristic that solves (DM), fixing the g_{ij} variables as they are set in the optimal solution of the corresponding problem (UDM). Then for each $k \in \mathbb{N}$, there exists an instance of (DM) such that*

$$\frac{F^{\text{HEU}} - F^{\text{DM}}}{F^{\text{DM}}} \;>\; k.$$

However – as we will show at then end of this Section – the heuristics do not behave as bad as one might think regarding the results above. Moreover, using the input data of the specific instance we might be able to derive upper bounds. This is exemplarily done to bound the relative error of EARLYFIX.

Lemma 14. *Let x^{relax} be an optimal solution of (PP($\mathcal{A}_{\text{fix}}$)) with events $\mathcal{E}$ and activities $\mathcal{A}_{\text{fix}} = \mathcal{A}_{\text{nice}} \cup \mathcal{A}^{\text{fix}}_{\text{change}}$ and x^{cap} an optimal solution of (PP($\mathcal{A}_{\text{fix}}$)) with events $\mathcal{E}$ and activities $\mathcal{A}_{\text{fix}} = \mathcal{A}_{\text{nice}} \cup \mathcal{A}^{\text{fix}}_{\text{change}} \cup \mathcal{A}^{\text{fix}}_{\text{head}}$. Let $\mathcal{A}^1 := \mathcal{A}^{\text{fix}}_{\text{head}} \cap \mathcal{A}_\pi$ and $\mathcal{A}^2 := \mathcal{A}^{\text{fix}}_{\text{head}} \setminus \mathcal{A}_\pi$. Then, we have*

$$x_i^{\text{cap}} \;\leq\; x_i^{\text{relax}} + \sum_{\substack{(k,l) \in \mathcal{A}^1: \\ k \in pre(i)}} (x_k^{\text{relax}} - \pi_k) + \sum_{\substack{(k,l) \in \mathcal{A}^2 \\ k \in pre(i)}} (x_k^{\text{relax}} + L_{kl}) \quad \forall i \in \mathcal{E}. \tag{15}$$

Proof. An optimal solution of (PP($\mathcal{A}_{\text{fix}}$)) can be computed by applying the critical path method that has been introduced in Section 2. We prove (15) by induction and distinguish two cases depending which term in (12) gets maximal. For $k = 1$ and for $x_k^{\text{cap}} = \pi_k + d_k$, (15) is true. We therefore assume $x_k^{\text{cap}} > \pi_k + d_k$ and that (15) is true for all $j < k$. Let

$$\tilde{a} = (j,k) \;:=\; \underset{a=(j,k)\in\mathcal{A}_{\text{fix}}}{\operatorname{argmax}} \; x_j^{\text{cap}} + L_a + d_a.$$

Case 1: Assume that $\tilde{a} \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}^{\text{fix}}$. Then, $x_k^{\text{cap}} = x_j^{\text{cap}} + L_{\tilde{a}} + d_{\tilde{a}}$, and using $x_k^{\text{relax}} - x_j^{\text{relax}} \geq L_{\tilde{a}} + d_{\tilde{a}}$, (15) to estimate x_j^{cap} and $pre(j) \subset pre(k)$, we see that (15) is satisfied.

Case 2: Assume that $\tilde{a} \in \mathcal{A}^1$. Then, $d_{\tilde{a}} = 0$ and $x_k^{\text{cap}} = x_j^{\text{cap}} + L_{jk}$. Using (15), we get

$$x_k^{\text{cap}} \;\leq\; x_k^{\text{relax}} + \sum_{\substack{(l,m)\in\mathcal{A}^1:\\ l\in pre(j)}} (x_l^{\text{relax}} - \pi_l) + \sum_{\substack{(l,m)\in\mathcal{A}^2:\\ l\in pre(j)}} (x_l^{\text{relax}} + L_{lm}) + x_j^{\text{relax}} - x_k^{\text{relax}} + L_{jk}. \tag{16}$$

Using $\pi_k - \pi_j \geq L_{jk}$, $x_k^{\text{relax}} \geq \pi_k$ and $pre(j) \subset pre(k)$, we see that (15) holds.

Case 3: Assume that $\tilde{a} \in \mathcal{A}^2$. Then, $d_{\tilde{a}} = 0$ and $x_k^{\text{cap}} = x_j^{\text{cap}} + L_{jk}$. As in the second case, we get inequality (16). We use $x_k^{\text{relax}} \geq 0$, move $x_j^{\text{relax}} + L_{jk}$ to the second sum and use $pre(j) \subset pre(k)$ to prove the lemma for the third case. □

We can use Lemma 14 to get an upper bound on the relative error of EARLYFIX: We replace x^{relax} by x^{DM} and x^{cap} by x^{EARLYFIX}, and define $\mathcal{A}_{\text{head}}^{\text{fix}} = \{(i,j) \in \mathcal{A}_{\text{head}} : x_i^{\text{DM}} \leq x_j^{\text{DM}}\}$. Using the delay $y_i = x_i - \pi_i$ of event i instead of its time x_i in the disposition timetable, we have

$$y_i^{\text{EARLYFIX}} - y_i^{\text{DM}} \;\leq\; \sum_{\substack{(k,l)\in\mathcal{A}^1:\\ k\in pre(i)}} y_k^{\text{DM}} + \sum_{\substack{(k,l)\in\mathcal{A}^2:\\ k\in pre(i)}} (y_k^{\text{DM}} + \pi_k + L_{kl}) \;\leq\; F^{\text{UDM}} + \sum_{(k,l)\in\mathcal{A}_{\text{head}}^{\text{fix}}} (L_{kl} + \pi_k),$$

where the second inequality holds if we assume $w_i \geq 1 \forall\; i \in \mathcal{E}$. With this assumption, we hence obtain

$$F^{\text{EARLYFIX}} - F^{\text{DM}} \;\leq \left(F^{\text{UDM}} + \sum_{(k,l)\in\mathcal{A}_{\text{head}}^{\text{fix}}} (L_{kl} + \pi_k) \right) \sum_{i\in\mathcal{E}} w_i.$$

If, in addition, $\mathcal{A}^2 = \emptyset$, we have $F^{\text{EARLYFIX}} - F^{\text{DM}} \;\leq\; F^{\text{UDM}} \sum_{i\in\mathcal{E}} w_i$. This yields the following result.

Theorem 15. *Consider an instance of (DM) with weights $w_i \geq 1$ for all $i \in \mathcal{E}$.*

a) If the solution x^{DM} of (UDM) satisfies $\pi_i \leq \pi_j \;\Rightarrow\; x_i^{\text{DM}} \leq x_j^{\text{DM}} \quad \forall\, (i,j) \in \mathcal{A}_{\text{head}}$, then

$$\frac{F^{\text{EARLYFIX}} - F^{\text{DM}}}{F^{\text{DM}}} \;\leq\; \sum_{i\in\mathcal{E}} w_i.$$

b) If $F^{\text{DM}} \geq 1$, we have

$$\frac{F^{\text{EARLYFIX}} - F^{\text{DM}}}{F^{\text{DM}}} \;\leq\; \left(1 + \sum_{(k,l)\in\mathcal{A}^1} (L_{kl} + \pi_k) \right) \sum_{i\in\mathcal{E}} w_i.$$

The theorem gives rise to the assumption that the quality of the solution depends on the size of $\sum_{(k,l)\in\mathcal{A}^1} L_{kl}$.

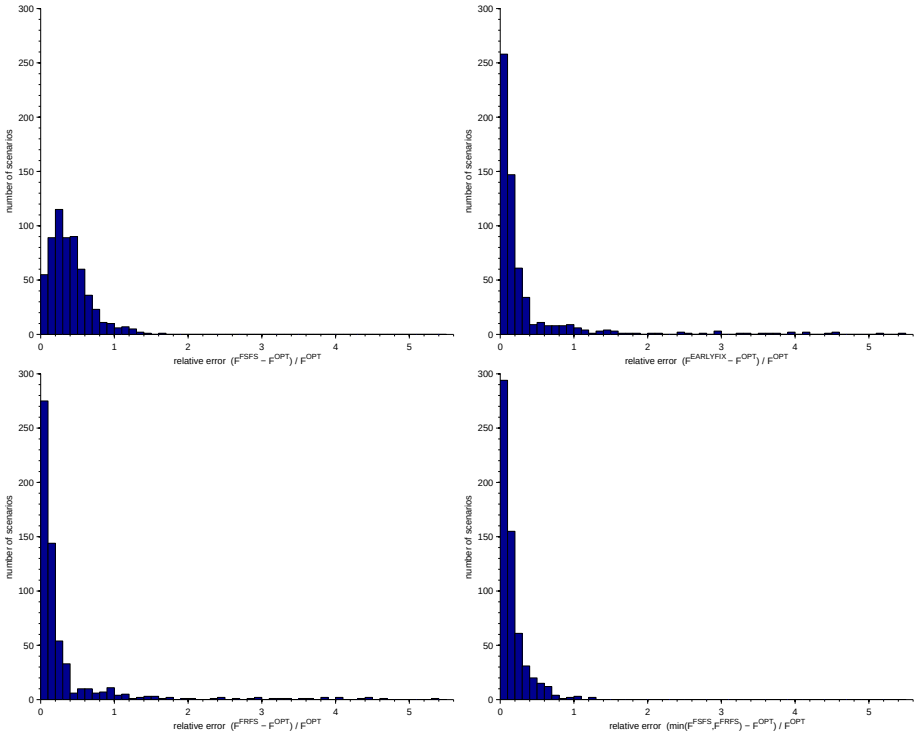


Fig. 3. The relative error of FSFS, EARLYFIX, FRFS and of the combination of FSFS and FRFS (observation period of 3 hours, 7 104 events, 9 570 activities).

Numerical Results

The dataset on which our numerical results for the heuristics are based is the same dataset as in Section 3. The sizes of the event-activity network for different observation periods are stated in Table 2. The IP formulation was solved using Xpress-MP 2006 on a Pentium IV 3 GHz processor with 2 GB RAM. We generated about 600 different delay scenarios; in each of them, we assigned 25 randomly generated source delays of 1-20 minutes to 25 randomly chosen driving and waiting activities. We also know the weights $\tilde{w}_a$ of the changing activities $a \in \mathcal{A}_{\text{change}}$. We hence set $w_i = 1$ for all events $i \in \mathcal{E}$ and $w_a = \frac{\tilde{w}_a}{\bar{w}}$ for all $a \in \mathcal{A}_{\text{change}}$. $\bar{w}$ is the arithmetic mean of the weights $\tilde{w}_a$ which is used in order to prevent an overestimation of the missed connections.

In Figure 3, we present four histograms of the relative errors for the heuristics FSFS, EARLYFIX and FRFS and for the approach running both FSFS and FRFS and taking the better solution. We took into account all events and all activities that take place during a fixed observation period of 3 hours. On the x-axis we graphed intervals of 0.1 length describing the relative error. The first interval corresponds to a relative error between 0 and 0.1, the second interval to a relative error between 0.1 and 0.2, and so on. We show in how many of the about 600 different delay scenarios the relative error of the respective heuristic takes a value in an interval of length 0.1 – for example, in about 55 scenarios out of 600, the relative error of FSFS is in the interval [0, 0.1].

FRFS is slightly better than EARLYFIX concerning the quality of their solutions. For both of them, the number of scenarios with a small relative error is significantly higher than for FSFS. On the other hand, there are some scenarios in which EARLYFIX and FRFS have a very high relative error – FSFS does not have these outliers. If we combine FSFS and FRFS – this means that for each scenario, we take the solution with the smaller objective value – we benefit from the large number of scenarios with a small relative error in FRFS and from the fact that FSFS does not have outliers as FRFS does have.

heuristic	observation period of 3 hours	6 hours	10 hours
FSFS	141 (23.58%)	239 (39.97%)	263 (43.98%)
EARLYFIX	219 (36.62%)	83 (13.88%)	75 (12.54%)
FRFS	457 (76.42%)	361 (60.37%)	336 (56.19%)

Table 1. How often (out of 598 different scenarios) is FSFS, EARLYFIX and FRFS at least as good as the two other heuristics, w.r.t different observation periods?

Table 1 shows the quality of FSFS, EARLYFIX and FRFS compared to each other. We specify for each heuristic in how many cases it computes a solution at least as good as the solutions of the other heuristics. We take into account different observation periods. For larger event-activity networks, EARLYFIX performs quite bad, while the number of scenarios in which FSFS computes the best solution grows significantly.

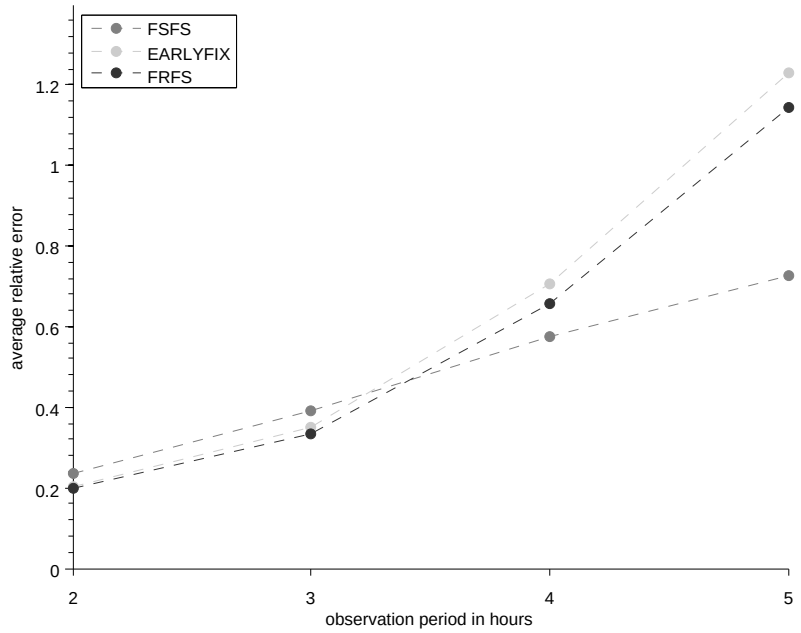


Fig. 4. Average relative error of FSFS, EARLYFIX and FRFS for different observation periods between two and five hours.

In Figure 4, we show how the relative errors of FSFS, EARLYFIX and FRFS grow with the length of the observation period, i.e. with the size of the relevant event-activity network. The larger the event-activity network, the larger the relative error of all heuristics.

In Table 2, we finally specify the runtime of the exact solution and of the heuristics FSFS, FRFS and EARLYFIX for different observation periods (i.e. for different sizes of the event-activity network). An observation period of k hours means that we considered events and activities during a fixed k-hours time slot. It turns out that all heuristics are significantly faster than the optimal solution (calculated via the ILP formulation by Xpress). EARLYFIX clearly outperforms FSFS and FRFS by a factor of 3. FSFS and FRFS are nearly equal considering the computation time.

size of the event-activity network					runtime (in seconds) of algorithm			
hours	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{A}_{\text{head}}\rvert$	$\lvert\mathcal{A}_{\text{change}}\rvert$	exact	FSFS	EARLYFIX	FRFS
2	4 726	5 865	1 110	125	19.45	0.24	0.11	0.26
3	7 104	9 570	2 378	187	185.33	0.46	0.17	0.49
4	9 446	14 079	4 428	307	584.66	0.69	0.25	0.74
5	11 824	18 605	6 514	369	1 075.52	0.91	0.31	1.00
6	14 166	23 396	8 846	489	-	1.16	0.39	1.26
8	18 888	32 673	13 260	632	-	1.65	0.53	1.83
10	23 596	41 992	17 656	852	-	2.04	0.68	2.34
15	33 718	61 944	27 138	1 209	-	3.01	1.01	3.63

Table 2. Average runtime for different sizes of the event-activity network.

5 Conclusion and Further Research

In this paper, we presented and analyzed an integer programming formulation for the capacitated delay management problem. We suggest a reduction technique and four different heuristics. In our analysis it turns out that the headway activities which are not in the same direction as in the original timetable play a basic role. We were also able to give a reasonable definition of the never-meet property and to extend properties known from the uncapacitated delay management problem.

There are more properties of the never-meet property that are currently exploited, e.g. to extend the linear-time algorithm of the uncapacitated problem to the capacitated case. We also work on a deeper analysis of the heuristics and on new approaches such as machine-based learning techniques. Moreover, other issues should be considered to make the approach applicable in practice. Sometimes a change of the vehicle routes is appropriate to reduce delays, and often it is necessary to include the microscopic routes of the trains, in particular at large stations.

Acknowledgment. We want to thank Jens Dupont of *Deutsche Bahn* and Christian Liebchen of TU Berlin for providing the data for the case study.

References

[BGJ+05] N. Bissantz, S. Güttler, J. Jacobs, S. Kurby, T. Schaer, A. Schöbel, and S. Scholl. DisKon - Disposition und Konfliktlösungs-management für die beste Bahn. *Eisenbahntechnische Rundschau (ETR)*, 45(12):809–821, 2005. (in German).

[BHLS07] Andre Berger, Ralf Hoffmann, Ulf Lorenz, and Sebastian Stiller. Online delay management: Pspace hardness and simulation. Technical Report ARRIVAL-TR-0097, ARRIVAL Project, 2007.

[CS07] C. Conte and A. Schöbel. Identifying dependencies among delays. In *proceedings of IAROR 2007*, 2007. ISBN 978-90-78271-02-4.

[Elm77] S.E. Elmaghraby. *Activity Networks.* Wiley Interscience Publication, 1977.

[Gat07] M. Gatto. *On the Impact of Uncertainty on Some Optimization Problems: Combinatorial Aspects of Delay Management and Robust Online Scheduling.* PhD thesis, ETH Zürich, 2007.

[GGJ+04] M. Gatto, B. Glaus, R. Jacob, L. Peeters, and P. Widmayer. Railway delay management: Exploring its algorithmic complexity. In *Proc. 9th Scand. Workshop on Algorithm Theory (SWAT)*, volume 3111 of *LNCS*, pages 199–211, 2004.

[GHL08] L. Giovanni, G. Heilporn, and M. Labbé. Optimization models for the delay management problem in public transportation. *European Journal of Operational Research*, 189(3):762–774, September 2008. to appear.

[GJPS05] M. Gatto, R. Jacob, L. Peeters, and A. Schöbel. The computational complexity of delay management. In D. Kratsch, editor, *Graph-Theoretic Concepts in Computer Science: 31st International Workshop (WG 2005)*, volume 3787 of *Lecture Notes in Computer Science*, 2005.

[GJPW07] M. Gatto, R. Jacob, L. Peeters, and P. Widmayer. On-line delay management on a single train line. In *Algorithmic Methods for Railway Optimization*, Lecture Notes in Computer Science. Springer, 2007. to appear.

[Gov98] R.M.P. Goverde. The max-plus algebra approach to railway timetable design. In *Computers in Railways VI: Proceedings of the 6th international conference on computer aided design, manufacture and operations in the railway and other advanced mass transit systems, Lisbon, 1998*, pages 339–350, 1998.

[GS07] A. Ginkel and A. Schöbel. To wait or not to wait? The bicriteria delay management problem in public transportation. *Transportation Science*, 41(4):527–538, 2007.

[LSS+07] C. Liebchen, M. Schachtebeck, A. Schöbel, S. Stiller, and A. Prigge. Computing delay-resistant railway timetables. Technical Report TR-0066, ARRIVAL Report, 2007. see http://arrival.cti.gr/index.php/Documents/Main.

[Nac98] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. Deutsches Zentrum für Luft– und Raumfahrt, Institut für Flugführung, Braunschweig, 1998. Habilitationsschrift.

[RdVM98] B. De Schutter R. de Vries and B. De Moor. On max-algebraic models for transportation networks. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 457–462, Cagliari, Italy, 1998.

[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. *Customer-oriented optimization in public transportation*. Optimization and Its Applications. Springer, New York, 2006.

[Sch07a] A. Schöbel. Capacity constraints in delay management. 2007. ARRIVAL Report TR-0017.

[Sch07b] 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.

[SM99] L. Suhl and T. Mellouli. Requirements for, and design of, an operations control system for railways. In *Computer-Aided Transit Scheduling*. Springer, 1999.

[SMBG01] L. Suhl, T. Mellouli, C. Biederbick, and J. Goecke. Managing and preventing delays in railway traffic by simulation and optimization. In *Mathematical methods on Optimization in Transportation Systems*, pages 3–16. Kluwer, 2001.

Line Planning on Paths and Tree Networks with Applications to the Quito Trolebús System

Luis M. Torres[1], Ramiro Torres[1], Ralf Borndörfer[2], and Marc E. Pfetsch[2]

[1] Escuela Politécnica Nacional, Quito, Ecuador,
{ltorres,rtorres}@math.epn.edu.ec
[2] Zuse Institute Berlin, 14195 Berlin, Germany,
{borndoerfer,pfetsch}@zib.de

Abstract. Line planning is an important step in the strategic planning process of a public transportation system. In this paper, we discuss an optimization model for this problem in order to minimize operation costs while guaranteeing a certain level of quality of service, in terms of available transport capacity. We analyze the problem for path and tree network topologies as well as several categories of line operation that are important for the Quito Trolebús system. It turns out that, from a computational complexity worst case point of view, the problem is hard in all but the most simple variants. In practice, however, instances based on real data from the Trolebús System in Quito can be solved quite well, and significant optimization potentials can be demonstrated.

1 Introduction

The major cities of South America are facing an enormous and constantly increasing demand for transportation and, unfortunately, also increase vehicular congestion, with all its negative effects. In Quito, the elongated topography of the city with 1.8 millions inhabitants (the urban area being 60 km long and 8 km wide) aggravates vehicular congestion even more, such that traffic almost completely breaks down during rush hours. As a consequence, the local government faces the necessity of improving the public mass transit system.

A low-cost option that has produced satisfactory results in recent years has been the implementation of major corridors of transportation. These corridors consist of street tracks that are reserved exclusively for high-capacity bus units, which, in this way, can operate independently of the rest of the traffic. Even though the topology of a corridor is extremely simple (just a path), bus operation on it is non-trivial. In fact, it is usually organized in a complex system of several dozen lines, which cover, in an overlapping way, different parts of the corridor, and which can operate in different ways, e.g., as "normal lines" or as "express lines" (stopping only at distinguished express stations), as "open lines" (unidirectional) or "closed lines" (bidirectional lines), and in any combination of these categories. The corridor lines are often complemented by feeding lines that transport passengers between special transshipment terminals of the corridor and the nearby neighborhoods.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1583

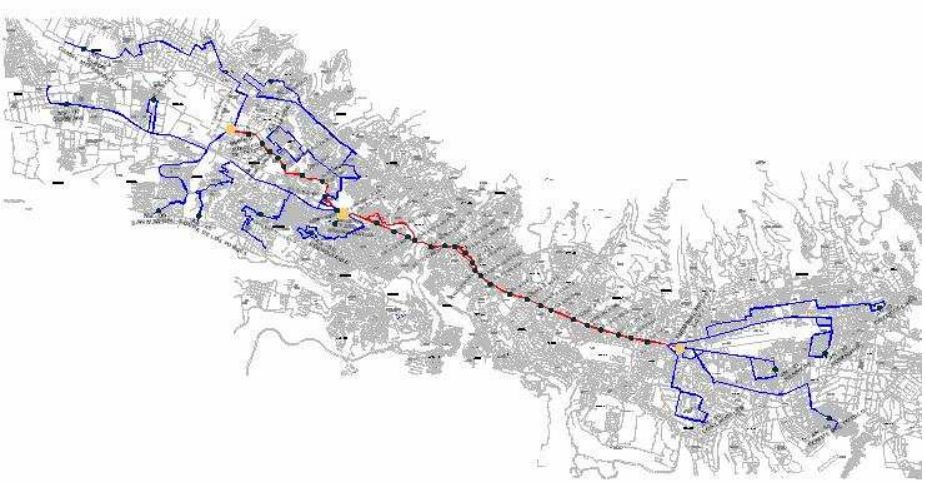


Fig. 1: Trolebús system and feeder line system in Quito.

In Quito, the most important of such corridors is the so-called *Trolebús System* (TS), see Figure 1. TS is currently the largest public transportation system in Quito, carrying around 250,000 passengers daily. However, the dramatic increase in transportation demand has had a negative impact on the quality of service, with overcrowded buses and long waiting times being commonly experienced by passengers. At the same time, operation costs have been continuously increasing. With the aim of contributing to the improvement of this situation, we have been working on optimization models that can be applied to improve the operation of the TS and similar corridor transportation systems. The question that we investigate is whether the design of the corridor line system can be optimized using mathematical methods in order to improve the quality of service and/or lower operation costs by a better vehicle utilization.

Mathematical optimization approaches to line planning have received growing attention in the operations research and the mathematical programming community in the last two decades, see Odoni, Rousseau, and Wilson [1] and Bussieck, Winter, and Zimmermann [2] for an overview. In particular, integer programming approaches to line planning have been considered since the late nineties. Bussieck, Kreuzer, and Zimmermann [3] (see also Bussieck [4]) and Claessens, van Dijk, and Zwaneveld [5] both propose cut-and-branch approaches to select lines from a previously generated pool of potential lines. Both articles are based on a "system-split" of the demand, i.e., an a priori distribution of the passenger flow on the arcs of the transportation network; these "aggregated demands" are then covered by lines of sufficient capacity. Bussieck, Lindner, and Lübbecke [6] extend this work by incorporating nonlinear components. Goossens, van Hoesel, and Kroon [7,8] improve the models and algorithms and show that real-world railway problems can be solved within reasonable time and quality. Approaches that integrate line planning and passenger routing have recently

been proposed by Borndörfer, Grötschel, and Pfetsch [9,10], and by Schöbel and Scholl [11,12]. The latter authors consider an expanded line-network that allows to minimize the number of transfers or the transfer time.

All of these articles consider general network topologies, but do not analyze line operation categories such as express lines, or open lines, probably because the line planning problem on general graphs is already hard without them. The corridor topology, however, opens up a chance to investigate complex line operation categories in a practically relevant setting. It also brings up the question whether perhaps some cases associated with different line operation categories can be solved in polynomial time. It will turn out in Section 3 that this is indeed the case if only closed lines and a homogeneous vehicle fleet are used; in all other cases, however, the problem is hard (there is one open case left). From a practical point of view, however, TS instances can be solved quite well. Indeed, our results show significant optimization potentials with respect to the currently operated solution, see Section 4.

2 A Flow-Based Model for Line Planning

We consider a bus transportation network as a digraph $D = (V, A)$, where each bus station is represented by a node $v \in V$ and arcs represent direct links between stations, i.e., $(i, j) \in A$ if and only if some bus may visit station j directly after station i. The fleet of buses is often heterogeneous; for instance, in Quito it contains trolley-buses and several other types of buses used for the feeding lines. We call a specific type of bus a transportation *mode* and define $\mathcal{M}$ to be the set of all transportation modes in the system, where each transportation mode $m \in \mathcal{M}$ has a specific capacity $\kappa_m \in \mathbb{Z}^+$. For each $m \in \mathcal{M}$, certain stations referred to as *terminals* are identified, where buses of mode m may start or end a service route. An *open line* for a mode m is a directed path whose first and last nodes are different terminals. Similarly, a *closed line* for m is a circuit containing at least one terminal. We consider for each $m \in \mathcal{M}$ a *line pool* $\mathcal{L}^m$, i.e., a set of a priori selected (open or closed) lines that can potentially be established. We denote by $\mathcal{L} := \cup_{m \in M} \mathcal{L}^m$ the set of all possible lines and by $\mathcal{L}_a^m$ the set of lines of mode m using arc a. For a line $\ell \in \mathcal{L}$, $c_\ell \in \mathbb{R}_+$ is the cost of a single trip via ℓ. Transportation demand is usually expressed in terms of an origin-destination matrix $(d_{uv}) \in \mathbb{Z}_+^{V \times V}$, where each entry d_{uv} indicates the number of passengers traveling from station u to station v within a certain time horizon T. In the following we assume that each passenger has been routed along some specific directed (u, v)-path in a preprocessing step, such that an *aggregated transportation demand* g_a on each arc a of the network has been computed.

We will consider three network topologies that are related to the TS structure. On the main corridor, trolley-buses move on a single path and are usually not allowed to overtake. This suggests to define a transportation network consisting of two directed paths (one for each transportation direction). Any line moving from a station u to a station v must stop at all intermediate stations. We call such a network topology a *Quito-Graph* (QG). However, transport au-

thorities are considering the possibility of allowing trolley-buses to overtake at certain segments of the main corridor in the future. This would make it possible to introduce *express lines* that stop only at certain stations. The trips between two express stations can be modeled using respectively longer arcs. We call a network of this type a *Quito-Hopping-Graph* (QHG). Finally, when considering both feeding lines and the main corridor together, we observe that the TS network can be modeled as a *tree*, since feeding lines are simple paths that start at transshipment stations along the main corridor.

The *line planning problem* is to choose a set of lines $L \subseteq \mathcal{L}$ and frequencies for the lines in L in such a way that there is enough transportation capacity to cover the aggregated demand on each arc of the network. It can be formulated as an integer programming problem, that we denote by *Demand Covering Model with Fixed Costs* (DCM-FC):

$$\min \quad \sum_{m \in \mathcal{M}} \sum_{\ell \in \mathcal{L}^m} (c_\ell \, f_\ell + K_\ell \, y_\ell) \tag{1}$$

subject to

$$\sum_{m \in \mathcal{M}} \sum_{\ell \in \mathcal{L}_a^m} \kappa_m \, f_\ell \geq g_a, \qquad \forall \, a \in A \tag{2}$$

$$0 \leq f_\ell \leq f_\ell^{\max} \, y_\ell \qquad \forall \, \ell \in \mathcal{L} \tag{3}$$

$$f_\ell \in \mathbb{Z}_+ \qquad \forall \, \ell \in \mathcal{L} \tag{4}$$

$$y_\ell \in \{0, 1\} \qquad \forall \, \ell \in \mathcal{L}. \tag{5}$$

Here, f_ℓ is an integer variable representing the frequency assigned to line $\ell \in \mathcal{L}$, and y_ℓ is a 0/1-variable that indicates whether a line is chosen in the solution ($y_\ell = 1$) or not ($y_\ell = 0$). The cost of line $\ell \in \mathcal{L}$ involves a fixed component K_ℓ as well as an operating cost $c_\ell \, f_\ell$ that depends on the frequency. The objective function (1) aims at minimizing the total operation costs. Constraints (2) ensure that the aggregated transportation demand is covered. Constraints (3) couple the line selection variables y_ℓ and the frequency variables f_ℓ and they impose upper bounds $f_\ell^{\max}$, for all $\ell \in \mathcal{L}$ on line frequencies. Finally, (4) and (5) are integrality constraints for the frequencies.

When fixed costs are zero ($K_\ell = 0, \forall \ell \in \mathcal{L}$), the model simplifies to the following form, that we denote by *Demand Covering Model* (DCM):

$$\min \quad \sum_{m \in \mathcal{M}} \sum_{\ell \in \mathcal{L}^m} c_\ell \, f_\ell \tag{6}$$

subject to

$$\sum_{m \in \mathcal{M}} \sum_{\ell \in \mathcal{L}_a^m} \kappa_m \, f_\ell \geq g_a, \qquad \forall \, a \in A \tag{7}$$

$$0 \leq f_\ell \leq f_\ell^{\max} \qquad \forall \, \ell \in \mathcal{L} \tag{8}$$

$$f_\ell \in \mathbb{Z}_+ \qquad \forall \, \ell \in \mathcal{L}. \tag{9}$$

DCM is a simplified version of the models appearing in Claessens, van Dijk, and Zwaneveld [5] and Bussieck, Kreuzer, and Zimmermann [3].

3 Computational Complexity

Solving DCM is NP-hard for general graphs, as the problem includes the *Set Covering Problem* as a special case ($\kappa \equiv 1, g \equiv 1, f^{\max} \equiv 1$), see also Schöbel and Scholl [11]. We now investigate how the network topology and several other factors affect the computational complexity of the model.

3.1 Fixed Costs are Hard

We first observe that fixed costs make the problem difficult. A reduction from the 0/1 Knapsack Problem can be used to prove:

Proposition 1 *DCM-FC is NP-hard, even if the underlying transportation network is a Quito graph consisting of two nodes joined by an arc, only closed lines are allowed, and there is only one transportation mode.*

3.2 Multiple Modes are Hard

It will turn out in Section 3.5 that the homogenous fleet case ($|\mathcal{M}| = 1$) allows a further simplification of the model DCM that leads to special complexity results. We therefore first discuss the case of *multiple modes* ($|\mathcal{M}| \geq 2$). Before doing this, however, let us consider an undirected version of the problem for Quito graphs.

Observe that if the line pool contains only closed lines, then each line using an arc $a = (u, v)$ must also use the arc $\overline{a} = (v, u)$, on which the bus is traveling in the opposite direction. Hence, both the arc set of the network and the arc set of each line can be partitioned into pairs of antiparallel arcs. Substituting these pairs by undirected edges, any instance of DCM with closed lines can be reduced to an equivalent *undirected* instance on an undirected graph $G = (V, E)$, where new aggregated demands on the edges are computed as follows:

$$g'_{uv} := \max\{g_{(u,v)}, g_{(v,u)}\}, \qquad \text{for all } (u, v) \in A.$$

In this version of the problem, the lines correspond to simple undirected paths in G, having the same costs. The task is to assign frequencies to these paths to cover the edge demands at minimum cost. Figure 2 gives an example of this problem transformation.
Using a reduction from the 3-Dimensional Matching Problem, one can prove:

Proposition 2 *If $|\mathcal{M}| \geq 2$, then DCM is NP-Hard even for undirected Quito graphs and if fixed costs are zero.*

3.3 Trees are Hard

Feeding line systems transport passengers from the main corridor to the neighborhoods. Each feeding line starts at a transshipment terminal, visits a set of

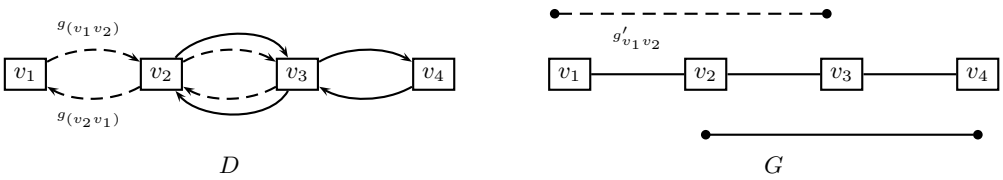


Fig. 2: Constructing the undirected version of DCM on a Quito graph. The closed lines $(v_1, v_2, v_3, v_2, v_1)$ and $(v_2, v_3, v_4, v_3, v_2)$ in D are substituted by simple undirected paths in G.

consecutive stations up to certain *turn-over station*, and returns back to the transshipment terminal stopping at the same stations on the way. Since only closed lines are admissible, there is again an undirected version of the DCM involving feeder lines. The underlying graph for this problem is a tree, with several terminals as initial nodes, and simple paths starting from it. Thus, each line is represented by an undirected path linking one terminal with a certain node where the turn-over takes place. The following result can be proved using a reduction from the 3-Dimensional Matching Problem.

Proposition 3 *DCM on trees is NP-hard, even if only closed lines and a homogeneous transportation fleet* ($|\mathcal{M}| = 1$) *is used and fixed costs are zero.*

3.4 Hopping is Hard

In this section we consider the Quito Hopping Graph topology. To this end let $D = (V, A)$ be defined by the set $V = \{v_1, v_2, \ldots, v_n\}$ of nodes representing all bus stations in the sequence along the path, and let $V_X \subseteq V$ be a subset of *express stations*. Similarly, there are express terminals, where express buses are allowed to start or end their routes.

Express lines are allowed to stop only at nodes from V_X, while *normal* (i.e., non-express) lines visit any node. Two nodes are joined by an arc if the corresponding stations can be visited consecutively by some line. Hence, the set of arcs is partitioned into three classes: a subset A_N containing arcs that may only be used by normal lines, a set A_X of arcs that may only be used by express lines, and a set A_S of "shared arcs". We assume that a transportation demand has been previously assigned to each arc of the network using some system split method. Using a reduction from 3-Dimensional Matching similar as for Proposition 2, one can prove:

Proposition 4 *DCM on Quito Hopping Graphs is NP-hard, even if only closed lines are considered and fixed costs are zero.*

3.5 Easy and Open Cases

We investigate now the Demand Covering Model on Quito graphs for a homogeneous transportation fleet ($|\mathcal{M}| = 1$) and fixed costs of zero. This model, that

we denote by *Demand Covering Model with Homogeneous Fleet* (DCM-HF), can be further simplified and formulated in the following matrix form:

$$\min \quad c^T f \tag{10}$$

subject to

$$A_H f \geq \tilde{g} \tag{11}$$

$$f \leq f^{max} \tag{12}$$

$$f \in \mathbb{Z}_+^{|\mathcal{L}|}. \tag{13}$$

Here, $\tilde{g}_a := \lceil \frac{g_a}{\kappa} \rceil$ for all $a \in A$, are the *transformed aggregated demands*, $c \in \mathbb{R}^{|\mathcal{L}|}$ is the vector of line (operating) costs, $f^{max} \in \mathbb{Z}_+^{|\mathcal{L}|}$ denotes the vector of upper bounds on the frequencies, and $A_H \in \{0,1\}^{|A| \times |\mathcal{L}|}$ is the arc-line incidence matrix.

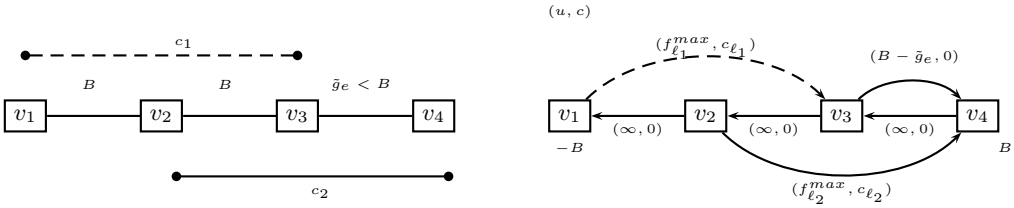


Fig. 3: Transforming undirected DCM-HF on Quito graphs to a minimum cost flow problem.

Closed Lines The undirected version of DCM-HF on Quito Graphs can be reduced to a minimum cost flow problem as follows. Let $G = (V, E)$ be an undirected Quito Graph with n nodes $v_1, \ldots, v_n$. We define $B := \max_{e \in E}\{\tilde{g}_e\}$ and define $\hat{D} = (V, \hat{A})$ to be a directed network on the node set of G, whose arc set is the disjoint union of three subsets: a set $\hat{A}_1$ containing all "backward arcs" of the form (v_i, v_{i-1}), for all $i \in \{2, 3, \ldots, n\}$; a set $\hat{A}_2$ that contains one "line arc" (v_i, v_j), with $i < j$ for every line having its ends points at v_i and v_j; and a set $\hat{A}_3$ containing one "slack arc" (v_i, v_{i+1}) for each edge $\{v_i, v_{i+1}\}$ in G with $B - \tilde{g}_e > 0$. Flow demands are defined as follows (negative demands meaning that the node is a source of flow):

$$b_{v_i} = \begin{cases} -B, \text{ if } i = 1, \\ B, \text{ if } i = n, \\ 0, \text{ otherwise.} \end{cases}$$

Arc costs are equal to zero and capacities are set to infinity on the arcs belonging to $\hat{A}_1$. For each arc in $\hat{A}_2$ representing a line $\ell \in \mathcal{L}$, the cost is equal to c_ℓ and the capacity is set to f_ℓ^{max}. Finally, each slack arc in $\hat{A}_3$ associated to an

edge e from G has capacity equal to $B - \tilde{g}_e$ and cost equal to zero. Figure 3 shows an example. Interpreting the values of a feasible flow on the line arcs as transportation capacities of the respective lines is the key to proving:

Proposition 5 *DCM-HF can be solved in polynomial time on undirected Quito Graphs.*

Open and Closed lines If both open and closed lines are present in the line pool, the symmetry of the problem is broken and the reduction of the last section does no longer work. We have not yet been able to determine the complexity of this case, but we show next that this problem is at least as difficult as the Exact Perfect Matching Problem, whose complexity is open.

The *Exact Perfect Matching Problem* (EPMP, see e.g. [13]) is a *perfect matching problem* defined on a bipartite graph with red and blue edges; there is also an integer k given. The task is to determine whether there exists a perfect matching containing exactly k blue edges. The complexity of this problem is unknown. We have proven the following proposition.

Proposition 6 *Every instance of EPMP can be transformed to an instance of DCM-HF in polynomial time.*

4 Optimizing the Trolebús System

We have carried out a computational study with various DCM models for the three network topologies considered in the previous section, based on data provided by the Trolebús System operator. The models were solved using the IP-solver SCIP [14] in its standard configuration, which was sufficient to obtain optimal solutions within a few seconds. All experiments were performed on a 3.0 GHz Pentium 4 PC with 512 MB RAM running Suse Linux 10.0.

The total fleet of the TS consists of 113 trolley-buses for the corridor and 89 normal buses for two different types of the feeding lines. The transportation network has 528 nodes, 52 of them located along the main corridor.

Table 1 reports some operational parameters for the line plan currently implemented by the TS operator in the main corridor (QG) and in the feeder line system (FLS): cost, average number of transfers per passenger, average travel times, and the accumulated frequency. We refer to this line plan as the *reference plan*. The statistics are given for time slices of one hour during the day. For the time interval 06:00–07:00, the reference plan does not provide enough capacity to cover the transportation demand with the nominal maximum capacity of a trolley bus ($\kappa = 180$); in fact, the solution requires 210 passengers to be transported by each bus unit on average, i.e., the buses are overcrowded. Passenger transfers were computed using the method described in Bouma and Oltrogge [15] (the frequency variables were fixed to the values given by the reference plan). Traveling times between stations were taken from historical data for QG and

Table 1: The current operation of the Quito Trolebús System (main corridor and feeding lines).

	Quito Graph				Feeding Lines			
T	Cost	# Tr.	Travel Time	$\sum_{\ell \in \mathcal{L}} f_\ell$	Cost	# Tr.	Travel Time	$\sum_{\ell \in \mathcal{L}} f_\ell$
06:00-07:00*	5379	–	–	57	3806.8	0.478	49.66	59
07:00-08:00	7271	0	30.7	79	4144.6	0.457	46.32	65
08:00-09:00	7246	0	28.1	83	3330.4	0.456	44.94	53
09:00-10:00	5991	0	24.3	75	3251.0	0.506	44.74	52
12:00-13:00	4858	0.0140	21.1	62	2873.6	0.452	41.16	46
13:00-14:00	4941	0.0322	21.8	63	3323.6	0.504	45.18	52
16:00-17:00	4945	0.0150	28.3	62	3473.6	0.500	46.77	54
17:00-18:00	7188	0	30.9	81	3455.8	0.415	42.89	53
18:00-19:00	7457	0	30.1	85	3050.0	0.394	43.29	48
19:00-20:00	6044	0	28.3	79	3050.2	0.548	52.47	49
20:00-21:00	5343	0	30.6	72	2597.6	0.661	56.09	41

Table 2: Optimizing the Quito Trolebús System using model DCM-HF on QG.

	Closed Lines					Closed+Open Lines				
T	Cost	# Tr.	Travel Time	$\sum_{\ell \in \mathcal{L}} f_\ell$	$\lvert L \rvert$	Cost	# Tr	Travel Time	$\sum_{\ell \in \mathcal{L}} f_\ell$	$\lvert L \rvert$
06:00-07:00	6275	0	30.02	79	19	4560.3	0	29.30	79	25
07:00-08:00	6911	0.00226	31.19	88	20	5232.7	0.00226	30.09	88	28
08:00-09:00	4792	0.00023	25.68	65	18	3785.8	0.00023	25.99	65	28
09:00-10:00	2992	0.00119	24.39	38	16	2522.2	0.00113	23.14	38	20
12:00-13:00	2230	0	20.05	26	10	2195.7	0	20.51	26	11
13:00-14:00	2342	0	21.54	28	11	2289.1	0	21.44	30	14
16:00-17:00	3234	0	26.33	39	13	2942.8	0	26.24	39	19
17:00-18:00	4847	0	29.02	58	16	4108.6	0	28.64	58	18
18:00-19:00	4625	0	27.08	58	17	3922.7	0.0116	26.79	60	20
19:00-20:00	3062	0	26.46	40	16	2667.2	0	26.50	41	17
20:00-21:00	1843	0	25.70	23	9	1711.4	0	26.10	24	10

FLS and estimated for express arcs in QHG. The transfer time for a change from line ℓ_1 to line ℓ_2 was estimated as $\frac{T}{2f_{\ell_2}}$.

As a first experiment, we carried out line planning for the main corridor based on the DCM-HF model on QG. We considered each one-hour time slice as an independent instance and ran two tests on it. In the first the line pool $\mathcal{L}$ consists of 66 closed lines and in the second one $\mathcal{L}$ contains 66 closed lines and 132 open lines. Table 2 reports the results obtained for this setting. Significant cost savings were obtained even in the case when only closed lines are allowed. The cost of our solution is smaller than that of the reference plan, with an average decrease of $ 2,119.31 per hour and a global decrease of $ 40,267. The total number of transfers increased in the morning time intervals, but decreased dramatically during midday and in the afternoon. The total number of transfers is 125, the average travel time is 25.56 minutes, compared to 26.4 minutes in the reference plan. If both open and closed lines are considered, solution costs are reduced even more. This can be explained by an asymmetry in the demand data. In fact, most passengers move in the S-N direction in the morning and return to their homes traveling in the N-S direction in the afternoon. The number of transfers is about the same as for the closed line scenario, except for time slices 15:00–16:00 and 18:00-19:00, where substantial increases are registered; the total number of transfers is 453. Nevertheless, average travel time is only 25.38 minutes.

Table 3 shows the results for the QHG instances, i.e., if express lines are considered. To this purpose, we identified 17 express stations along the main corridor. We considered a line pool with 84 closed lines and 168 open lines, of which 18 closed and 36 open lines were express lines.

In both scenarios (closed lines and closed+open lines) the cost increased compared with the results obtained for QG. The global cost for the transportation plan with only closed lines was $ 60,825, which still represents savings of 36%, when compared to the current plan. The total number of transfers increased in comparison to QG, mainly for time slices 11:00-12:00 (from 7 to 458 transfers) and 21:00-22:00 (from 0 to 288 transfers) in the scenario with open+closed lines. The increases in cost and number of transfers are, however, compensated by better service for passengers, in terms that average travel time was reduced to 23.66 minutes if only closed lines are considered and 23.35 if closed and open lines are included in $\mathcal{L}$.

Our last experiment consisted in computing a line plan for the feeder line system. The TS has three independent systems of feeder lines that intersect the main corridor at three different transshipment terminals and contain 12, 17, and 13 turn-over stations, respectively. Currently, the vehicle fleet used for serving the feeder lines is heterogeneous and contains two types of buses with transportation capacities $\kappa_1 = 90$ and $\kappa_2 = 110$. Two planning scenarios were considered, depending on the number of "branches" that a feeder line is permitted to visit. In the first scenario, feeder lines are required to visit only one branch, i.e., they are paths having the transshipment terminal as one end node. In the second scenario, up to two branches may be visited by the same line, i.e., feeder lines are paths that contain the terminal in any position. In the first scenario,

Table 3: Optimizing the Quito Trolebús System using express lines.

	Closed Lines					Closed+Open Lines				
T	Cost	# Tr.	Travel T.	$\sum_{l\in\mathcal{L}} f_l$	$\lvert L\rvert$	Cost	# Tr.	Travel T	$\sum_{l\in\mathcal{L}} f_l$	$\lvert L\rvert$
06:00-07:00	6284	0	27.42	79	24	4892.2	0.0028	25.09	80	30
07:00-08:00	7092	0	27.66	87	21	5924.0	0	26.07	94	27
08:00-09:00	5167	0.00176	22.91	65	18	4556.6	0	22.91	74	25
09:00-10:00	3207	0.00251	21.82	39	19	2898.5	0.0102	21.58	42	21
12:00-13:00	2431	0	18.75	29	12	2407.6	0	18.60	29	13
13:00-14:00	2462	0.00365	20.10	28	12	2433.2	0	20.16	29	15
16:00-17:00	3772	0	23.48	44	16	3297.9	0.0017	23.44	44	23
17:00-18:00	5255	0.00214	25.75	61	16	4429.5	0.0067	25.70	61	22
18:00-19:00	5125	0	24.25	62	20	4257.9	0.0187	24.18	62	26
19:00-20:00	3446	0	24.22	43	18	2939.5	0.0092	24.49	44	24
20:00-21:00	2083	0.00702	24.45	26	14	1899.7	0.0136	24.29	26	15

Table 4: Optimizing the Quito Trolebús System including the feeder line systems.

	One Branch						One+Two Branches						
T	Cost	# Tr.	$\sum_{l\in\mathcal{L}} f_l$	$\lvert L\rvert$	T. Time	CPU	Cost	# Tr.	$\sum_{l\in\mathcal{L}} f_l$	$\lvert L\rvert$	T. Time	CPU	Gap
06:00-07:00	3142.4	0.501	59	44	53.08	0.01	2562.4	0.496	30	28	56.03	10000	6.96
07:00-08:00	3434.0	0.454	65	43	49.23	0.04	2794.0	0.454	33	32	54.31	10000	7.03
08:00-09:00	2740.8	0.481	53	42	48.60	0.02	2220.8	0.449	27	26	51.24	10000	6.21
09:00-10:00	2698.8	0.501	52	39	49.04	0.01	2198.8	0.499	27	24	51.76	0.23	3.25
12:00-13:00	2341.2	0.444	46	37	44.78	0.03	1881.2	0.425	23	22	47.80	0.66	4.68
13:00-14:00	2707.6	0.496	52	35	46.81	0.01	2207.6	0.494	27	24	49.80	10000	8.29
16:00-17:00	2804.6	0.496	53	37	48.88	0.01	2289.0	0.473	27	24	51.40	1.54	4.75
17:00-18:00	2837.8	0.409	54	41	46.20	0.01	2309.0	0.405	28	28	49.29	10000	7.42
18:00-19:00	2464.6	0.386	47	39	45.83	0.01	2002.4	0.383	24	24	48.37	1.38	4.33
19:00-20:00	2579.4	0.531	49	38	55.79	0.02	2110.6	0.521	26	24	58.02	1.38	4.33
20:00-21:00	2279.0	0.631	43	35	63.84	0.04	1872.2	0.622	22	22	68.34	0.23	3.01
Average	2443.6	0.549	46.2	36.1	55.42	0.020	1997.5	0.532	23.8	22.8	58.43	3692.0	4.99

a total of 84 lines were considered in the line pool (containing all three feeder systems), while in the second scenario 470 new lines were added. In our runs, we allowed an optimality gap of 5% and set a time limit of 10000 for each instance.

Table 4 reports the results (aggregated for all three feeder systems). As expected, the average number of transfers is much larger than in the previous experiments, since trips of the form "feeding line-main corridor-feeding line", which involve at least two transfers, are common in the solution. In both the "one branch" and "two branches" scenarios, the cost was reduced in comparison to the currently implemented solution by about 18% (one branch) and 32% (two branches). On the other hand, these savings are related to larger travel times for the passengers, which are slightly increased in all instances.

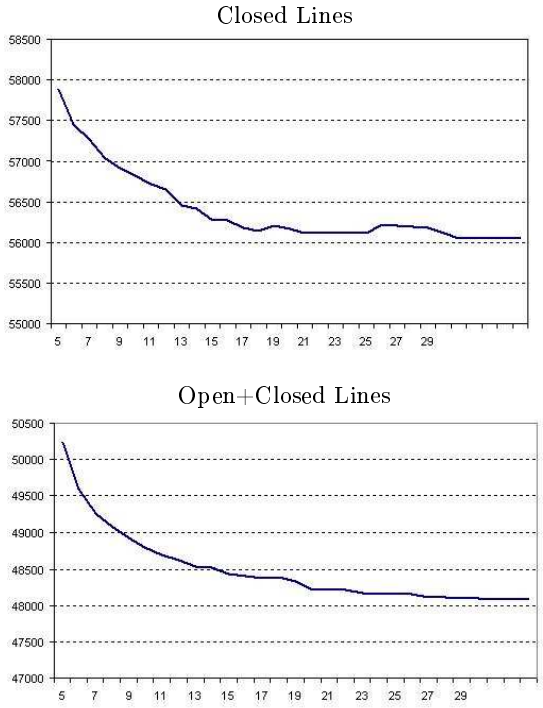


Fig. 4: Tradeoff cost vs. maximum number of lines.

The dramatic cost decrease in our solutions over the reference solution can be explained by two factors. First, our DCM model does not impose a limit on the number of lines in a solution. In practice, however, it is not desirable to have too many lines, as the whole system becomes too complicated for the user and the operator. Adding new binary variables to DCM that indicate whether a line is chosen in the solution or not, we carried out new experiments for the QG network topology limiting the allowed numbers of lines to a maximum between five (the number of lines currently used by the TS operator) and 30. Figure 4

summarizes the results for the whole day. As expected, the optimum solution cost increases as the number of allowed lines decrease, but the increase is less than 10% from 30 to 5 lines. A second reason can be found in the planning policies that the TS operator is currently using. Up to now, line planning has been carried out in a single step together with duty scheduling for the bus drivers by pre-assigning bus drivers to buses. It might be that this scheme is just too inflexible, since hard laboral constraints might discard some good solutions for the line planning problem. It would certainly be worthwhile to compute a vehicle and a duty schedule based on our line plans, in order to get a better assessment of the operational consequences of such an optimization.

References

1. Odoni, A.R., Rousseau, J.M., Wilson, N.H.M.: Models in urban and air transportation. In S. M. Pollock et al., ed.: Handbooks in OR & MS 6. North Holland (1994) 107–150
2. Bussieck, M.R., Winter, T., Zimmermann, U.T.: Discrete optimization in public rail transport. Math. Program. **79**(1–3) (1997) 415–444
3. Bussieck, M.R., Kreuzer, P., Zimmermann, U.T.: Optimal lines for railway systems. Eur. J. Oper. Res. **96**(1) (1997) 54–63
4. Bussieck, M.R.: Optimal lines in public rail transport. PhD thesis, TU Braunschweig (1997)
5. Claessens, M.T., van Dijk, N.M., Zwaneveld, P.J.: Cost optimal allocation of rail passanger lines. Eur. J. Oper. Res. **110**(3) (1998) 474–489
6. Bussieck, M.R., Lindner, T., Lübbecke, M.E.: A fast algorithm for near optimal line plans. Math. Methods Oper. Res. **59**(2) (2004)
7. Goossens, J.W.H.M., van Hoesel, S., Kroon, L.G.: On solving multi-type line planning problems. METEOR Research Memorandum RM/02/009, University of Maastricht (2002)
8. Goossens, J.W.H.M., van Hoesel, S., Kroon, L.G.: A branch-and-cut approach for solving railway line-planning problems. Transportation Sci. **38**(3) (2004) 379–393
9. Borndörfer, R., Grötschel, M., Pfetsch, M.E.: A column-generation approach to line planning in public transport. Transportation Sci. **41**(1) (2007) 123–132
10. Borndörfer, R., Grötschel, M., Pfetsch, M.E.: Models for line planning in public transport. In Hickman, M., Mirchandani, P., Voß, S., eds.: Computer-aided Systems in Public Transport. Volume 600 of Lecture Notes in Economics and Mathematical Systems., Springer-Verlag (2008) 363–378
11. Schöbel, A., Scholl, S.: Line planning with minimal travelling time. Technical Report 1-2005, University of Göttingen, Germany (2005)
12. Scholl, S.: Customer-Oriented Line Planning. PhD thesis, University of Göttingen (2005)
13. Papadimitriou, C.H., Yannakakis, M.: The complexity of restricted spanning tree problems. J. ACM **29**(2) (1982) 285–309
14. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin (2007)
15. Bouma, A., Oltrogge, C.: Linienplanung und Simulation für öffentliche Verkehrswege in Praxis und Theorie. Eisenbahntechnische Rundschau **43**(6) (1994) 369–378

Recoverable Robustness for Railway Rolling Stock Planning

Valentina Cacchiani[1], Alberto Caprara[1], Laura Galli[1], Leo Kroon[2,3], Gábor Maróti[3], and Paolo Toth[1]

[1] D.E.I.S., University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy. {alberto.caprara,valentina.cacchiani,l.galli,paolo.toth}@unibo.it
[2] Netherlands Railways, Utrecht, The Netherlands.
[3] Rotterdam School of Management, Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR, Rotterdam, The Netherlands. {gmaroti,lkroon}@rsm.nl.

Abstract. In this paper we explore the possibility of applying the notions of Recoverable Robustness and Price of Recoverability (introduced by [5]) to railway rolling stock planning, being interested in recoverability measures that can be computed in practice, thereby evaluating the robustness of rolling stock schedules. In order to lower bound the Price of Recoverability for any set of recovery algorithms, we consider an "optimal" recovery algorithm and propose a Benders decomposition approach to assess the Price of Recoverability for this "optimal" algorithm. We evaluate the approach on real-life rolling stock planning problems of NS, the main operator of passenger trains in the Netherlands. The preliminary results show that, thanks to Benders decomposition, our lower bound can be computed within relatively short time for our case study.

1 Introduction

Recently [5] introduced the concept of Recoverable Robustness as a generic framework for modelling robustness issues in railway scheduling problems. Also, the Price of Recoverability (PoR) was defined as a measure of recoverability. However, this notion is mainly of theoretical nature, and cannot be used in a straightforward way for concrete problems. In particular, computing PoR requires, according to [5], minimisation over a set of recovery algorithms, and it is not clear how this can be carried out.

Reference [3] considers the robustness of shunting problems. It overcomes these difficulties by analysing PoR for a few *concrete* recovery algorithms and by proving lower and upper bounds.

The purpose of this paper is to investigate another way of bringing the theoretical notion of PoR closer to practice. In particular, we consider what happens if recovery is done in the best possible way: the resulting value of PoR for this unique "optimal" recovery algorithm is then a lower bound on the value of PoR for *any* set of recovery algorithms. Under mild assumptions, this lower bound can be computed by solving a single mathematical program.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1590

We address the practical evaluation of the lower bound above for a specific case study, concerning the medium-term rolling stock planning problem of NS, the main operator of passenger trains in the Netherlands. It arises 2–6 months before the actual train operations, and amounts to assigning the available rolling stock to the trips in a given timetable. The objectives of the problem that is traditionally solved, call nominal problem, are related to service quality, efficiency, and – to a limited extent – to robustness. Reference [4] describes a Mixed Integer Linear Programming (MILP) model for this nominal problem. Using commercial MILP software, the solution times on real-life problems of NS are quite low, ranging from a few minutes (on most instances) to a couple of hours (on some particularly complex instances). A software tool based on this model has been in operation within NS since 2004.

The solutions of the nominal problem are optimal under undisrupted circumstances only. However, infrastructure failures, bad weather, and engine breakdowns often lead to disruptions where the nominal solution cannot be carried out any more. In such cases, disruption management must take place to come up with an adjusted rolling stock schedule. In this re-scheduling process, the original objective criteria are of marginal importance, the goal being to quickly find a feasible solution that is "close" to the nominal one and can be implemented in practice.

The computation of the lower bound on PoR mentioned above for our case study requires the solution of a very-large Linear Programming (LP) model, in which several possible disruption scenarios are considered. We propose a Benders decomposition approach for the solution of this LP, leading to a subproblem for each scenario. Our preliminary computational results on a real-life rolling stock planning instance of NS indicate that the method takes relatively short time, and widely outperforms the straightforward solution of the whole LP by a state-of-the-art solver.

This paper is structured as follows. In Section 2 we quote the definition of the Recoverable Robustness and Price of Recoverability from [5]. In Section 3 we describe the lower bound that we consider, based on a "best possible recovery" policy, and the associated mathematical programming problem. Section 4 describes the railway rolling stock scheduling problem of NS. Section 5 is devoted to our preliminary computational results. Finally, Section 6 outlines our plans for further research.

2 The Price of Recoverability

In this section we give a short summary of the definition of the Price of Recoverability by [5]. The main idea is to compute a solution to an optimisation problem and at the same time to analyse the recovery costs in case of disturbed input data. More concretely, one considers a (limited) set of scenarios with their own feasible regions, as well as a set of admissible recovery algorithms. The objective is to find a solution of the original (nominal) problem *and* a recovery algorithm in the given set. The requirement is that, using the recovery algorithm, the solution

of the original optimisation problem can be transformed to a feasible solution of each scenario at "low cost". The Price of Recoverability measures both the objective function of the original problem and the recovery costs.

The set of admissible recovery algorithms can be chosen in several ways. One may consider algorithms with limited (e.g. linear) running time, or algorithms that are obtained from a particular heuristic framework (e.g. a crew re-scheduling algorithm based on iterated crew duty swaps).

The notions of Recoverable Robustness and Price of Recoverability are defined formally as follows. First of all, we are given a *Nominal Problem* of the form:

$$\mathrm{NP} = \min\{c(x) \mid x \in K\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is the variable vector, $K \subseteq \mathbb{R}^n$ is the feasible region and $c : K \to \mathbb{R}_+$ is the cost function.

Moreover, we are given a set $\mathcal{S}$ of *scenarios*; each scenario $s \in \mathcal{S}$ having its own feasible region K_s. (For example, a scenario may refer to the case of cancelling some trains due to infrastructure failure, thereby requiring some kind of recovery action.) Furthermore, we are given a set $\mathcal{A}$ of *recovery algorithms*: a recovery algorithm $A \in \mathcal{A}$ takes on input a nominal solution $x \in K$ and a scenario $s \in \mathcal{S}$ and produces a solution $A(x, s) \in K_s$ which is feasible in scenario s. Finally, we are given, for $s \in \mathcal{S}$, a function $d_s : K \times K_s \to \mathbb{R}_+$ measuring the *deviation* $d_s(x, x_s)$ of a solution x_s for scenario s from a nominal solution x, and a monotone non-decreasing function $f : \mathbb{R}_+^{\mathcal{S}} \to \mathbb{R}_+$ penalising the deviation over all scenarios.

The *Recovery-Robust Optimisation Problem* defined in [5] is then:

$$\mathrm{RPOP}_{\mathcal{A}} = \min\{c(x) + f(z) \mid x \in K,\ A \in \mathcal{A},\ z_s = d_s(x, A(x, s))\ \ (s \in \mathcal{S})\}, \tag{2}$$

where $z = (z_{s_1}, z_{s_2}, \ldots) \in \mathbb{R}_+^{\mathcal{S}}$ is a vector of auxiliary variables representing the deviations.

Reference [5] chooses $f(z) = \max_{s \in \mathcal{S}} z_s$, i.e. penalises the *maximum deviation* in the objective function. A stochastic-programming approach would be to define a probability p_s for each scenario $s \in \mathcal{S}$, and to consider $f(z) = \sum_{s \in \mathcal{S}} p_s z_s$, penalising the *expected (average) deviation*.

The Price of Recoverability (PoR) is then defined as the ratio between the optimal values of the Recovery-Robust and the Nominal Problem:

$$\mathrm{PoR}_{\mathcal{A}} = \frac{\mathrm{RPOP}_{\mathcal{A}}}{\mathrm{NP}}. \tag{3}$$

Reference [5] also compares Recoverable Robustness to well-known concepts such as stochastic programming (see e.g. [2]) or robust optimisation ([1]), and discusses the similarities and differences of the approaches to capture robustness.

2.1 Reformulation of PoR

Define the function

$$\Phi_A(x) = c(x) + f\left(d_{s_1}(x, A(x, s_1)), d_{s_2}(x, A(x, s_2)), \dots\right).$$

Then $\mathrm{RPOP}_{\mathcal{A}}$ corresponds to the minimisation of the function Φ:

$$\mathrm{RPOP}_{\mathcal{A}} = \min_{A \in \mathcal{A}} \min_{x \in K} \Phi_A(x). \tag{4}$$

In later sections of this paper we shall consider a simplified version for the case in which $\mathcal{A}$ contains a single algorithm A only:

$$\mathrm{RPOP}_{\{A\}} = \min_{x \in K} \Phi_A(x). \tag{5}$$

2.2 How to Compute PoR?

The definition (3) (via the definition (2)) requires minimisation over the set $\mathcal{A}$ of recovery algorithms. How (and if) this can be done clearly depends on how the set $\mathcal{A}$ is specified. In any case, one can follow (at least) two approaches to compute (or approximate) PoR.

In the first approach, one considers a class of small and well-behaved problems together with a small set of recovery algorithms. Then one proves worst-case bounds on PoR by an appropriate theoretical analysis. Reference [3] reports such results for the shunting problem. With our notation, such an approach essentially amounts to deriving bounds on the minimum of the function Φ_A for each $A \in \mathcal{A}$. Note that this approach is likely to succeed on fairly simplified test problems; real-life (railway) scheduling problems often have features that cannot be handled easily in theoretical worst-case proofs.

In this paper we follow a second approach, namely we restrict attention to the best possible recovery algorithm, observe that the computation of PoR for this single algorithm leads to a lower bound on PoR for each set $\mathcal{A}$, and numerically solve a mathematical programming problem to compute the value of this lower bound for a particular real-life railway resource scheduling problem. Therefore the results that we obtain are of empirical nature.

3 PoR with an Optimal Recovery Algorithm

Let $\mathcal{A}_{\mathrm{all}}$ be the set of *all* recovery algorithms and define

$$A_{\mathrm{opt}}(x, s) = \arg\min\{d_s(x, x_s) \mid x_s \in K_s\}. \tag{6}$$

In words, for each scenario s and for each $x \in K$, A_{opt} determines the solution in K_s with the smallest possible deviation from x. That is, A_{opt} represents the best possible recovery action. This is formalised in the following proposition.

Proposition 1 $\mathrm{RPOP}_{\mathcal{A}_{\mathrm{all}}} = \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$.

Proof. Clearly, $\mathrm{RPOP}_{\mathcal{A}_{\mathrm{all}}} \leq \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$. On the other hand, for each $x \in K$, $A \in \mathcal{A}$ and $s \in \mathcal{S}$ we have

$$d_s(x, A(x,s)) \geq d_s(x, A_{\mathrm{opt}}(x,s))\,.$$

Therefore

$$\min_{A \in \mathcal{A}_{\mathrm{all}}} \Phi_A(x) \geq \Phi_{A_{\mathrm{opt}}}(x)$$

and (4) yields

$$\mathrm{RPOP}_{\mathcal{A}_{\mathrm{all}}} \geq \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}.$$

In other words, the minimum of (2) if $\mathcal{A} = \mathcal{A}_{\mathrm{all}}$ is attained at A_{opt}. This of course implies that the minimum of (2) for a generic $\mathcal{A}$ cannot be better than $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$, as stated in the following corollary.

Corollary 2 $\mathrm{RPOP}_{\mathcal{A}} \geq \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ *for every set* $\mathcal{A}$ *of algorithms.*

This implies that the computation of $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ yields a lower bound on $\mathrm{RPOP}_{\mathcal{A}}$, and therefore $\mathrm{PoR}_{A_{\mathrm{opt}}}$ a lower bound on $\mathrm{PoR}_{\mathcal{A}}$, for a generic set of recovery algorithms $\mathcal{A}$. Moreover,

$$\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}} = \min\{c(x) + f(z) \mid x \in K,\ x_s \in K_s\ (s \in \mathcal{S}),\ z_s = d_s(x, x_s)\ \ (s \in \mathcal{S})\}. \tag{7}$$

That is, $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ is the optimum value of a mathematical program, which is not the case for $\mathrm{RPOP}_{\mathcal{A}}$ for a generic $\mathcal{A}$. This is the reason why in this paper we focus our attention on the practical computation of the former.

3.1 Solution Methodology

For the sake of concreteness, we will restrict our attention to the case of $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ in which the following hold:

- $f(z) = \max_{s \in \mathcal{S}} z_s$, i.e. only the largest deviation is penalised in the objective function;
- $c(x) = c^{\mathsf{T}} x$ for a given $c \in \mathbb{R}^n$, i.e. the objective function is linear;
- $x \in K$ can be expressed as $Ax \geq b$ for given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, i.e. feasibility of a nominal solution can be expressed by linear constraints (and possibly by the integrality of some components of x, see below);
- for each $S \in \mathcal{S}$, $x_s \in K_s$ can be expressed as $A_s x_s \geq b_s$ for given $A_s \in \mathbb{R}^{m_s \times n}$ and $b_s \in \mathbb{R}^{m_s}$;
- for each $S \in \mathcal{S}$, $z_s = d_s(x, x_s)$ can be expressed as $z_s = d_s^{\mathsf{T}} x + e_s^{\mathsf{T}} x_s + g_s$ for given $d_s, e_s \in \mathbb{R}^n$ and $g_s \in \mathbb{R}$, i.e. the deviation is a linear function of the nominal and the recovered solution.

If we include the possible integrality restriction on some of the x and x_s components, the above assumptions are not really restrictive, since they amount to require that the nominal problem, the feasibility of a recovered solution, and

the value of the deviation can be expressed as a MILP. In the computational experiments carried over in this paper, we will restrict attention to the case in which such integrality restriction is not imposed. Depending on the specific application, this may be the case, or it may lead to solution of the LP relaxation of the actual MILP, which yields a lower bound on $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ and therefore on $\mathrm{RPOP}_{\mathcal{A}}$ for each $\mathcal{A}$. In any case, the Benders decomposition approach that we illustrate can easily be modified to have integrality restrictions on the x variables. Since the purpose of this paper is to study the possibility to practically compute lower bounds on PoR for real-world instances, it is natural to restrict attention to LP relaxations.

Given the above assumptions, (7) can be formulated as follows, where λ is an auxiliary variable expressing the deviation penalty:

$$\begin{aligned}
\min \quad & c^{\mathsf{T}}x & & + \lambda & & & (8)\\
\text{s.t.} \quad & Ax & & & \geq b, & & (9)\\
& & A_s x_s & & \geq b_s, & \forall s \in \mathcal{S}, & (10)\\
& - d_s^{\mathsf{T}}x - & e_s^{\mathsf{T}}x_s & + \lambda & \geq g_s, & \forall s \in \mathcal{S}. & (11)
\end{aligned}$$

For solving (8) – (11) one can apply various mathematical programming techniques. In this paper we focus on Benders decomposition (also known as L-shaped method) (see e.g. [7]), a cutting plane method that exploits the block-diagonal structure of the problem. This is an approach widely used for such problems (such as for stochastic programming).

Briefly, the Benders decomposition approach keeps solving the (gradually extended) nominal problem (8) – (9). Based on the current optimal solution, the feasibility of the subproblem (10) – (11) is checked. The procedure terminates if the subproblem is feasible, in which case the current optimal solution is optimal also for (8) – (11). In case of infeasibility, inequalities in terms of x and λ are derived and added to the nominal problem, and the updated nominal problem is re-optimised.

Benders decomposition is applicable if the subproblems are LPs, i.e. if the x_s variables are continuous, whereas integrality on the x variables can be handled, although (as already mentioned) it will be relaxed in our computational experiments.

4 The Test Problem: Rolling Stock Re-scheduling

This section is devoted to the description of the specific real-world case study on which we focused our attention.

4.1 The Nominal Problem

We consider the medium-term railway rolling stock scheduling problem of NS. It arises 2–6 months before the actual railway operations, and has the task of

assigning the available rolling stock to the *trips* in a given timetable. In this section we give a brief problem description. Further details about the problem can be found in [4] and in [6].

The rolling stock consists of *units*. Each unit has driver's seats at both ends and an own engine. It is composed of a number of carriages, and cannot be split up in every-day operations. Units are available in different *types* and can be combined with each other to form *compositions*. This allows a fine adjustment of the seat capacity to the passenger demand.

The timetable of NS is quite dense, and the turning time of the trains is short, often less then 20 minutes. The rolling stock connections are explicitly given in the input timetable by the *successor trips*: The units that serve in a trip go over to the successor trip, even though certain *composition changes* can take place. Due to the short turning times, the composition change possibilities are limited to *coupling* or *uncoupling* of one or two units at the appropriate side of the train.

The objective is three-fold. *Service quality* is measured by seat shortage kilometres. It is computed by comparing the assigned seat capacity to the a priori given expected number of passengers; by multiplying the number of unseated passengers by the length of the trip; and finally by summing these values over all trips. *Efficiency* is expressed by the carriage-kilometres which is roughly proportional both to the electricity or fuel consumption and to the maintenance costs. *Robustness* is taken into account by counting the number of composition changes. Indeed, coupling or uncoupling of units causes additional traffic through the railway nodes, and thereby may lead to delay propagation if some passing trains are late.

We note again that the the nominal problem is solved several months before the operations. This leaves enough time to plan the low-level train operations at the railway nodes. In particular, shunting drivers are scheduled to carry out the coupling and uncoupling operations. Moreover, the end-of-day rolling stock balances are such that the units are at the right place for the next day's operations.

In order to define a MILP for the problem, the set of rolling stock types is denoted by M, the set of trips by T, the set of compositions by P, and the set of stations by S. For any $m \in M$, a_m denotes the number of available rolling stock units of type m.

The main binary decision variables are $x_{t,p}$, expressing whether composition p is assigned to trip t. Moreover, we have the binary variables $z_{t,p,p'}$ whose value is 1 if trip t has composition p and if the successor of t has composition p'. The z variables are only defined for those triples (t, p, p') where the composition change from p to p' is allowed after trip t, i.e. the constraints on the composition changes are implicitly represented by these variables.

The stations are modelled by the *inventories*. The inventory of a station at a certain time instant consist of all units that are located there. The basic rule is that units to be coupled to a train are pulled from the inventory immediately upon departure, while uncoupled units are added to the inventory a certain time

(say 30 minutes) after arrival. This ensures enough time for necessary shunting operations.

The integer variables $y_{t,m}$ count the inventories of the units of type m at the departure station of trip t right after the departure of trip t. The beginning-of-day and end-of-day inventories of station s of type m are represented by the variables $y^0_{s,m}$ and $y^\infty_{s,m}$.

Letting the successor of trip t be denoted by $\sigma(t)$, the departure station of t be denoted by $d(t)$, c, d be appropriate objective function coefficients, and α, β, γ be appropriate inventory coefficients, a MILP formulation is the following.

$$\min \sum_{t\in T}\sum_{p\in P} c_{t,p}x_{t,p} + \sum_{t\in T}\sum_{p\in P}\sum_{p'\in P} d_{t,p,p'}z_{t,p,p'} \tag{12}$$

$$\text{s.t.} \sum_{p\in P} x_{t,p} = 1, \qquad \forall t \in T, \tag{13}$$

$$x_{t,p} = \sum_{p'\in P} z_{t,p,p'}, \qquad \forall t \in T, p \in P, \tag{14}$$

$$x_{\sigma(t),p'} = \sum_{p\in P} z_{t,p,p'}, \qquad \forall t \in T, p' \in P, \tag{15}$$

$$\begin{aligned} y_{t,m} = y^0_{d(t),m} &+ \sum_{t'\in T}\sum_{p\in P}\sum_{p'\in P} \alpha_{t,t',p,p',m} z_{t',p,p'} \\ &+ \sum_{t'\in T}\sum_{p\in P} \beta_{t,t',p,m} x_{t',p}, \qquad \forall t \in T, m \in M, \end{aligned} \tag{16}$$

$$y^\infty_{t,m} = y^0_{d(t),m} + \sum_{t'\in T}\sum_{p\in P}\sum_{p'\in P} \gamma_{t,t',p,m} x_{t',p}, \quad \forall t \in T, m \in M, \tag{17}$$

$$\sum_{s\in S} y^0_{s,m} = a_m, \qquad \forall m \in M, \tag{18}$$

$$x_{t,p}, z_{t,p,p'} \text{ binary}, \qquad \forall t \in T, p \in P, p' \in P, \tag{19}$$

$$y_{t,m}, y^0_{s,m}, y^\infty_{s,m} \geq 0, \text{ integer}, \qquad \forall t \in T, s \in S, m \in M. \tag{20}$$

The objective (12) takes into account the trip assignments and the compositions of consecutive trips. Constraints (13) state that each trip gets exactly one composition. Constraints (14) and (15) link the z variables to the x variables. Constraints (16) and (17) compute the inventories with appropriate coefficients α, β and γ. Constraints (18) specify the available rolling stock.

The objective function can incorporate a wide variety of objective criteria related to service quality, efficiency and robustness. Experience shows that, for the practically meaningful objective coefficients, the LP relaxation of the model above is very tight, the associated lower bound being always within a few percents of the MILP optimum. The MILP model can be solved for medium-sized instances of NS within a few seconds to optimality. Simple LP rounding heuristics turned out to be powerful for the most challenging problem instances.

4.2 The Scenarios and the Associated Deviations

In our robustness framework, the solutions of the nominal problem are to be operated subject to disruption scenarios. Each scenario is obtained by assuming that a certain part of the network is blocked for a certain time interval of several hours. All the trips that interfere with the infrastructure blockage are removed. Such disruptions are quite common in practice. These are the ones that require significant resource re-scheduling.

It is worthwhile to note that the timetabling and resource scheduling decisions are strictly separated. In the Netherlands, for example, an independent infrastructure managing authority is responsible for the timetable adjustments, while the railway operators themselves are responsible for resource re-scheduling. Therefore from the resource planning's point of view, the adjusted timetable that takes care of the disruption is to be considered as input.

We assume that a disruption becomes known at the beginning of the blockage. The task is then to re-schedule the rolling stock from that point on till the end of the day. The solution has to fulfil the same requirements as the nominal problem, the only additional option being to cancel a trip.

In this research we also assume that the exact duration of the disruption is known at its beginning. Admittedly, this assumption is very optimistic for practical purposes. On the other hand, it simplifies the mathematical model, and still enables one to gain insight of the recovery capacity of rolling stock schedules.

The three main criteria in re-scheduling are as follows (in decreasing order of importance): (*i*) minimise the number of cancelled trips; (*ii*) minimise the number of newly introduced couplings and uncouplings; (*iii*) minimise the deviation of the planned end-of-day rolling stock balance. The first criterion limits the passenger inconvenience. The second criterion aims at keeping the schedule of the shunting drivers intact. The third criterion tries to restrict the consequences of the disruption on a single day.

Although the model (12) – (20) was originally developed for the nominal problem, it can be adjusted for rescheduling as well. That is, the feasibility of a recovered solution and the associated recovery costs can be computed as a variant of the model above. We express the model for a single scenario, omitting the index s that represents the scenario and noting that here s stands for the index of a station.

First of all, constraints (13) – (20) with variables $\widetilde{x}$, $\widetilde{z}$, $\widetilde{y}^0$ and $\widetilde{y}^\infty$ are to be stated for the trips of each scenario. In this case, as anticipated, we also allow the empty composition $\emptyset$, where $\widetilde{x}_{t,\emptyset} = 1$ means that trip t is cancelled. Then, one has to impose constraints that the rolling stock schedule is not changed until the beginning of the disruption, adding the constraints $\widetilde{x}_{t,p} = x_{t,p}$ for each $p \in P$ and trip $t \in T$ ending before the disruption. Finally, the model is extended to

express the recovery costs:

$$\lambda \geq c_1 \sum_t \widetilde{x}_{t,\emptyset} + c_2 \sum_{t \in T} \widetilde{e}_t + \sum_{s \in S} \sum_{m \in M} \widetilde{d}_{s,m}, \tag{21}$$

$$\widetilde{d}_{s,m} \geq y^{\infty}_{s,m} - \widetilde{y}^{\infty}_{s,m}, \qquad \forall s \in S, m \in M, \tag{22}$$

$$\widetilde{d}_{s,m} \geq \widetilde{y}^{\infty}_{s,m} - y^{\infty}_{s,m}, \qquad \forall s \in S, m \in M, \tag{23}$$

$$w_t = \sum \left(z_{t,p,p'} \mid \ p \to p' \text{ is coupling or uncoupling} \right), \quad \forall t \in T, \tag{24}$$

$$\widetilde{w}_t = \sum \left(\widetilde{z}_{t,p,p'} \mid \ p \to p' \text{ is coupling or uncoupling} \right), \quad \forall t \in T, \tag{25}$$

$$\widetilde{e}_t \geq \widetilde{w}_t - w_t, \qquad \forall t \in T, \tag{26}$$

$$\widetilde{d}_{s,m} \geq 0, \qquad \forall s \in S, m \in M, \tag{27}$$

$$\widetilde{e}_t \geq 0, \qquad \forall t \in T. \tag{28}$$

The auxiliary variables $\widetilde{d}$ measure the deviation of the planned end-of-day rolling stock inventories (i.e. that of the nominal solution) from the realised end-of-day rolling stock inventories (i.e. those in the scenario). The value of the auxiliary variable w_t is 0 or 1 depending on whether the nominal solution has a composition change (i.e. coupling or uncoupling of units) after trip t. Similar role is played by $\widetilde{w}_t$ in the scenario. The auxiliary variable $\widetilde{e}_t$ has a value at least 1 if a new shunting is introduced after trip t, i.e. if there was no composition change after trip t in the nominal solution whereas there is one in the recovered solution. The objective penalises the variables $\widetilde{d}$ and $\widetilde{e}$ as well as all variables $\widetilde{x}$ that assign an empty composition to a trip.

5 Computational Results

We implemented the robust scheduling problem (8) – (11) with the rolling stock (re-)scheduling model described in Section 4 for the so called 3000 line of NS. This is an Inter-City line with a closed rolling stock circulation. The instance contains about 400 trips connecting 8 stations, and is served by two rolling stock types with 11 and 24 units, respectively. The 3000 line is one of the medium-sized rolling stock instances of NS.

The nominal problem is based on the actual timetable of NS. The scenarios have been generated artificially using a program of [8], which simulates the decisions of the infrastructure manager about train cancellations, including the successors of the trips after disruption. In that respect, the input data of the scenarios follow the same rules and assumptions as the nominal problem.

As already discussed, the goal of our preliminary computational tests is to investigate whether the suggested optimisation framework can be used at all to assess PoR for our rolling stock scheduling problem. Therefore we restricted ourselves to the solution of LP relaxations.

We implemented two solution methods: (*i*) solving (8) – (11) directly as a single LP; (*ii*) applying a canonical Benders decomposition Approach. Our

computer codes are written in C and run on a personal computer, solving the LPs by ILOG CPLEX 10.0. The master problem has about 14,500 variables, 8,600 constraints and 310,000 non-zeros in the matrix.

The solution approaches have been tested with 2–20 scenarios, implying that the LPs solved by method (i) feature 43,000–305,000 variables, 25,000–180,000 constraints and 950,000–6,500,000 non-zeros.

For each number of scenarios, we solved two variants of the problem: Test-I and Test-II. They share the same constraint matrix but differ in the objective function. The cost coefficients are given in Table 1. Test-I focuses on service quality (by penalising seat shortages more heavily) while Test-II emphasises efficiency (by penalising carriage kilometres more heavily).

Table 1. Coefficients for the nominal objective function as well as for the recovery costs.

Criterion in nominal problem	Test-I	Test-II
seat shortage km	100	50
carriage km	9	100
composition change	5	10
Criterion for recovery	Test-I	Test-II
cancellation	1,000,000	1,000,000
inventory deviation	20,000	20,000
new shunting	10,000	10,000

The computational results with the two solution approaches are summarised in Table 2. It turns out that the huge LPs in method (i) are barely solvable. For more than 6 or 7 scenarios, the solution time exceeds our time limit of 1800 seconds. The cases with 10 or more scenarios appear to be far from being solved after several hours of CPU time. The Benders decomposition approach, on the other hand, is able to cope with the problems. After applying 24–640 and 30–360 Benders cuts for Test-I and Test-II, respectively, optimality was reached within the time limit.

The above results prove that, at least for our case study, the general lower bound on PoR that we propose can be computed within reasonable time.

6 Summary and Future Research

In this paper we summarised our understanding of the Price of Recoverability. In addition, we proposed a mathematical programming approach to compute a lower bound on the Price of Recoverability for real-life railway scheduling problems.

A Benders decomposition approach has been implemented for a medium-sized rolling stock scheduling problem of NS. The preliminary computational results indicate the problem is widely tractable with up to 20 disruption scenarios. Note

Table 2. The number of applied Benders cuts as well as the running times in seconds for the Benders decomposition approach and for the direct solution of the whole LP (referred to as 'CPLEX') on Test-I and Test-II. A dash indicates the running time of CPLEX exceeding 1800 seconds.

	Test-I			Test-II		
	Benders		CPLEX	Benders		CPLEX
# scens.	# cuts	CPU time	CPU time	# cuts	CPU time	CPU time
2	24	22	122	32	30	75
3	27	27	366	24	25	372
4	140	224	795	100	141	880
5	175	264	1,055	125	175	1,078
6	168	277	1,962	150	209	—
7	210	319	—	140	205	—
8	248	454	—	152	223	—
9	288	603	—	171	278	—
10	320	688	—	190	332	—
11	352	734	—	209	364	—
12	384	798	—	228	400	—
13	325	662	—	234	439	—
14	420	1,014	—	252	499	—
15	450	1,090	—	300	653	—
16	480	1,163	—	320	698	—
17	595	1,710	—	306	642	—
18	540	1,380	—	324	701	—
19	570	1,459	—	342	730	—
20	640	1,840	—	360	816	—

that the whole problem (a single LP) cannot be solved within several hours even with just 10 scenarios.

In order to improve the proposed method, we will go on with more thorough computational tests. First, the preliminary computations concern the LP relaxation of the rolling stock scheduling problem; we are going to study the original MILP models as well. Second, the current way of selecting the Benders cuts is very simple; we are going to evaluate the effect of more sophisticated cut selection methods. Third, Benders decomposition is not the only possible approach for solving (8) – (11). In our future research we are going to explore other mathematical programming techniques, such as convex optimisation (e.g. through the subgradient algorithm). Last but not least, we are going to investigate implications of the Price of Recoverability to railway practice.

Acknowledgment

This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

References

1. D. Bertsimas and M. Sim. The Price of Robustness. *Operations Research*, 52(1): 35–53, 2004.
2. J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.
3. S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In *Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*, 2007.
4. P. Fioole, L. Kroon, G. Maróti, and A. Schrijver. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174:1281–1297, 2006.
5. C. Liebchen, R. Möhring, M. Lübbecke, and S. Stiller. Recoverable robustness. Technical Report ARRIVAL-TR-0066, ARRIVAL Project, 2007.
6. G. Maróti. *Operations Research Models for Railway Rolling Stock Planning*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.
7. G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.
8. L. Nielsen. Disruption Generator for Railway Rolling Stock Re-scheduling, 2008. Software.

Robust Line Planning under Unknown Incentives and Elasticity of Frequencies *

Spyros Kontogiannis[1,2] and Christos Zaroliagis[1,3]

[1] R.A. Computer Technology Institute, N. Kazantzaki Str., Patras University Campus, 26500 Patras, Greece
[2] Computer Science Department, University of Ioannina, Ioannina, Greece
[3] Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece
Email: kontog@cs.uoi.gr, zaro@ceid.upatras.gr

Abstract. The problem of *robust line planning* requests for a set of origin-destination paths (lines) along with their traffic rates (frequencies) in an underlying railway network infrastructure, which are robust to fluctuations of real-time parameters of the solution.
In this work, we investigate a variant of robust line planning stemming from recent regulations in the railway sector that introduce competition and free railway markets, and set up a new application scenario: there is a (potentially large) number of *line operators* that have their lines fixed and operate as competing entities struggling to exploit the underlying network infrastructure via frequency requests, while the management of the infrastructure itself remains the responsibility of a single (typically governmental) entity, the *network operator*.
The line operators are typically unwilling to reveal their true incentives. Nevertheless, the network operator would like to ensure a fair (or, socially optimal) usage of the infrastructure, e.g., by maximizing the (unknown to him) aggregate incentives of the line operators. We show that this can be accomplished in certain situations via a (possibly anonymous) incentive-compatible pricing scheme for the usage of the shared resources, that is *robust* against the unknown incentives and the changes in the demands of the entities. This brings up a new notion of robustness, which we call *incentive-compatible robustness*, that considers as robustness of the system its tolerance to the entities' unknown incentives and elasticity of demands, aiming at an eventual stabilization to an equilibrium point that is as close as possible to the social optimum.

1 Introduction

An important phase in the strategic planning process of a railway (or any public transportation) company is to establish a suitable *line plan*, i.e., to determine the routes of trains that serve the customers. In the *line planning* problem, we

* This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1581

are given a network $G = (V, L)$ (usually referred to as the public transportation network), where the node set V represents the set of stations (including important junctions of railway tracks) and the edge set L represents the direct connections or links (of railway tracks) between elements of V. A line is a path in G. Typically, a *line pool* is also provided, i.e., a set of potential lines among which the final set of lines will be decided. The *frequency* of a line l is a rational number indicating how often service to customers is provided along l within the planning period considered. For an edge $\ell \in L$, the *edge frequency* f_ℓ is the sum of the frequencies of the lines containing ℓ and is upper bounded by the *capacity* c_ℓ of ℓ, i.e., a maximum edge frequency established for safety reasons. The goal of the line planning problem is to provide the final set of lines offered by the public transportation company along with their frequencies (also known as the *line concept*).

The line planning problem has mainly been studied under two main approaches (see e.g., [6, 7]). In the *cost-oriented* approach, the goal is to minimize the costs of the public transportation company, under the constraint that all customers can be transported. In the *customer-oriented* approach, the goal is to maximize the number of customers with direct connections (under a similar constraint), or at least minimize the traveling time of the customers. A recent approach aims at minimizing the travel times over all customers including penalties for the transfers needed [9, 11].

The aforementioned approaches do not take into account certain fluctuations of input parameters; for instance, due to disruptions to daily operations (e.g., delays), or due to fluctuating customer demands. This aspect introduces the so-called *robust line planning* problem: provide a set of lines along with their frequencies, which are robust to fluctuations of input parameters. Very recently, a game theoretic approach for robust line planning was presented in [10]. In that model, the lines act as players and the strategies of the players correspond to line frequencies. Each player aims to minimize the expected delay of her own lines. The delay depends on the traffic load and hence on the frequencies of all lines in the network. The objective is to provide lines that are robust against delays. This is pursued by distributing the traffic load evenly over the network (respecting edge capacities) such that the probability of delays in the system is as small as possible.

In this work, we investigate a different perspective of robust line planning stemming from recent regulations in the railway sector (at least within Europe) that introduce competition and free railway markets, and set up a new application scenario: there is a (possibly large) number of *line operators* that should operate as commercial organizations, while the management of the network remains the responsibility of a single (typically governmental) entity; we shall refer to the latter as the *network operator*. Under this framework, line operators act as competing entities for the exploitation of shared goods and are (possibly) unwilling to reveal their actual level-of-satisfaction functions that determine their true incentives. Nevertheless, the network operator would like to ensure the maximum possible level of satisfaction of these competing entities, e.g., by maximizing the

(unknown due to privacy) aggregate levels of satisfaction. This would establish a notion of a socially optimal solution, which could also be seen as a fair solution in the sense that the average level of satisfaction is maximized. Additionally, the network operator should ensure that the operational costs of the whole system are covered by a fair cost sharing scheme announced to the competing entities. This implies that a (possibly anonymous) pricing scheme for the usage of the shared resources should be adopted that is *robust* against changes in the demands of the entities (line operators). That is, we consider as *robustness* of the system its tolerance to the entities' unknown incentives and elasticity of demand requests, and the eventual stabilization at an equilibrium point that is as close as possible to the social optimum.

In this paper, we explore this rationale by considering the case where the (selfishly motivated) line operators request frequencies (traffic demands) over a pool of already fixed line routes (one per line operator). Rather than requesting end-to-end frequencies, the line operators offer bids, which they (dynamically) update, for buying frequencies. Each line operator has a utility function determining her level of satisfaction that is *private*; i.e., she is not willing to reveal it to the network operator or her competitors, due to her competitive nature. The network operator announces an (anonymous) resource pricing scheme, which indirectly implies an allocation of frequencies to the line operators, given their own bids. By applying techniques from the network congestion control literature, we show that for the case of a single pool of routes, there exists a distributed, dynamic, (user) bidding and (resource) price updating protocol, whose equilibrium point is the unknown social optimum. We first study the single pool case, assuming strict concavity and monotonicity of the private utility functions. All dynamic updates of bids or prices may be done at the line operator or resource level, based only on local information, that concerns the particular line operator or resource. The key assumption is that the line operators can control only a negligible amount of frequency along a single line compared to its total frequency. We extend our technique to the case of multiple line pools, whose mix is determined by the network operator for the sake of social optimality, and prove similar results.

Our solution is robust against the imperfect knowledge imposed by the private (unknown) utility functions and the arbitrary (dynamically updated) bids, since the proposed protocol enforces convergence to an equilibrium which is the social optimum. Our approach introduces a new notion of robustness, which we call *incentive-compatible robustness*, that is complementary to the notion of *recoverable robustness* introduced in [2]. The latter appears to be more suitable in the context of railway optimization, as opposed to the classical notion of robustness within robust optimization; see [2] for a detailed discussion on the subject as well as for the limitations of the classical approach as suggested in [4].

Recoverable robustness is about computing solutions that are robust against a limited set of scenarios (that determine the imperfection of information) and which can be made feasible (recovered) by a limited effort. One starts from a feasible solution x of an optimization problem which a particular scenario s, that

introduces imperfect knowledge (i.e., by adding more constraints), may turn to infeasible. The goal is to have handy a recovery algorithm A that takes x and turns it to a feasible solution under s (i.e., under the new set of constraints). In other words, in recoverable robustness there is uncertainty about the feasibility space: imperfect information generates infeasibility and one strives to (re-)achieve feasibility.

Incentive-compatible robustness is about computing an incentive-compatible recovery scheme for achieving robustness (interpreted as convergence to optimality). By an incentive-compatible scheme, we mean that the players act (update their bids, in our application) in a selfish manner during the convergence sequence. In this context, the feasibility space is known and incomplete information refers to complete lack of information about the optimization problem, due to the unknown utility functions. The goal is to define an incentive-compatible (pricing) scheme so that the players converge (recover) to the system's optimum. In other words, in incentive-compatible robustness there is uncertainty about the objectives: feasibility is guaranteed, since imperfect knowledge does not introduce new constraints, and one strives to achieve optimality, exploiting the selfish nature of the players.

Note that incentive-compatible robustness is different from the concept of game-theoretic robustness as developed in [1]. The approach in [1] is a centralized, deterministic paradigm to uncertainty in strategic games, mainly in the flavor of the Bertsimas and Sim approach [4] to robust LP optimization. Our approach differs from that in the following: (i) It is decentralized to a large extent, based only on local information that the participating entities (line operators and resources) have at any time; (ii) we impose no restriction on the kind of the utility functions of the players, other than their strict concavity, whereas the approach in [1] has to somehow quantify the "magnitude" of uncertainty of the constraints and/or the payoffs, in order to keep the solvability of the problem comparable to that of the nominal counterpart; (iii) the solvability of the robust counterpart in [1] is largely based on the solvability of the nominal counterpart (which is strongly questionable for the general game-theoretic framework).

Related to our work is that of Borndörfer et al [5] that considers the allocation of slots in railway networks. That work considers the improvement of existing schedules of lines and frequencies, by reconsidering the allocation of (scarce) bundles of slots (i.e., lines with given frequencies in our own terminology) that have positive synergies with each other. The remaining schedule is assumed to remain intact, so that the resulting optimization problem is solvable. Initially, the involved users (line operators) make some bids and consequently a centralized optimization problem is solved to determine the changes in the allocation of these slots so as to maximize the welfare of the whole system. This approach is different from ours in the following points: (i) It assumes no incentive-compatibility for the involved users and the eventual allocation is determined by a centralized scheduler. In our case, there is a simple pricing policy per resource (track), which is a priori known to all the players, and the winner is determined by the players' bids. The selfish behavior of the line operators (in our case) is, not only taken

into account, but also exploited by the system in order to assure convergence to the social optimum of the whole network. (ii) The approach in [5] makes some local improvements in hope of improving the whole system, but does not exclude being trapped at some local optimum, which may be far away from the social optimum of the system. Our proposed scheme provably converges towards the social optimum, even if changes in the parameters of the game (e.g., in the players' secret utilities) change in the future. (iii) In [5], it is required that a centralized optimization problem is solved (considering the data regarding the whole network) and its solution is enforced in the current schedule. In our work (at least for the single-pool case) there is no need for global knowledge of the whole network. Each player dynamically adapts her bids according to her own (secret) utility and the aggregate cost she faces along her own path.

The rest of this paper is organized as follows. Section 2 provides the set up of our modeling. The decentralized pricing mechanism both for the single and the multiple line pool case is given in Section 3. We conclude in Section 4.

2 The Model

Suppose that a set P of line operators behave as competing service providers, willing to offer regular train routes to the end users of a railway public transportation system. The railway network operator provides the (aforementioned) public transportation network $G = (V, L)$, with the set L of edges (railway tracks connecting directly two nodes of G) being the *resources* of the network. These resources are assumed to be subject to (fixed) capacity constraints, described by the capacity vector $\mathbf{c} = (c_\ell)_{\ell \in L} > 0$. The capacity of each edge is considered as a shared resource provided by the network operator.

There is a fixed pool of routes (i.e., origin–destination paths), one per line operator, that the line operators are willing to use. This pool is represented by a **routing matrix** $R \in \{0,1\}^{|L| \times |P|}$, in which each row $R_{\ell,\star}$ corresponds to a different edge $\ell \in L$, and each column $R_{\star,p}$ corresponds (actually, is the characteristic vector of) the route of a distinct line operator $p \in P$. Each line operator $p \in P$ has complete control over the *frequency* or *traffic rate* (of trains) she decides to route over her path, $R_{\star,p}$, given that no edge capacity constraint is violated in the network. A utility function $U_p : \mathbb{R} \mapsto \mathbb{R}$ determines the level of satisfaction of the line operator $p \in P$ for committing an end-to-end traffic rate $x_p > 0$ along her route $R_{\star,p}$, for the purposes of her clients. These utility functions are assumed to be strictly increasing, strictly concave, nonnegative real functions of the end–to–end traffic rate x_p allocated to the line operator $p \in P$. It is also assumed that these functions are *private*: Each line operator is not willing to reveal it to the network operator or her competitors, due to her competitive nature.

The railway network operator is only interested in having a socially optimal (fair) solution. This is usually interpreted as maximizing the aggregate satisfaction of the line operators. Therefore, the social welfare objective is considered to be the maximization of the aggregate utilities of the line operators, subject

to the capacity constraints. That is, the network operator is interested in the solution of the following convex optimization[1] problem:

$$\boxed{\text{SOCIAL}} \quad \max \left\{ \sum_{p \in P} U_p(x_p) : R\mathbf{x} \leq \mathbf{c};\ \mathbf{x} \geq \mathbf{0} \right\}$$

Since all utility functions are strictly concave, then SOCIAL has a unique optimal solution, which is the social optimum. To solve SOCIAL directly, the network operator, apart from the inherent difficulty in centrally solving (even convex) optimization programs of the size of a railway network, faces the additional obstacle of *not knowing* the exact shape of the objective function. Moreover, there exist some operational costs that have to be split among the line operators who use the infrastructure, and this has to be done also in a fair way: Each line operator should only be charged for the usage of the resources standing on her own route. In addition, the per–unit cost for using a line should be independent of the line operator's identity (i.e., we would like to have an *anonymous* pricing scheme for using the resources). But of course, this cost depends on the aggregate congestion induced by all the line operators in these edges, due to the congestion effect. Indeed, it would be desirable for the network operator to be able to exploit the announcement of a pricing scheme not only for covering these operational costs, but also in such a way that a fair solution for all the line operators is induced, despite the fact that there is no global knowledge of the exact utility functions of the line operators.

In this work, we explore the possibilities of having such a pricing and traffic rate allocation mechanism. We would like this mechanism to depend only on the information affecting either a specific line operator (e.g., the amount of money she is willing to spend) or a specific resource (e.g., the aggregate congestion induced by the line operators' demands on this resource), but as we shall see this is not always possible.

As for the line operators (the players), each of them is interested in selfishly utilizing her own payoff, which is determined by the difference of the private utility value minus the operational cost that the network operator charges her for claiming an amount of traffic rate along her own route. The strategy space of a line operator is to claim (via bidding) the value of the traffic rate she is willing to buy, subject to the global capacity constraints (for all the players). It is mentioned here that this linear combination of the private utility and the cost share is not a real restriction, as there is no restriction for the shape of the utility function, other than the strict concavity and the monotonicity, which are quite natural assumptions.

[1] We make the tacit assumption that convex optimization refers to minimizing a convex function f, which is equivalent to maximizing the concave function $-f$.

3 A Decentralized Pricing Scheme

The selfish perspective of the competing line operators (the players) implies a strategic game among them, in which the network operator intervenes only implicitly (as the game designer), by setting the resource usage (per–unit) costs. In order to study the effect of the selfish behavior in this setting, we consider the following **Frequency Game** in Line Planning:

- Each player $p \in P$ is a line operator, whose strategy is to choose a line frequency (traffic rate) over her (already fixed) route $R_{\star,p}$ connecting her own origin–destination pair (s_p, t_p) of stations/stops.
- The strategy space for all the players is the set of feasible flows from origin to destination nodes, so that the edge capacity constraints are preserved. That is, the strategy space of the game is the set of vectors $\left\{\mathbf{x} \in \mathbb{R}_{\geq 0}^{|P|} : R\mathbf{x} \leq \mathbf{c}\right\}$.
- Each player's payoff is determined both by the value of the private utility function $U_p(x_p)$ (for having a traffic rate of x_p over her route) and the operational cost $C_p(\mathbf{x})$ she has to pay along her own route, due to the required traffic rate vector $\mathbf{x}$ induced by all the players in the network. Hence, player p's individual payoff is defined as: $IP_p(x_p, \mathbf{x}_{-p}) = U_p(x_p) - C_p(x_p, \mathbf{x}_{-p})$, where $\mathbf{x}_{-p}$ is the traffic rate vector for all the players but for player p. Therefore, the sole goal of player $p \in P$ is to choose her traffic rate so as to maximize her individual payoff:

$$\boxed{\text{USER}} \quad \max\{IP(x_p, \mathbf{x}_{-p}) = U_p(x_p) - C_p(x_p, \mathbf{x}_{-p}) : x_p \geq 0\}$$

- We consider as shared resources the capacities of the available network edges, for which the line operators compete with each other.

As we shall see shortly, we will allow the players to affect their own choices (traffic rates) only indirectly, via bidding. That is, each player is not assumed to freely choose her own traffic rate along her route, but rather offer a larger bid for (hopefully) getting higher traffic rate.

3.1 Describing the Social Optimum

Due to our assumption on the convexity of $\boxed{\text{SOCIAL}}$, we know that a traffic rate vector $\hat{\mathbf{x}}$ is optimal for it (we call it the social optimum) if there exists a vector of Lagrange Multipliers $\hat{\lambda} = (\hat{\lambda}_\ell)_{\ell \in L}$ satisfying the following Karush-Kuhn-Tucker (KKT) conditions (see e.g., [3, Chap. 3]):

KKT-SOCIAL

$$U_p'(\hat{x}_p) = \hat{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P, \tag{1}$$

$$\hat{\lambda}_\ell \left(c_\ell - R_{\ell,\star} \cdot \hat{\mathbf{x}}\right) = 0, \quad \forall \ell \in L, \tag{2}$$

$$R_{\ell,\star} \cdot \hat{\mathbf{x}} \leq c_\ell, \quad \forall \ell \in L, \tag{3}$$

$$\hat{\lambda}, \hat{\mathbf{x}} \geq \mathbf{0} \tag{4}$$

Of course, the problem with the KKT-SOCIAL system is that the utility functions (and hence their derivatives) are unknown to the system. The question is whether there exists a way for the network designer to enforce the optimal solution of SOCIAL, also described in KKT-SOCIAL, without demanding this knowledge. The answer to this is *partially affirmative*, and this is by exploiting the selfish nature of the players (line operators) as we shall see shortly.

3.2 Setting the Right Pricing Scheme for the Players

In order to allow usage of his resources (the capacities of the edges in the network), the network operator has to define a pricing scheme that will (at least) pay back the operational costs of the edges. This scheme should be anonymous, in the sense that all the line operators willing to use a given edge, will have to pay the same per–unit–of–frequency price for using it. But these prices may vary for different edges, depending on the popularity and the availability of each edge.

For the moment let's assume that we already know the optimal Lagrange Multipliers, $(\hat{\lambda}_\ell)_{\ell \in L}$ of KKT-SOCIAL. Interpreting these values as the per–unit–of–frequency prices of the resources, we have a pricing scheme for the traffic induced by the line operators to their own routes: Each line operator pays exactly for the marginal cost of her own traffic rate at the resources she uses in her route. That is,

$$\forall p \in P, C_p(x_p, \mathbf{x}_{-p}) = \hat{\mu}_p \cdot x_p$$

where $\hat{\mu}_p \equiv \sum_{\ell \in L: R_{\ell,p}=1} \hat{\lambda}_\ell = \hat{\lambda}^T R_{\star,p}$ is the per–unit price for committing a unit of traffic along the route $R_{\star,p}$ of player $p \in P$.

One should mention here that indeed there is an indirect effect of the other players' congestion in the marginal cost of each player, despite the fact that this seems to be only linear in her own traffic rate. This is because the scalar $\hat{\mu}_p$ actually depends on the optimal primal–dual pair $(\hat{\mathbf{x}}, \hat{\lambda})$.

We next assume that the players are actually controlling only *negligible* amounts of traffic rates compared to the aggregate ones[2]. Then, their effect in the total congestion (and therefore in the values of the marginal prices) is also negligible. This implies that the players consider the per-unit-prices they face to be constant, even if this is actually affected by the traffic rate vector as well. In such a case we say that the players are **price takers**, i.e., they accept the prices without anticipating to have an effect on them by their own strategy. In such a case each player solves the following optimization problem:

$$\boxed{\text{USER-I}} \quad \max \{U_p(x_p) - \hat{\mu}_p x_p : x_p \geq 0\}$$

Due to the convexity of USER-I, $\tilde{x}_p \geq 0$ is an optimal solution if $U'_p(\tilde{x}_p) = \hat{\mu}_p$. That is, each player (selfishly) tries to satisfy her own part of the first set of equalities in KKT-SOCIAL. Of course, we still have to deal with the crucial

[2] For the considered application scenario, this is not unrealistic.

problem that the optimal Lagrange Multipliers (that define the marginal prices for the users) cannot be directly computed, due to both the size of SOCIAL and the lack of knowledge of the private utility functions, in the framework of railway optimization.

In order to handle this situation, we consider the following two-level scenario for dynamically setting per–unit prices of the edges and frequencies of the selfish players: Initially each player $p \in P$ announces a bid $w_p \geq 0$ concerning the total amount of money she is willing to pay for buying traffic rate along her own route. The exact amount of traffic rate that she will eventually buy, depends on the per-unit price that will be announced by the network operator, and is not yet known to her. Consequently, the network designer considers the following optimization problem, whose Lagrange Multipliers define the per–unit prices of the edges:

$$\boxed{\text{NETWORK}} \quad \max \left\{ \sum_{p \in P} w_p \cdot \log(x_p) : R\mathbf{x} \leq \mathbf{c};\ \mathbf{x} \geq \mathbf{0} \right\}$$

That is, the network operator considers that the private utility $U_p(x_p)$ is substituted by the (also strictly concave and increasing) function $w_p \log(x_p)$. The choice of this function along with the selfishness of the players allows us to obtain a convex program with linear inequalities, whose KKT system is very similar (except for the first line) to KKT-SOCIAL:

KKT-NETWORK

$$\frac{w_p}{\bar{x}_p} = \bar{\lambda}^T \cdot R_{\star,p}, \quad \forall p \in P, \tag{5}$$

$$\bar{\lambda}_\ell \left(c_\ell - R_{\ell,\star} \cdot \bar{\mathbf{x}}\right) = 0, \quad \forall \ell \in L, \tag{6}$$

$$R_{\ell,\star} \cdot \bar{\mathbf{x}} \leq c_\ell, \quad \forall \ell \in L, \tag{7}$$

$$\bar{\lambda}, \bar{\mathbf{x}} \geq \mathbf{0} \tag{8}$$

At this point, one could argue that the convex program NETWORK could be directly solved, and compute (along with KKT-NETWORK) the requested Lagrange Multipliers. The huge scale of a railway network optimization instance makes this approach rather unappealing. Therefore, we shall compute an optimal solution of NETWORK in a distributed fashion, as follows:

– Each edge is equipped with a dynamically updated charging mechanism, which is the same (per–unit) price for all the line operators using it. This charging mechanism is updated according to the following system of differential equations:

$$\forall \ell \in L, \quad \dot{\lambda}_\ell(t) = \max\{y_\ell(t) - c_\ell, 0\} \cdot \mathbb{I}_{\{\lambda_\ell(t)=0\}} + (y_\ell(t) - c_\ell) \cdot \mathbb{I}_{\{\lambda_\ell(t)>0\}} \tag{9}$$

where $y_\ell(t) \equiv \sum_{p \in R: R_{\ell,p}=1} x_p(t) = R_{\ell,\star} \cdot \mathbf{x}(\mathbf{t})$ is the cumulative traffic rate committed at edge $\ell \in L$ at time $t \geq 0$, and $\mathbb{I}_{\{\mathcal{E}\}}$ is the indicator variable of

the truth of a logical expression $\mathcal{E}$. The system of differential equations (9) is obtained from the well-known approach (see e.g., [12]) that considers the Lagrange multipliers of an optimization problem as the (per unit) prices of the resources corresponding to the constraints represented by each Lagrange multiplier. Therefore, the above system has the following intuitive interpretation. For each resource ℓ that currently has a zero price, the tendency is to increase the price only if this resource is over-used (i.e., the aggregate traffic rate exceeds the capacity of the resource). When a resource has positive price, then the tendency is either to increase or reduce this price, depending on whether its current traffic rate is below or exceeds the capacity of the resource, respectively. Thus, the only stable situation is only when a resource is either under-used and has zero price (since there is no interest in using the residual capacity), or its traffic has already reached its capacity. Observe that the equilibrium of this system of differential equations has $\forall \ell \in L, \bar{y}_\ell \equiv R_{\ell,\star} \cdot \bar{\mathbf{x}} = c_\ell \quad \vee \quad \bar{\lambda}_\ell = 0$. That is, the complementarity conditions of both KKT-SOCIAL and KKT-NETWORK (equations (2) and (6)) are satisfied.

– Each line operator $p \in P$ is charged an instantaneous per-unit price $\mu_p(t) \equiv \sum_{\ell \in L: R_{\ell,p}=1} \lambda_\ell(t) = \lambda(\mathbf{t})^T \cdot R_{\star,p}$, at any time $t \geq 0$. Therefore, given their commitment on spending w_p for buying traffic rate, at equilibrium player p is allocated a traffic rate $\bar{x}_p = \frac{w_p}{\bar{\mu}_p}$. From this we deduce that at equilibrium also the equations (5) of KKT-NETWORK are satisfied.

The above distributed scheme is a congestion control algorithm, in which each player (line operator) reacts to signals she gets about the congestion along her route. These signals are the per-unit prices $\mu_p(t)$ that the line operator gets from the network operator at any time.

The question is whether the above system converges at all. This is indeed true, if we assume that the routing matrix R has full rank. This assures that given a set $\lambda(t) = (\lambda_\ell(t))_{\ell \in L}$ of instantaneous per-unit prices at the resources, the set $\mu(t) = (\mu_p(t))_{p \in P}$ of per-unit prices for the line operators, that is computed as the solution of the system $\mu(t) = R^T \cdot \lambda(t)$, is *unique*. Using a proper Lyapunov function argument, it can be shown (cf. [8, Chapter 22]) that this dynamic (and distributedly implemented) pricing scheme, for *fixed* player bids $(w_p)_{p \in P}$, is stable and converges to the optimal solution $(\bar{\mathbf{x}}, \bar{\lambda})$ of NETWORK.

In particular, consider the Lyapunov function $V(\lambda(t)) = \frac{1}{2}(\lambda(t) - \bar{\lambda})^T(\lambda(t) - \bar{\lambda})$. To show stability of our scheme, it suffices to show that $dV(\lambda(t))/dt \leq 0$.

Then we have:

$$\begin{aligned}
\frac{dV(\lambda(t))}{dt} &= \sum_{\ell\in L}(\lambda_\ell(t)-\bar{\lambda}_\ell)\cdot\dot{\lambda}(t)\\
&= \sum_{\ell\in L}(\lambda_\ell(t)-\bar{\lambda}_\ell)\cdot[\max\{y_\ell(t)-c_\ell,0\}\cdot\mathbb{I}_{\{\lambda_\ell(t)=0\}}+(y_\ell(t)-c_\ell)\cdot\mathbb{I}_{\{\lambda_\ell(t)>0\}}]\\
&\leq \sum_{\ell\in L}(\lambda_\ell(t)-\bar{\lambda}_\ell)\cdot(y_\ell(t)-c_\ell)\\
&= \sum_{\ell\in L}(\lambda_\ell(t)-\bar{\lambda}_\ell)\cdot[(y_\ell(t)-\bar{y}_\ell)+(\bar{y}_\ell-c_\ell)]\\
&\leq \sum_{\ell\in L}(\lambda_\ell(t)-\bar{\lambda}_\ell)\cdot(y_\ell(t)-\bar{y}_\ell)\\
&= \sum_{\ell\in L}(\lambda_\ell(t)-\bar{\lambda}_\ell)\cdot R_{\ell,\star}\cdot(\mathbf{x}(t)-\bar{\mathbf{x}})
\end{aligned}$$

$$\begin{aligned}
&= \sum_{p\in P}(\mu_p(t)-\bar{\mu}_p)\cdot(x_p(t)-\bar{x}_p)\\
&= \sum_{p\in P}\left(\frac{w_p}{x_p(t)}-\frac{w_p}{\bar{x}_p}\right)\cdot(x_p(t)-\bar{x}_p)=\sum_{p\in P}w_p\cdot\left(2-\frac{x_p(t)}{\bar{x}}-\frac{\bar{x}_p}{x_p(t)}\right)\\
&\leq 0
\end{aligned}$$

The first inequality holds because: $\forall\ell\in L$, (i) if $\lambda_\ell(t)>0$ then $\dot{\lambda}_\ell(t)=y_\ell-c_\ell$; (ii) if $\lambda_\ell(t)=0$ then $\max\{y_\ell-c_\ell,0\}\geq 0$ and $\lambda_\ell(t)-\bar{\lambda}_\ell=-\bar{\lambda}_\ell\leq 0$. Therefore, for $\lambda_\ell(t)=0$ it holds that $(\lambda_\ell(t)-\bar{\lambda}_\ell)\max\{y_\ell(t)-c_\ell,0\}=-\bar{\lambda}_\ell\max\{y_\ell(t)-c_\ell,0\}\leq 0$. But so long as $\lambda(t)=0$, it holds that the total flow $y_\ell(t)$ is at most as large as the capacity c_ℓ (otherwise the price for this resource would have raised earlier). That is, $0\leq-\bar{\lambda}_\ell(y_\ell(t)-c_\ell)$. The second inequality holds because at equilibrium no aggregate flow $\bar{y}_\ell$ can exceed the capacity c_ℓ of the resource, and $\bar{\lambda}_\ell(\bar{y}_\ell-c_\ell)=0$. The third inequality holds because $\forall z>0, z+\frac{1}{z}\geq 2\Rightarrow 2-z-\frac{1}{z}\leq 0$. We have also exploited the facts that $\forall t\geq 0$, $\mathbf{y}(t)=R\cdot\mathbf{x}(t)$ and $\mu(t)=\lambda(t)^T\cdot R$.

Let's now return to the line operators. We have assumed that these players announce some fixed bids, but the truth is that since the pricing scheme changes over time, it is in the interest of each of them to vary her own bid. Indeed, if the players are assumed to be price takers and act myopically (i.e., without anticipating to affect the prices via their own pricing policy), then they will try to solve the following system, which is parameterized by the instantaneous set of per–unit prices $\mu(\mathbf{t})=(\mu_p(t))_{p\in P}$ (seen as constants) they are charged at time $t\geq 0$:

$$\boxed{\text{USER-II}}\quad \max\left\{U_p\left(\frac{w_p}{\mu_p(t)}\right)-w_p : w_p\geq 0\right\}$$

Due to convexity, the optimal solution $\tilde{w}_p(t)$ of this unconstrained optimization program, will be the bid chosen by player $p\in P$ at time $t\geq 0$, and is be given

by:

$$\frac{1}{\mu_p(t)} \cdot U_p'\left(\frac{\tilde{w}_p(t)}{\mu_p(t)}\right) = 1 \Leftrightarrow$$
$$U_p'\left(\tilde{x}_p(t)\right) = U_p'\left(\frac{\tilde{w}_p(t)}{\mu_p(t)}\right) = \mu_p(t) \Leftrightarrow$$
$$\tilde{x}_p(t)U_p'\left(\tilde{x}_p(t)\right) = \mu_p(t) \cdot \tilde{x}_p(t) = \tilde{w}_p(t)$$

That is, the price taking, myopic players have an incentive to set their bids properly so that $\forall t \geq 0, \forall p \in P, w_p(t) = x_p(t)U_p'(x_p(t))$. This will also hold at equilibrium, i.e., $\forall p \in P, \bar{w}_p = \bar{x}_p U_p'(\bar{x}_p)$. But when this is true, it also holds that KKT-NETWORK and KKT-SOCIAL coincide. Therefore, the selfish behavior of the myopic, price taking players, under the dynamic price setting mechanism and bidding scheme, converges to the optimal solution $(\hat{\mathbf{x}}, \hat{\lambda})$ of SOCIAL.

3.3 Extension to Multiple Pools of Routes

In this subsection we extend the freedom of the railway network operator, assuming that he can periodically use different pools of routes for the players, from a set K of pools. The set of different pools is motivated by the fact that usually there are dependencies between lines; for instance, the choice of a high-speed line affects the choice of lines for other trains. These dependencies split naturally the set of all lines into a small number of subsets determined by the network operator, resulting in different line pools.

Each pool $k \in K$ is represented by its routing matrix $R(k) \in \{0,1\}^{|L|\times|P|}$, as before. The line operators still try to have (indirect) control only over the end-to-end traffic rates they get by the network operator. We assume that player $p \in P$ gets a unique traffic rate x_p for the whole period of time considered. It is up to the network operator to decide how to multiplex the distinct pools of routes, in order to achieve the optimal social welfare value. That is, the network operator now directly participates in the optimization problem, via the variables $f_k : k \in K$ indicating the portion of time each pool consumes over the whole time period we study. This social welfare optimization problem is the following:

MULTI-SOCIAL

$$\max\left\{\sum_{p\in P} U_p(x_p) : \forall \ell \in L, \forall k \in K, (R(k))_{\ell,\star} \cdot \mathbf{x} \leq c_\ell \cdot f_k;\ \sum_{k\in K} f_k \leq 1;\ \mathbf{x}, \mathbf{f} \geq \mathbf{0}\right\}$$

The Lagrangian function is the following:

$$
\begin{aligned}
&L(\mathbf{x}, \mathbf{f}, \Lambda, \zeta) \\
&= \sum_{p \in P} U_p(x_p) - \sum_{\ell \in L} \sum_{k \in K} \Lambda_{\ell,k} \cdot [(R(k))_{\ell,\star} \cdot \mathbf{x} - c_\ell \cdot f_k] - \zeta \left[\sum_{k \in K} f_k - 1 \right] \\
&= \sum_{p \in P} \left[U_p(x_p) - \sum_{\ell \in L} \sum_{k \in K} \Lambda_{\ell,k} \cdot (R(k))_{\ell,p} \cdot x_p \right] + \sum_{k \in K} f_k \cdot \left[\mathbf{c}^T \Lambda_{\star,k} - \zeta \right] + \zeta
\end{aligned}
$$

If we set $\mu_p(\Lambda) = \sum_{\ell \in L} \sum_{k \in K} \Lambda_{\ell,k} \cdot (R(k))_{\ell,p} = \sum_{k \in K} \Lambda_{\star,k}{}^T (R(k))_{\star,p}$, the system of KKT conditions of MULTI-SOCIAL is written as follows:

KKT-MULTI-SOCIAL

$$
\begin{aligned}
U'(\hat{x}_p) &= \hat{\mu}_p \equiv \mu_p(\hat{\Lambda}), \quad \forall p \in P, && (10)\\
\mathbf{c}^T \cdot \hat{\Lambda}_k = \sum_{\ell \in L} \hat{\Lambda}_{\ell,k} \cdot c_\ell &= \hat{\zeta}, \quad \forall k \in K, && (11)\\
\hat{\Lambda}_{\ell,k} \cdot \left[(R(k))_{\ell,\star} \cdot \hat{\mathbf{x}} - c_\ell \cdot \hat{f}_k \right] &= 0, \quad \forall \ell \in L, \forall k \in K, && (12)\\
\hat{\zeta} \cdot \left(\sum_{k \in K} \hat{f}_k - 1 \right) &= 0 && (13)\\
(R(k))_{\ell,\star} \cdot \hat{\mathbf{x}} &\leq c_\ell \cdot \hat{f}_k, \quad \forall \ell \in L, && (14)\\
\sum_{k \in K} \hat{f}_k &\leq 1, && (15)\\
\hat{\mathbf{x}} \geq \mathbf{0}, \quad \hat{\mathbf{f}} \geq \mathbf{0}, \hat{\mathbf{\Lambda}} \geq \mathbf{0}, \hat{\zeta} \geq 0 && && (16)
\end{aligned}
$$

Observe that, by equation (11), in the optimal solution all the pools have the same weighted aggregate price, equal to $\hat{\zeta}$, if we use the edge capacities as weights. Moreover (due to equation (13)), unless this optimal aggregate price is zero, it holds that the edge capacities are totally distributed among the distinct pools: **if** $\hat{\zeta} > 0$ **then** $\sum_{k \in K} \hat{f}_k = 1$.

We can once more set the instantaneous per-unit prices for the players as a (linear) function of the Lagrange Multipliers, as follows:

$$
\forall t \geq 0, \ \forall p \in P, \ \mu_p(t) = \sum_{\ell \in L} \sum_{k \in K} \Lambda_{\ell,k}(t) \cdot (R(k))_{\ell,p} = \sum_{k \in K} (\Lambda_k(t))^T \cdot (R(k))_p
$$

The congestion at edge $\ell \in L$ due to route $R(k)$ at time $t \geq 0$, is given by $y_{\ell,k}(t) = (R(k))_{\ell,\star} \cdot \mathbf{x}(t)$. A dynamic (by the edges) pricing scheme, is described by the following system of differential equations:

$$
\begin{aligned}
&\forall \ell \in L, \ \forall k \in K, \\
&\dot{\Lambda}_{\ell,k}(t) = \max\{y_{\ell,k}(t) - c_\ell f_k, 0\} \cdot \mathbb{I}_{\{\Lambda_{\ell,k}(t)=0\}} + (y_{\ell,k}(t) - c_\ell f_k) \cdot \mathbb{I}_{\{\Lambda_{\ell,k}(t)>0\}} && (17)
\end{aligned}
$$

This differential system would then assure the validity of equations (12), at equilibrium, if the pool frequencies provided by the network operator were fixed. Assuming again that the players announce (instantaneous) bids, and providing them with a traffic rate $x_p(t) = w_p(t)/\mu_p(t)$ at any time, along with their price taking and myopic behavior, would also assure (at equilibrium) the validity of equations (10).

But we also have to assure the validity of equations (11) and (13). In order to achieve this, we study the dual of MULTI-SOCIAL: The dual problem of MULTI-SOCIAL is the following:

$$\boxed{\text{DUAL-MULTI-SOCIAL}} \quad \max\{D(\Lambda,\zeta) : \forall \ell \in L, \forall k \in K, \Lambda_{\ell,k} \geq 0; \zeta \geq 0\}$$

where:

$$\begin{aligned} D(\Lambda,\zeta) &= \max\{L(\mathbf{x},\mathbf{f},\Lambda,\zeta) : \mathbf{x},\mathbf{f} \geq \mathbf{0}\} \\ &= \max_{\mathbf{x}\geq\mathbf{0}} \left\{ \sum_{p\in P} \left[U_p(x_p) - \sum_{\ell\in L}\sum_{k\in K} \Lambda_{\ell,k} \cdot (R(k))_{\ell,p} \cdot x_p \right] \right\} \\ &+ \max_{\mathbf{f}\geq\mathbf{0}} \left\{ \sum_{k\in K} f_k \cdot \left[\sum_{\ell\in L} \Lambda_{\ell,k} \cdot c_\ell - \zeta \right] \right\} + \zeta \end{aligned}$$

Observe that the dual objective $D(\Lambda,\zeta)$ is split in two parts. The first part

$$F(\Lambda) = \max_{\mathbf{x}\geq\mathbf{0}} \left\{ \sum_{p\in P} \left[U_p(x_p) - \sum_{\ell\in L}\sum_{k\in K} \Lambda_{\ell,k} \cdot (R(k))_{\ell,p} \cdot x_p \right] \right\}$$

is a maximization problem similar to the one already dealt with in the single pool case of the previous section. The second part

$$\begin{aligned} G(\Lambda,\zeta) &= \max_{\mathbf{f}\geq\mathbf{0}} \left\{ \sum_{k\in K} f_k \cdot \left[\sum_{\ell\in L} \Lambda_{\ell,k} \cdot c_\ell - \zeta \right] \right\} + \zeta \\ &= \max_{\mathbf{f}\geq\mathbf{0}} \left\{ \sum_{k\in K} f_k \cdot \left(\mathbf{c}^T \Lambda_{\star,k}\right) + \zeta \cdot \left(1 - \sum_{k\in K} f_k \right) \right\} \end{aligned}$$

It is now clear that so long as there exists a pool with weighted aggregate price (using the capacities as weights) strictly larger than the value of ζ, the optimal choice of $\mathbf{f}$ is to be a probability distribution that assigns positive mass only to pools of maximum aggregate price (according to Λ). Therefore:

$$\begin{aligned} G(\Lambda,\zeta) &= \max_{\mathbf{1}^T\mathbf{f}=1;\ \mathbf{f}\geq\mathbf{0}} \left\{ \sum_{k\in K} f_k \cdot \left(\mathbf{c}^T \Lambda_{\star,k}\right) \right\} = \max_{k\in K} \left\{ \mathbf{c}^T \Lambda_{\star,k} \right\} \\ &= \min\left\{ z : z \cdot \mathbf{1}^T \geq \mathbf{c}^T \Lambda \right\} \end{aligned}$$

We augment our price updating scheme described in the system (17) as follows:

– After the players having announced the current bids $(w_p(t))_{p\in P}$ and (consequently) the edges having updated their current prices in each pool $(\Lambda_{\ell,k}(t))_{\ell\in L,k\in K}$, $\zeta(t)$ is set to the *average* price of a pool:

$$\zeta(t) = \frac{1}{|K|}\sum_{k\in K} \mathbf{c}^T(\Lambda(t))_{\star,k} \tag{18}$$

– The network operator then updates the portions of time granted to each of the pools, so that pools exceeding the current average price tend to increase their portion of time (in hope of decreasing their weighted cost), while pools that are cheaper than the average price slightly decrease their portion of time. That is:

$$\forall k \in K,\ \dot{f}_k(t) = \phi \cdot \max\left\{0, \mathbf{c}^T \cdot (\Lambda(t))_{\star,k} - \zeta(t)\right\} \tag{19}$$

where, $\phi > 0$ is a scaling factor ensuring that the resulting vector $\mathbf{f}$ is again a probability distribution over the pools.

Observe that, at equilibrium $(\hat{\mathbf{x}}, \hat{\mathbf{f}}, \hat{\Lambda}, \hat{\zeta})$, our augmented pricing scheme has all pools with the same aggregate price: $\forall k \in K, \hat{\mathbf{c}}^T \cdot \hat{\Lambda}_{\star,k} = \hat{\zeta}$. It is only then that the time portions of the pools stabilize (cf. equation (19)). This assures the validity of equations (11). Moreover, by brute force we assure the validity of equation (13).

It is mentioned at this point that the two updating schemes concerning the average pool price and the vector of time portions of the pools, have to be centrally computed by the network operator, since it is his decision how to change them and these changes take into account the state of the whole network. Unfortunately we cannot (at least in this approach) avoid this bottleneck, since we involve the network operator in the competing environment.

4 Conclusions and Open Issues

We presented a distributed protocol for a new application scenario in line planning that achieves *incentive-compatible robust* solutions. Our protocol allows line operators to negotiate line frequencies over fixed lines in a dynamic fashion. In a broader context, our approach comprises a generic technique to set up a dynamic market for (re-)negotiating usage of resources over subsets of resources. Consequently, it could be applied to set up a dynamic frequency market over other transportation settings (e.g., in the airline domain).

A crucial question would be to devise protocols that demonstrate *fast* convergence to the equilibrium point, even approximately. Additionally, it would be interesting to find ways to tackle the assumption on price taking and myopic behavior of the users. It would be nice to do this even at the cost of suboptimal equilibrium points. It is noted that when the players are not price takers and myopic (they are called then *price anticipators* in the congestion control jargon), then the above scheme does not lead to socially optimal solutions, even for the

case where there is only a single resource to share. Nevertheless, it would be quite interesting to know how far one can be from the social optimum, given that a distributed (and localized) updating scheme is adopted for the user requests and the prices of the resources.

Acknowledgements. We are indebted to Rolf Möhring, Christian Liebchen, and Sebastian Stiller for their comments on an earlier draft of this work, and to Kostas Tsihlas for many fruitful discussions.

References

1. M. Aghassi and D. Bertsimas, "Robust Game Theory", *Mathematical Programming Ser. B*, 107 (2006), pp. 231-273.
2. ARRIVAL Deliverable D1.2, "New Theoretical Notion of the Prices of Robustness and Recoverability", ARRIVAL Project, Version 2, July 2007.
3. D. Bertsekas, *Nonlinear Programming*, Athena Scientific, 2nd Edition, 1999.
4. D. Bertsimas and M. Sim, "The Price of Robustness", *Operations Research*, 52:1 (2004), pp. 35-53.
5. R. Borndörfer, M. Grötschel, S. Lukac, M. Mitusch, T. Schlechte, S. Schultz, A. Tanner, "An Auctioning Approach to Railway Slot Allocation", *Competition and Regulation in Network Industries*, 1:2 (2006), pp. 163–196.
6. H. Dienst, "Linienplanung im spurgeführten Personenverkehr mit Hilfe eines heuristischen Verfahrens", PhD thesis, Technische Universität Braunschweig, 1978.
7. J. Goossens, C. van Hoesel, and L. Kroon, "A branch and cut approach for solving line planning problems", *Transportation Science* 38 (2004), pp. 379393.
8. A. Ozdaglar and R. Srikant, "Incentives and Pricing in Communications Networks", Chapter 22 in *Algorithmic Game Theory*, ed. N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, Cambridge University Press, 2007.
9. A. Schöbel and S. Scholl, "Line Planning with Minimal Traveling Time", In *Proc. 5th Workshop on Algorithmic Methods and Models for Optimization of Railways* – ATMOS 2005.
10. A. Schöbel and S. Schwarze, "A Game-Theoretic Approach to Line Planning", in *Proc. 6th Workshop on Algorithmic Methods and Models for Optimization of Railways* – ATMOS 2006.
11. S. Scholl, "Customer-oriented line planning", PhD thesis, Technische Universität Kaiserslautern, 2005.
12. R. Srikant, *The Mathematics of Internet Congestion Control*, Birkauser, 2004.

Simultaneous Network Line Planning and Traffic Assignment

Karl Nachtigall and Karl Jerosch

Technical University Dresden, Faculty of Traffic Sciences,
Institute for Logistics and Aviation
karl.nachtigall@tu-dresden.de
karl-friedemann.jerosch@siemens.com

Abstract. One of the basic problems in strategic planning of public and rail transport is the line planning problem to find a system of lines and its associated frequencies. The objectives of this planning process are usually manifold and often contradicting. The transport operator wants to minimize cost, whereas passengers want to have travel time shortest routes without any or only few changings between different lines. The travel quality of a passenger route depends on the travel time and on the number of necessary changings between lines and is usually measured by a disutility or impedance function. In practice the disutility strongly depends on the line plan, which is not known, but should be calculated. The presented model combines line planning models and traffic assignment model to overcome this dilemma. Results with data of Berlin's city public transportion network are reported.

Key words: line planning problem, integer programming

1 Related literature

In the last years a lot of mathematical integer programming models have been proposed (see e.g. [1–3, 6, 10]). The line plan model presented by Borndörfer, Grötschel and Pfetsch [1, 2] minimizes the combination of line costs and system travel time disregarding transfers between lines and waiting times. To make the model feasible for real instances of local traffic systems, the line frequency variables are not forced to be integer, however knowing that the fractual solution can be quite far from the integer optimum. Because in general the minimum system travel time (called Beckmann-User-Equality) do not really reflect the passenger's "selfish" behavior, the use of the system travel time seems to be disadvantageous.

Both facts are respected in the model of Schöbel and Scholl [11]. This line plan model minimizes the passengers inconvenience under the restriction of a fixed budget for all line costs, whereas the passengers inconvenience is the sum of the travel time and the time needed for transfers concerning one origin-destination relation. The model in [11] assumes that the given passenger travel demand must be satisfied by the line plan. For all origin-destination pairs the passengers will

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1589

travel on a time shortest path. In the model presented here not all passenger demand must be covered. This is controlled by a limited budget for the operational cost of the line plan. For larger networks the model of Schöbel and Scholl suffers from large memory requirements for solving the shortest path problem in the *change&go network*, which contains for each line and each pair of consecutive served stations a special travel edge. Large number of lines and stations requires a big amount of memory. We try to overcome this problem by the use of a column generation scheme. The resulting pricing problem for those passenger flow variables is a shortest path problem.

2 The Basic Combined Model

2.1 General notations

The proposed model is restricted to the case to find a line plan for one homogeneous transport carrier. Each of the edges $e \in E$ of the underlying network (V, E) is assigned with a homogeneous travel time t_e. The nodes or vertices $n \in V$ of the network represent stations. Terminal stations are nodes, at which a line is allowed to start or to terminate. By a potential line L we will understand a path between two terminal nodes, which running time is in maximum the product of the detour factor ρ and the shortest travel time between the two terminal nodes. For simplicity we use the notation $e \in L$ to indicate, that the line L uses edge e. By $L_{|(A,B)}$ we denote the set of all edges, which are used by line L running from station A to station B. This set might be empty, if A and B are not served by L.

2.2 Traffic Assignment

The traffic assignment problem is modelled by some kind of multi-commodity flow problem, where we consider for each travel demand pair (O, D) and lines $L \in \mathcal{L}$ partial passenger routes $p^{O,D}_{(A,B),L}$. The variables

$$\varphi^{O,D}_{(A,B),L}$$

define the traffic flow of all OD-passengers using line L between station A and station B.

A total passenger route p from the origin node O to the destination node D is simply the concatenation of partial routes (see figure 1). The disutility of impedance of a passenger route is a measure for the inconvenience for the trip by this route. In traffic assignment theory this impedance is modelled in dependence of the travel time, the number of changings and the frequency of the service. In practice, there are used rather complex and nonlinear models for this function (see e.g. [4, 7]). Here, we simplify the model by using a linear approximation: For a path p denote the travel time by $t(p)$ and define the number of line changings by $c(p)$, which are penalized by the parameter β. Then $imp(p) := t(p) + \beta c(p)$ is a reasonable measure for the disutility of using the trip route p.

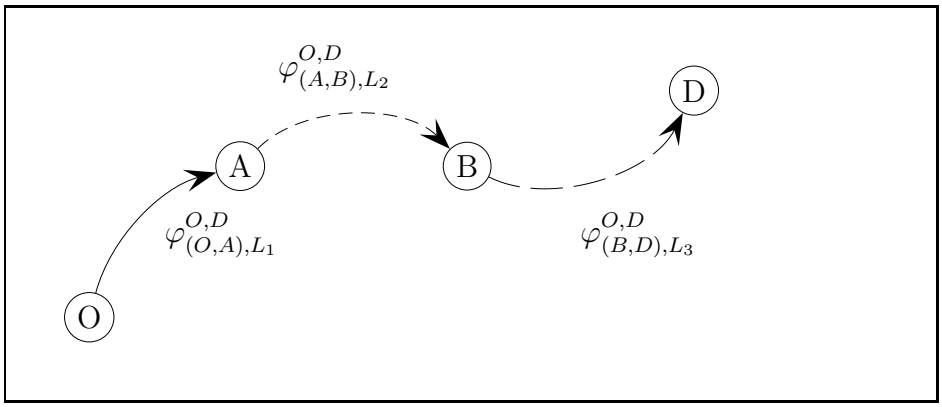


Fig. 1. Partial passenger route flow variables

In order to keep the resulting model linear, we do **not** use the common approach ([4]) to measure quality or utility by a function $f(p) \sim \frac{1}{imp(p)}$, but define the *travel quality* by

$$\omega(p) = \rho \cdot t^*_{OD} - t(p) - \beta c(p) \tag{1}$$

t^*_{OD} denotes the minimum possible travel time from origin O to destination D. The detour factor $\rho \geq 1$ should be defined in such a way, that passengers will accept all those routes for which the travel time added with the change penalty is at most a $\rho-$ multiple of the minimum travel time. The quality measure $\omega(p)$ can be interpreted as that time, what a passenger can save by using a connection p compared to the maximal accepted travel time for that OD-relation.

The best quality is given by $t^*_{OD} \cdot (\rho - 1)$. If the travel time or/and the number of necessary changings becomes too large, the quality turns out to be negative. If there exists no alternative, more favourable path with positive quality, we will assume to 'loose' those passengers. Figure 2 illustrates this approach.

In order to split the total travel time onto the line parts, we define $t_{L;(A,B)}$ to be the travel time of line L from station A to station B. The quality weights

$$\omega^{O,D}_{(A,B),L} := \begin{cases} -t_{L;(A,B)}, & if A = O \text{ and } B \neq D \\ -t_{L;(A,B)} - \beta, & if A \neq O \text{ and } B \neq D \\ \rho t^*_{OD} - t_{L;(A,B)} - \beta, & if A \neq O \text{ and } B = D \\ \rho t^*_{OD} - t_{L;(A,B)}, & if A = O \text{ and } B = D \end{cases}$$

obviously have the property, that its sum on each total path from origin node O to destination node D equals with (1) (see Figure 3).

Combining flow conservation laws and travel demand leads to the traffic assignment model:

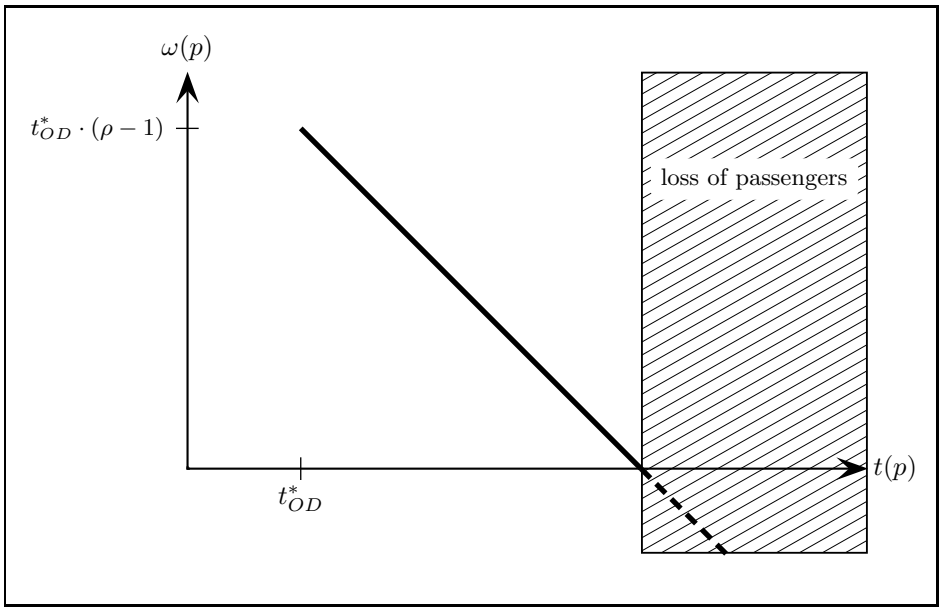


Fig. 2. Quality measure for passenger routes.

Traffic Assignment Model

$$\sum_{O,D,A,B,L} \omega^{O,D}_{(A,B),L} \varphi^{O,D}_{(A,B),L} \rightarrow max$$

$$\sum_{L,A} \varphi^{O,D}_{(A,D),L} \leq v_{OD}$$

$$\sum_{L,A} \varphi^{O,D}_{(A,X),L} - \sum_{L,B} \varphi^{O,D}_{(X,B),L} = 0 \quad \text{forall } X \notin \{O,D\}$$

$$\varphi^{O,D}_{(A,B),L} \in \mathbb{R}^+$$

Note that this model **maximizes travel quality**.

2.3 Line planning model

For the line planning part of our model we use the basic linear programming approach of M. Bussieck ([3]), who defines for each line L a frequency variable $x_L \in \mathbb{Z}^+$, which gives the service frequency of the line per time unit, say one hour. $x_L = 0$ implies, that L is not considered in the line plan. Values $x_L = 1, 2, ...$ indicate, that this line is served once, twice,... per time period. Since we consider a homogeneous line system, we may assume that the passenger capacity of all

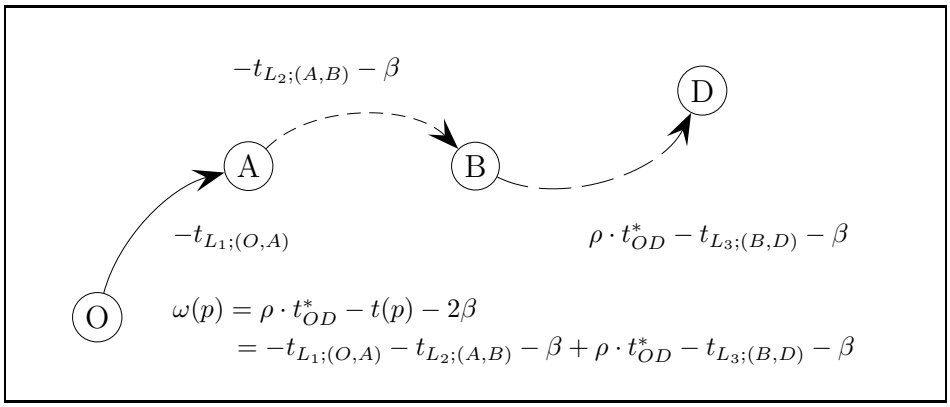


Fig. 3. Splitting route quality to the partial passenger route flows.

single trains is defined by a constant value C. Infrastructural capacity constraints for edges and nodes can be simply modelled by the restrictions

$$\sum_{e \in L} x_L \leq \overline{f}(e) \qquad (\mu_e)$$

$$\sum_{L \text{ serves node } n} x_L \leq \overline{f}(n) \qquad (\nu_n)$$

To model operational cost, we use a simple, linear approach and define the costs for serving line L with frequency x_L, by $c_L x_L$. Operational costs are discussed in more detail by K.-F. Jerosch (see [5]).

2.4 Combining traffic assignment and line planning

Traffic assignment and line planning are combined by the capacity constraints

$$\sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L} \leq C x_L$$

which guarantees for each edge e that the line plan provides enough capacity for the passengers. The right side of this inequality is the amount of available capacity provided by the line L with frequency x_L. The left hand summarizes the passengers from all origin-destination pairs (O, D) and all route parts (A, B), which are using the line L on edge e. The modelling approach allows two different possibilities to treat operational cost:

1. Operational cost may be placed into the objective

$$\max \sum_{O,D,A,B,L} \omega^{O,D}_{(A,B),L} \varphi^{O,D}_{(A,B),L} - \alpha \left(\sum_L c_L x_L \right)$$

2. Operational may be restricted by a given budget, which leads to the constraint

$$\sum_L c_L x_L \leq c^{max}$$

In summary, we obtain the full model

Full model

$$\begin{aligned}
\sum_{O,D,A,B,L} \omega^{O,D}_{(A,B),L} \varphi^{O,D}_{(A,B),L} &\to max \\
\sum_{L,A} \varphi^{O,D}_{(A,D),L} &\leq v_{OD} \\
\forall X \notin \{O,D\} : \sum_{L,A} \varphi^{O,D}_{(A,X),L} - \sum_{L,B} \varphi^{O,D}_{(X,B),L} &= 0 \\
\forall e \in E, L \in \mathcal{L} : \sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L} &\leq C x_L \\
\sum_L c_L x_L &\leq c^{max} \\
\sum_{e \in L} x_L &\leq \overline{f}(e) \\
\sum_{L \text{ serves node } n} x_L &\leq \overline{f}(n) \\
\varphi^{O,D}_{(A,B),L} &\in \mathbb{R}^+ \\
x_L &\in \mathbb{N}
\end{aligned}$$

2.5 Model strengthening by cutting planes

Using standard techniques from integer programming (see e.g. [8]), it is easy to see that

$$\varphi - \left(b - C\left\lfloor \frac{b}{C} \right\rfloor\right) x \leq \left[C - \left(b - C\left\lfloor \frac{b}{C} \right\rfloor\right)\right] \left\lfloor \frac{b}{C} \right\rfloor \tag{2}$$

is a valid inequality of the polyhedron

$$P = \{(\varphi, x) \in \mathbb{R} \times \mathbb{Z} \mid \varphi - Kx \leq 0;\ \varphi \leq b\}$$

Due to limited travel demand, the passenger flows

$$\varphi^{max}_e := \sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L}$$

are bounded from above. Reasonable bounds can be calculated from the origin-destination matrix and the detour factor ρ. Such bounds are discussed by M. Bussieck ([3]) and can be calculated in polynomial time. Define 'reduced' capacity by

$$\tilde{C}_e := \left(\varphi_e^{max} - C \left\lfloor \frac{\varphi_e^{max}}{C} \right\rfloor \right)$$

and the residual capacity by

$$C_e^* := \left[C - \left(\varphi_e^{max} - C \left\lfloor \frac{\varphi_e^{max}}{C} \right\rfloor \right) \right] \left\lfloor \frac{\varphi_e^{max}}{C} \right\rfloor,$$

then by (2) the valid inequality

$$\sum_e \sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi_{(A,B),L}^{O,D} - C \left(\sum_{L:e \in L} x_L \right) \leq 0 \tag{3}$$

implies the cutting plane

$$\sum_e \sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi_{(A,B),L}^{O,D} - \tilde{C}_e \left(\sum_{L:e \in L} x_L \right) \leq C_e^* \tag{4}$$

For the case, that $\varphi_e^{max} < C$, we have $C_e^* = 0$ and $\tilde{C}_e \leq C$. Then we may replace the inequality (3) by the more tighten inequality (4).

3 Solution Methods

3.1 Overview Solution Approach

In real world examples, there will arise a huge amount of possible passenger flow parts. Hence, those variables **can** only be handled by the use of a column generation scheme. For the potential lines (i.e. all frequency variables are contained in the model) the resulting pricing problem is quite straightforward and leads to a shortest path problem with non-negative arc weights. If the potential line frequency variables are not known a priori, the approach becomes much more complicated, since x_L and passenger route variables must be generated simultaneously.

The following algorithm summarizes the discussed method for the case that we use a fixed line pool. A more detailed description of the algorithm steps is given in the next sections.

Algorithm 1

1. *Calculate the pool of potential lines.*
2. *Calculate an initial line plan, which is feasible with the budget constraint.*

3. *Solve the relaxation of the initial linear model.*
4. *Pricing iteration:*
   (a) *solve the pricing problem for the passenger flow variables and add those with negative reduced cost to the formulation*
   (b) *resolve relaxation. Otherwise stop, the optimal solution has been found.*
   (c) *solve the knapsack problem (3.3) and the flow problem (3.3)*[1] *to calculate an upper bound of the problem.*
   (d) *in case of fractional variables $\tilde{x}_L$ apply the primal heuristic to find a possibly improved solution.*

3.2 Column Generation

In the following we consider the normalized model with associated dual variables given in brackets on the right side.

$$\begin{aligned}
\sum_{O,D,A,B,L} \omega^{O,D}_{(A,B),L}\varphi^{O,D}_{(A,B),L} &\to max \\
\sum_{L,A} \varphi^{O,D}_{(A,D),L} &\le v_{OD} \qquad (\pi^{OD}_D) \\
\forall X \notin \{O,D\} : \sum_{L,A} \varphi^{O,D}_{(A,X),L} - \sum_{L,B} \varphi^{O,D}_{(X,B),L} &= 0 \qquad (\pi^{OD}_X) \\
\forall e \in E, L \in \mathcal{L} : \sum_{O,D,A,B:e\in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L} - Cx_L &\le 0 \qquad (\xi_{e,L}) \\
\sum_{L} c_L x_L &\le c^{max} \qquad (\gamma) \\
\sum_{e\in L} x_L &\le \overline{f}(e) \qquad (\mu_e) \\
\sum_{L \text{ serves node } n} x_L &\le \overline{f}(n) \qquad (\nu_n) \\
\varphi^{O,D}_{(A,B),L} &\in \mathbb{R}^+ \\
x_L &\in \mathbb{N}
\end{aligned}$$

The dual *potential* variables π^{OD}_X of the underlying passenger route flow problem are only defined for $X \neq O$. For technical reasons we define $\pi^{OD}_O = 0$ [2]

[1] The solution requires no extra effort, because this problem is already solved during the pricing problem for the passenger flow variables.

[2] This allows to model the dual shortest path problem by regarding $\pi^{OD}_Y - \pi^{OD}_X$ to be a shortest path length from node X to node Y.

Pricing of partial travel routes In dependence on the nodes A, B define

$$s_{(A,B)} := \begin{cases} 0, & if A = O \text{ and } B \neq D \\ -\beta, & if A \neq O \text{ and } B \neq D \\ \rho t^*_{OD} - \beta, & if A \neq O \text{ and } B = D \\ \rho t^*_{OD}, & if A = O \text{ and } B = D \end{cases}$$

Then, the reduced cost of a partial travel route flow variable $\varphi^{O,D}_{(A,B),L}$ are given by

$$\begin{aligned} r\omega^{O,D}_{(A,B),L}) &= -\omega^{O,D}_{(A,B),L} + \sum_{e \in L_{|(A,B)}} \xi_{e,L} + \pi^{OD}_B - \pi^{OD}_A \\ &= -s_{(A,B)} + \sum_{e \in L_{|(A,B)}} t_e + \sum_{e \in L_{|(A,B)}} \xi_{e,L} + \pi^{OD}_B - \pi^{OD}_A \\ &= -s_{(A,B)} + \sum_{e \in L_{|(A,B)}} (\xi_{e,L} + t_e) + \pi^{OD}_B - \pi^{OD}_A \end{aligned}$$

In order to find flow variables with negative reduced cost, we have to minimize the generalized path length $\alpha^* := \min_L \sum_{e \in L_{|(A,B)}} (\xi_{e,L} + t_e)$ and compare this shortest path length by

$$r < 0 \iff \alpha^* < s_{(A,B)} + \pi^{OD}_B - \pi^{OD}_A$$

Note, that the right side $s_{(A,B)} + \pi^{OD}_B - \pi^{OD}_A$ is independently from line L.

In summary, pricing out partial passenger routes can be solved by a shortest path problem from node A to node B with respect to non-negative edge cost $\xi_{e,L} + t_e \geq 0$.

Pricing of line variables The reduced cost of a frequency line variable x_L is given by

$$r(x_L) = \gamma c_L - C \sum_{e \in L} \xi_{e,L} + \sum_{e \in L} \mu_e + \sum_{L \text{ serves node } n} \nu_n$$

Setting $\nu_e = \nu_n$ for $e = (n, x)$ and using a linear approximation

$$c_L = \sum_{e \in L} c_{e,L}$$

for the operational costs, we receive reduced cost as path length

$$\begin{aligned} r(x_L) &= \gamma c_L - C \sum_{e \in L} \xi_{e,L} + \sum_{e \in L} \mu_e + \sum_{L \text{ serves node } n} \nu_n \\ &= \sum_{e \in L} \left(\underbrace{\gamma c_{e,L} + \mu_e + \nu_e}_{\text{shadow cost}} - \underbrace{C\xi_{e,L}}_{\text{shadow profit}} \right) \end{aligned}$$

The dual variables of the arc cost split into

- a cost part $\gamma c_{e,L} + \mu_e + \nu_e$ which are shadow cost with respect to infrastructural capacity limits (μ and ν) and budget restriction $\gamma c_{e,L}$ and
- a negative signed gain part $C\xi_{e,L}$ which may be interpreted as some kind of income in the context of the active line system. Large positive values of $\xi_{e,L}$ indicate mismatched supply and demand (large demand or small supply).

Hence, the negative value $-r(x_l) = \sum_{e \in L}(C\xi_{e,L}) - \sum_{e \in L}(\gamma c_{e,L} + \mu_e + \nu_e)$ can be interpreted to be some kind of profit = income - expenditures, which should be maximized during the pricing step.

Theorem Line pricing
Line pricing is a longest path problem, i.e. to find an elementary (cycle free) path with maximum 'profit'

$$\delta_e := \underbrace{C\xi_{e,L}}_{\text{shadow profit}} \underbrace{- \gamma c_{e,L} + \mu_e + \nu_e}_{\text{shadow cost}}$$

3.3 Lagrange Relaxation

Lagrangian relaxation is a standard technique which moves hard constraints into the objective. Traffic assignment and line planning model are only coupled by the constraints

$$\sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L} \leq Cx_L \iff \sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L} - Cx_L \leq 0$$

By using the associated dual variables $\xi_{e,L}$, the movement of those constraints leads to the Lagrangian objective

$$\begin{aligned} &\sum_{O,D,A,B,L} \omega^{O,D}_{(A,B),L} \varphi^{O,D}_{(A,B),L} + \sum_{e,L} \xi_{e,L} \left[Cx_L - \sum_{O,D,A,B,L:e \in L} \varphi^{Q,Z}_{(A,B),L} \right] \\ = &\underbrace{\sum_{Q,Z,A,B,L} \left(\omega^{O,D}_{(A,B),L} - \sum_{L:e \in L} \xi_{e,L} \right) \varphi^{O,D}_{(A,B),L}}_{\text{flow part}} + \underbrace{C \sum_{e,L} \xi_{e,L} x_L}_{\text{line plan part}} , \end{aligned}$$

which separates into the sum of two linear terms depending either on the passenger flow variables or the line plan variables. Hence, solving the problems

FlowProblem

$$\varphi^* := \sum_{O,D,A,B,L} \left(\omega^{O,D}_{(A,B),L} - \sum_{O,D,A,B,L:e\in L} \xi_{e,L} \right) \varphi^{O,D}_{(A,B),L} \to \max$$

$$\sum_{L,A} \varphi^{O,D}_{(A,D),L} \leq v_{OD}$$

$$\forall X \notin \{O,D\} : \sum_{L,A} \varphi^{O,D}_{(A,X),L} - \sum_{L,B} \varphi^{O,D}_{(X,B),L} = 0$$

$$\varphi^{Q,Z}_{(A,B),L} \in \mathbb{R}^+$$

Generalized Knapsack Problem

$$\xi^* := C \sum_{e,L} \xi_{e,L} x_L \to \max$$

$$\sum_{L} c_L x_L \leq c^{max}$$

$$\sum_{e\in L} x_L \leq \overline{f}(e)$$

$$\sum_{L \text{ serves node } n} x_L \leq \overline{f}(n)$$

$$x_L \in \mathbb{N}$$

leads to the upper bound $\varphi^* + \xi^*$.

Any integer solution $(\tilde{x}_L)_{L\in\mathcal{L}}$ of the generalized knapsack problem leads to feasible line plan of the origin problem and can therefore be used to generate a primal feasible solution by solving the traffic assignment problem.

Primal Heuristic Traffic Assignment

$$\sum_{O,D,A,B,L} \omega^{O,D}_{(A,B),L} \varphi^{O,D}_{(A,B),L} \rightarrow max$$

$$\sum_{L,A} \varphi^{O,D}_{(A,D),L} \leq v_{OD}$$

$$\sum_{L,A} \varphi^{O,D}_{(A,X),L} - \sum_{L,B} \varphi^{O,D}_{(X,B),L} = 0 \quad \text{forall } X \notin \{O,D\}$$

$$\forall e \in E, L \in \mathcal{L}: \sum_{O,D,A,B:e \in L_{|(A,B)}} \varphi^{O,D}_{(A,B),L} \leq C\tilde{x}_L$$

$$\varphi^{O,D}_{(A,B),L} \in \mathbb{R}^+$$

4 Computational Results

4.1 The Traffic Sample

The software tool LINOP, developed by the Technical University Dresden, Faculty of Traffic Sciences, Institute for Logistics and Aviation, includes the presented mathematical model. Linear programs are solved by using the COIN-BCP-Solver.

We applied our method to the bus network of Berlin city. By kindly support of the Berlin city authority of city development and transportation we used origin destination data, which are not allowed to be published. For this reason the computational results do not report the total passenger demand. Instead, the results are given in percentage of optimal travel quality. 100% percent means, that for each origin destination pair a direct connection could be provided by the line plan.

Using an origin-destination matrix of passenger demands including all stops of Berlin city, it was necessary to add tramway, subway and the local train network to the existing bus infrastructure. A system split (see [9]) splits the total origin destination matrix into separate matrices for the tramway, subway, local train, express bus and bus.

4.2 Instance 1: Bus network of Berlin City

The modeled infrastructure consists of 2.590 nodes and 3.200 edges represents the Berlin city public bus network of July 2005. Bus stops are defined as nodes and edges describe possible connections beetween two bus stops. There are three different kinds of bus services operating on the Berlin infrastructure network, called express bus, metro bus and local bus. The bus services are different considering operation time, frequency or travel time. For instance the expressbus

Fig. 4. Resulting bus line plan

stops not as often as a local bus and consequently the expressbus has a reduced travel time. Due to large differences of the travel times we defined the expressbus as independent transportation product in our model. However the metro bus and the local bus are very similar and consequently these bus services are combined in the model as one transport carrier.

Table 1. *characteristics of one bus line plan*

parameter	value
minimum length of a line:	10 minutes travel time
budget:	7755 minutes operation time
changing penalty:	1 min/changing
detour factor:	1.1
number of lines:	119 lines
lines with frequency of 10 minutes:	33 lines
lines with frequency of 20 minutes:	86 lines
optimization objective value (travel quality):	914.354

For the most challenging instance, what is the expressbus and bus network, we performed for different budget parameter up to 3 pricing iterations, (see table 1), each of which took approximately 40 minutes. The instance was computed by an UNIX-PC with 2x3,2 Ghz and 8 gigabyte RAM. Due to the huge amount

of memory needed (Out of memory!), for the bus network only a few number of iterations could be performed.

A typical solution is reported by the Figure 4 and Table 1.

4.3 Instance 2: Tram network of Berlin city

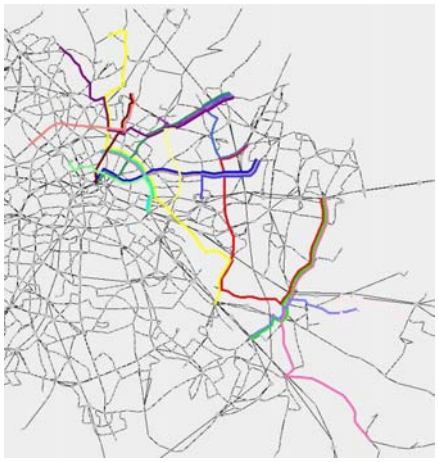


Fig. 5. tramway infrastructure

The second instance is the tram network of Berlin city. Which is of much smaller size compared to the bus network. The initial heuristic solution has an objective of 66.20, which could be improved during the iterations up to a travel quality of 71.00. By Langrange Relaxation we obtained the best upper bound by 89.45.The characteristics are presented in table 2 and in figure 6.

Table 2. *characteristics of one tramway line plan*

parameter	value
nodes	382
edges	435
minimum length of a line:	20 minutes travel time
budget:	1905 minutes operation time
changing penalty:	5 min/changing
detour factor:	1.2
number of lines:	18 lines
lines with frequency of 10 minutes:	18 lines
optimization objective value (travel quality):	71.000

Figure 6 illustrates the performance of the algorithm. Please note that the first line plan represents the start heuristic solution. Initialisation and preprocessing took approximately 10 minutes. The best solution with our hardware-performance for the tram network is found after 12 minutes. The tram line plan was solved with the same hardware as the bus network. In contrast to the bus network for this smaller instance the route node of the branch and bound tree was priced out rather quickly. But after a branch and bound depth of 120 the computer run out of memory again.

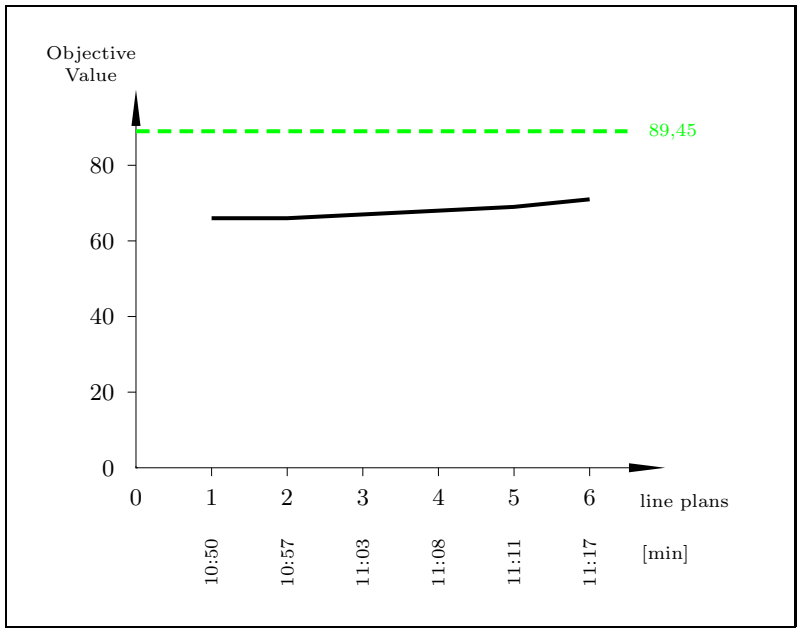


Fig. 6. *objective value (example tram line plan of Berlin city)*

References

1. Borndörfer, R., Grötschel, M. , Pfetsch, M. E.: Models for Line Planning in Public Transport. Research report 04-10, ZIB, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Takusstr. 7, 14195 Berlin-Dahlem (2004)
2. Borndörfer, R., Grötschel, M. , Pfetsch, M. E.: A Path-Based Model for Line Planning in Public Transport. Research report 05-18, ZIB, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Takusstr. 7, 14195 Berlin-Dahlem (2005)
3. Bussieck, M. R.: Optimal Lines in Public Rail Transport. PhD thesis, Technical University Braunschweig (1998)

4. Dugge, B.: Ein simultanes Erzeugungs-, Verteilungs- und Routenwahlmodell (EVA-U), Phd thesis, Technical University Dresden, Institute for Traffic Planning and Road Traffic, ISSN 1432-5500 (2006)
5. Jerosch, K.-F.: Evaluierung und Weiterentwicklung eines Modells zur Liniennetzoptimierung. Diploma thesis, Technical University Dresden (2006)
6. Klier, M. J., Hasse, Knut: Line Optimization in Public Transport Systems. Operations research proceedings, Saarbrücken (2007)
7. Lohse, D., Schnabel, W.: Grundlagen der Strassenverkehrstechnik und der Verkehrsplanung. Verlag für Bauwesen, Berlin (1997)
8. Nemhauser, G. L., Wolsey, L. A.: Integer and Combinatorical Optimization. J. Wiley and Sons, New York (1972)
9. Oltrogge, C.: Linienplanung für mehrstufige Bedienungssysteme im öffentlichem Personenverkehr. PhD thesis, Technical University Braunschweig (1993)
10. Scholl, S.: Customer-Oriented Line Planning. PhD thesis, Technical University Kaiserslautern (2005)
11. Schoebel, A., Scholl, S.: Line Planning with Minimal Traveling Time. reprint-serie, Institut for Numerical and Applied Mathematics, Technical University Göttingen (2005)

Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations

Karl Nachtigall and Jens Opitz

Technical University Dresden, Faculty of Traffic Sciences,
Institute for Logistics and Aviation
karl.nachtigall@tu-dresden.de
jens.opitz@tu-dresden.de

Abstract. In the last 15 years periodic timetable problems have found much interest in the combinatorial optimization community. We will focus on the optimisation task to minimise a weighted sum of undesirable slack times. This problem can be formulated as a mixed integer linear problem, which for real world instances is hard to solve. This is mainly caused by the integer variables, the so-called modulo parameter. At first we will discuss some results on the polyhedral structure of the periodic timetable problem. These ideas allow to define a modulo simplex basic solution by calculating the basic variables from modulo equations. This leads to a modulo network simplex method, which iteratively improves the solution by changing the simplex basis.

Key words: periodic event scheduling problem, integer programming, modulo network simplex

1 Introduction

In the last 15 years periodic timetable problems have found much interest in the combinatorial optimization community. Most results presented in [6, 10, 3, 4, 7, 8, 2] are based on a periodic event scheduling model published by Serafini and Ukovich 1989 ([11]).

The associated periodic event activity networks allow a flexible modelling of fixed interval timetables in public transport. A lot of practical requirements, like sequencing of trains, safety headway distances and limits for rolling stock can be incorporated into this network theoretical model. In this paper we will focus on the optimisation task to minimise a weighted sum of undesirable slack times, e.g., waiting time for passengers.

Define a *railway system* as a system of lines $\mathcal{L}$ and stations $\mathcal{S}$. Each line $L \in \mathcal{L}$ is understood to be a *transportation chain*, where the trains of L are serving a certain sequence of stations (see e.g. [12]). If line L serves stations S, then define (L, arr, S) and (L, dep, S) to be the arrival (departure) event of L at S.

A *schedule* assigns *event times* $\pi_i \in \mathbb{R}$ to all events $i = (L, dep, S)$ or $i = (L, arr, S)$. An activity $a : i \to j$ is a time consuming process, which then will consume the amount $x_a := \pi_j - \pi_i$. of time. A line can be understood as an alternating sequence of

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1588

– run activities : $(L, dep, S) \to (L, arr, S')$ and
– stop activities : $(L, arr, S) \to (L, dep, S)$.

Run and stop activities are associated with time spans $\Delta_a = [\ell_a, u_a]$, where ℓ_a is the minimum running or stopping time and u_a is an upper bound. [1] A schedule $\boldsymbol{\pi}$ is said to be *feasible*, if $x_a = \pi_j - \pi_i \in \Delta_a$ for all $a : i \to j$. Apart from running and stopping activities, in real world problems there are many other types of constraints arising from operational, safety- or marketing-related restrictions. Almost all practical requirements can be formulated in terms of span constraints $\ell_a \leq \pi_j - \pi_i \leq u_a$ defined on a suitable arc $a : i \to j$ of the event network. Some examples are:

– *Headway constraints:* Trains using the same parts of the infrastructure have to keep a certain safety distance. This distance can be expressed as a time difference between the arrival or departure times of the lines at the stations.
– *Traveller connection constraints:* In general, there are some stations where travellers have to change from one train to another. In this case, these travellers would like to have a short waiting time at the station. Again, this constraint is a time difference constraint between arrival and departure times of lines.

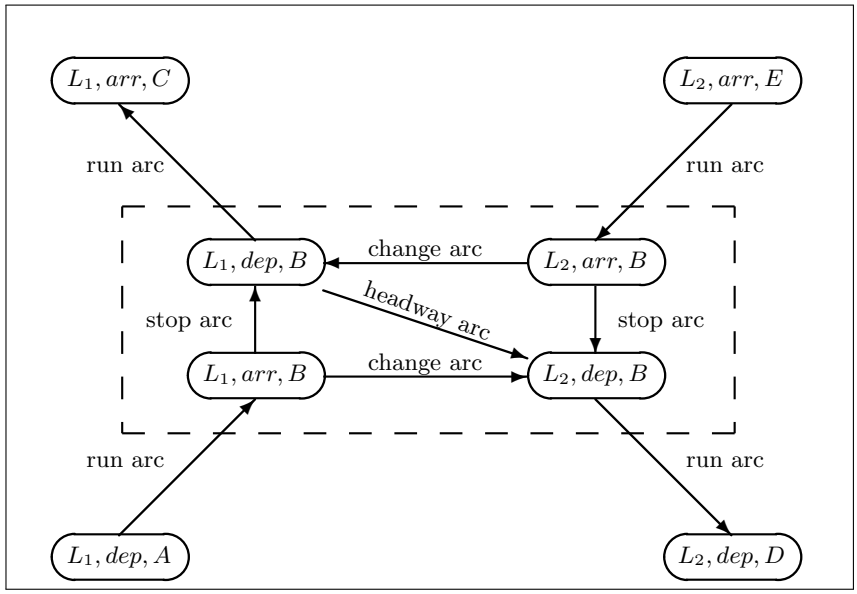


Fig. 1. Event-activity network

[1] If the running time is fixed, a running activity and the following stop activity can be simply described by one combined constraint.

Non-periodic timetable problems are very easy to solve by shortest path calculations. For fixed interval timetables, where all departure and arrival events will be repeated periodically, such a simple model is no more appropriate. The reasons are manifold: A priori it is not clear between which trains passengers are changing or in which sequence trains are leaving or entering stations. All this can only be decided after the time ordering of all events is known. A *periodic schedule* assigns periodic *event times* $\pi_i \in \mathbb{R}$ to all events, which will take place at all time points $\pi_i + zT\,(z \in \mathbb{Z})$. The integer multiples z of the period are called *modulo parameter* and code the periodic sequence of all events.

For reasons of simplicity we assume one common period T for the complete system. Different periods for the lines can be handled by using the least common multiple (compare for [5]).

A solution of the *periodic* timetable problem is defined by a vector $\boldsymbol{\pi} \in \mathbb{R}^n$, which defines for each event i one point of time π_i, such that i will be periodically repeated at all times $\pi_i + z_i T\,(z_i \in \mathbb{Z})$.

Define ℓ_a and u_a to be the minimum and maximum allowed process times of a constraint $a : i \to j$. Then a periodic timetable $\boldsymbol{\pi}$ is feasible, if

$$\forall a : i \to j \in \mathcal{A} : \exists z_a \in \mathbb{Z} : \ \ell_a \leq \pi_j - \pi_i - z_a T \leq u_a. \tag{1}$$

Lower and upper slack time measures that amount of time for which the tension $\pi_j - \pi_i$ on this arc may be increased or decreased and is defined by

$$\begin{aligned} y_a^{low} &:= [x_a - \ell_a]_T = x_a - \ell_a - z_a T \text{ for a suitable } z_a \in \mathbb{Z} \\ y_a^{upp} &:= [u_a - x_a]_T = u_a - x_a + z_a T \text{ for a suitable } z_a \in \mathbb{Z}. \end{aligned}$$

The modulo operator is defined by $[t]_T := \min\{\, t + zT \mid t + zT \geq 0\}$ and fulfills $0 \leq [t]_T < T$.

Since lower and upper slack times may be exchanged by inverting the direction of the arc a, the problem to minimize the slack time in a periodic timetable can be defined in terms of lower slack time y_a^{low}. In summary, the **periodic timetable slack problem** can be formulated as the mixed integer program

$$\min \left\{ \sum_{a:i\to j} \boldsymbol{\omega}_a(\pi_j - \pi_i - \ell_a - z_a T) \;\middle|\; \forall a \in \mathcal{A} : \ \ell_a \leq \pi_j - \pi_i - z_a T \leq u_a;\ z_a \in \mathbb{Z} \right\} \tag{2}$$

The resulting planning problems are known to be NP-hard.

2 The Periodic Timetable Polyhedron

At first we will briefly summarize the basic concepts and notations of network flow models.

The incidence matrix of a network is an $n \times m$ matrix $\Theta = (\theta_{ia})$ which contains one row for each arc a and one column i for each node:

$$\theta_{ai} = \begin{cases} 1, & \text{if } a : j \to i \\ -1, & \text{if } a : i \to j \\ 0, & \text{else} \end{cases}$$

A potential $\boldsymbol{\pi} \in \mathbb{R}^n$ associates with each node $i = 1, .., n$ a real value $\pi_i \in \mathbb{R}$.

$$\mathcal{Q} := conv.hull\left(\left\{ \begin{pmatrix} \boldsymbol{\pi} \\ \boldsymbol{z} \end{pmatrix} \;\middle|\; \boldsymbol{\ell} \le \Theta^t \boldsymbol{\pi} - T\boldsymbol{z} \le \boldsymbol{u};\ \boldsymbol{z} \in \mathbb{Z}^m; \boldsymbol{\pi} \in \mathbb{R}^n \right\}\right)$$

is said to be the periodic timetable polyhedron.

The potential difference $x_a := \pi_j - \pi_i$ is said to be the tension on arc $a : i \to j$ and can be expressed as $\Theta^t \boldsymbol{\pi} = \boldsymbol{x}$. Adding a co-tree arc a to the arcs of a spanning tree $\mathcal{T}$, defines a uniquely determined cycle c. Its incidence vector $\boldsymbol{\gamma}_c = (\gamma_{ca})$ is defined by

$$\gamma_{ca} := \begin{cases} 1 & \text{, if the cycle contains arc } a \text{ in positive direction} \\ -1 & \text{, if the cycle contains arc } a \text{ in negative direction} \\ 0 & \text{, else.} \end{cases}$$

The network matrix $\Gamma = (\gamma_{ca})$ of a tree $\mathcal{T}$ contains for each co-tree arc the incidence vector of the associated cycle as one row. $\boldsymbol{x} \in \mathbb{R}^m$ is a tension (i.e. there exists a potential $\boldsymbol{\pi} \in \mathbb{R}^n$ with $\Theta^t \boldsymbol{\pi} = \boldsymbol{x}$), if and only if there holds $\Gamma \boldsymbol{x} = \boldsymbol{0}$. A *periodic tension* $\boldsymbol{x}$ fulfils $\Gamma \boldsymbol{x} \equiv_T \boldsymbol{0}$.

A spanning tree structure $\mathcal{T} = \mathcal{T}^\ell + \mathcal{T}^u$ is a spanning tree, whose tree arcs are partitioned into those arcs $\mathcal{T}^\ell$ and $\mathcal{T}^u$, where the tension is restricted to be at its lower or upper bound, respectively [2]. Each spanning tree structure determines a unique potential $\boldsymbol{\pi}^{(\mathcal{T})}$, which fullfills $\pi_j^{(\mathcal{T})} - \pi_i^{(\mathcal{T})} = \ell_a$ for $(a : i \to j) \in \mathcal{T}^\ell$ and $\pi_j^{(\mathcal{T})} - \pi_i^{(\mathcal{T})} = u_a$ for $(a : i \to j) \in \mathcal{T}^u$. The spanning tree structure is said to be *feasible*, if the generated potential is feasible with respect to the span constraints for all arcs (1).

By using $\boldsymbol{b} :\equiv_T -\Gamma \boldsymbol{\ell}$ and $\boldsymbol{\delta} := \boldsymbol{u} - \boldsymbol{\ell}$, the periodic slack space is defined by

$$\mathcal{Y} := \{ \boldsymbol{y} \in \mathbb{Z}^m \mid \Gamma \boldsymbol{y} \equiv_T \boldsymbol{b}; \boldsymbol{0} \le \boldsymbol{y} \le \boldsymbol{\delta} \}$$

and the optimisation task is to determine $\min \{ \boldsymbol{\omega}^t \boldsymbol{y} \mid \boldsymbol{y} \in \mathcal{Y} \}$.

If the modulo parameters z_a are fixed, optimisation problem (2) becomes

$$\min \left\{ \sum_{a:i\to j} \boldsymbol{\omega}_a (\pi_j - \pi_i - \ell_a - z_a T) \;\middle|\; \forall a \in \mathcal{A} :\ \ell_a \le \pi_j - \pi_i - z_a T \le u_a \right\}$$

[2] This definition differs from that definition given in [1]. This is caused by the circumstances, that the dual timetable problem is a modified minimum cost flow problem **without** capacity on the arc flow values.

$$= \min \left\{ \sum_{a:i\to j} \boldsymbol{\omega}_a(\pi_j - \pi_i - \ell'_a) \;\middle|\; \forall a \in \mathcal{A}:\ \ell'_a = \ell_a + z_a T \le \pi_j - \pi_i \le u'_a = u_a + z_a T \right\}$$
$$= \min \left\{ \boldsymbol{\omega}^t \left(\Theta^t \boldsymbol{\pi} - \boldsymbol{\ell}' \right) \mid \boldsymbol{\ell}' \le \Theta^t \boldsymbol{\pi} \le \boldsymbol{u}' \right\} \tag{3}$$

the dual of a minimum cost flow problem (see [1]). The extreme points of the feasible region of this problem are associated with spanning tree structures. The network simplex method described in [1] interprets the core concept of the simplex method appropriately as network operations. In particular, each optimal basis can be characterized by the underlying spanning tree structure.

If $\boldsymbol{z}^{\mathcal{T}}$ denotes the associated modulo parameter, then $\begin{pmatrix} \boldsymbol{\pi}^{(\mathcal{T})} \\ \boldsymbol{z}^{\mathcal{T}} \end{pmatrix}$ is called a periodic basic solution with respect to the spanning tree structure $\mathcal{T}$. The following theorem is due to [6].

Theorem 21 (Extreme Points and Spanning Tree Structures) *$\begin{pmatrix} \boldsymbol{\pi} \\ \boldsymbol{z} \end{pmatrix} \in \mathcal{Q}$ is an extremal point of $\mathcal{Q}$, if and only, if it is a periodic basis solution with respect to a spanning tree structure.*

The orthogonal complement of the tension space is known to be the space of all flows ([9]), i.e. it holds $\{ \boldsymbol{x} \mid \Gamma \boldsymbol{x} = \boldsymbol{0} \}^{\perp} = \{ \boldsymbol{\varphi} \mid \Theta \boldsymbol{\varphi} = \boldsymbol{0} \}$. The space of all periodic tensions is defined by

$$\mathcal{X} := \{ \boldsymbol{x} \in \mathbb{Z}^m \mid \Gamma \boldsymbol{x} \equiv_T \boldsymbol{0} \}$$

In the periodic case, we obtain

$$\{ \boldsymbol{x} \in \mathbb{Z}^m \mid \Gamma \boldsymbol{x} \equiv_T \boldsymbol{0} \}^{\perp_T} = \{ \boldsymbol{\varphi} \in \mathbb{Z}^m \mid \Theta \boldsymbol{\varphi} \equiv_T \boldsymbol{0} \} \tag{4}$$

The following structural characterization of valid inequalities is due to [4] and are discussed in more detail in [3].

Lemma 2.1 Let $\mathcal{Q} \neq \emptyset$. Then $\boldsymbol{\vartheta}^t \boldsymbol{\pi} - \boldsymbol{f}^t \boldsymbol{z} \ge r$ can only be a valid inequality for the polyhedron

$$\mathcal{Q} := \mathit{conv.hull} \left(\left\{ \begin{pmatrix} \boldsymbol{\pi} \\ \boldsymbol{z} \end{pmatrix} \;\middle|\; \boldsymbol{\ell} \le \Theta^t \boldsymbol{\pi} - T\boldsymbol{z} \le \boldsymbol{u};\ \boldsymbol{z} \in \mathbb{Z}^m; \boldsymbol{\pi} \in \mathbb{R}^n \right\} \right)$$

with $\boldsymbol{\vartheta}^t \boldsymbol{\pi}^{(0)} - \boldsymbol{f}^t \boldsymbol{z}^{(0)} = r$ for at least one $\begin{pmatrix} \boldsymbol{\pi}^{(0)} \\ \boldsymbol{z}^{(0)} \end{pmatrix} \in \mathcal{Q}$, if and only if $\boldsymbol{f}$ is a flow with balance $\boldsymbol{\vartheta}$, i.e. it holds $T\boldsymbol{\vartheta} = \Theta \boldsymbol{f}$ and

$$Tr = \min \left\{ \boldsymbol{f}^t \boldsymbol{x} \;\middle|\; x \in \mathcal{X} \right\}$$

■

Theorem 22 *There exists a matrix F, where each of its rows is a periodic tension (i. e. $\Theta F \equiv_T \boldsymbol{0}$) and a right hand side $\boldsymbol{r}$, such that*

$$\mathit{conv.hull} \left(\{ \boldsymbol{x} \in \mathbb{Z}^m \mid \exists \boldsymbol{z} \in \mathbb{Z}^m : \Gamma \boldsymbol{x} - T\boldsymbol{z} = \boldsymbol{0}; \boldsymbol{\ell} \le \boldsymbol{x} - T\boldsymbol{z} \le \boldsymbol{u} \} \right)$$

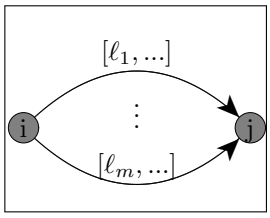
$$= \{\, \boldsymbol{x} \mid F\boldsymbol{x} \geq \boldsymbol{r} \}\,,$$

or equivalently

$$\begin{aligned} &\mathit{conv.hull}\left(\{\, \boldsymbol{y} \in \mathbb{Z}^m \mid \exists \boldsymbol{z} \in \mathbb{Z}^m : \Gamma\boldsymbol{y} - T\boldsymbol{z} = \boldsymbol{b}; \mathbf{0} \leq \boldsymbol{y} - T\boldsymbol{z} \leq \boldsymbol{\delta}\}\right) \\ &= \{\, \boldsymbol{y} \mid F\boldsymbol{y} \geq \tilde{\boldsymbol{r}} := \boldsymbol{r} - F\boldsymbol{\ell}\} \end{aligned}$$

■

An example for the construction of such inequalities is as follows. Consider a system of parallel arcs connecting two nodes i and j. The unbounded periodic timetable slack problem (without upper bounds on the arcs) deals with timetables from the set



$$P^*(\boldsymbol{\ell}) := \left\{ \begin{pmatrix} \pi_i \\ \pi_j \\ z_1 \\ \cdot \\ \cdot \\ z_m \end{pmatrix} \in \mathbb{Z}^{m+2} \;\middle|\; \forall a = 1, ...m : \ell_a \leq \pi_j - \pi_i - z_a T \right\}$$

Without loss of generality, a non degenerate[3] lower bound vector $\boldsymbol{\ell}$ can be assumed to be normalized in the sense, that

$$0 \leq \ell_1 = [\ell_1]_T < \ell_2 = [\ell_2]_T < ... < \ell_m = [\ell_m]_T < T \tag{5}$$

Lemma 2.2 For $\ell \in \mathbb{Z}^m$ with

$$0 \leq \ell_1 < \ell_2 < ... < \ell_m < T \tag{6}$$

define the vector $\boldsymbol{f}$ by

$$f_a := \begin{cases} \ell_1 - \ell_m + T & \text{if } a = 1 \\ \ell_a - \ell_{a-1} & \text{if } a{>}1 \end{cases} \tag{7}$$

Then there holds

1. $\forall a = 1, ..., m: \; 0 \leq f_a < T$
2. $\sum_{a=1}^{m} f_a = T$
3. $\forall a' : \sum_{a=a'+1}^{m} f_a = \ell_m - \ell_{a'}$

[3] $\boldsymbol{\ell}$ is called non degenerate, if $[\ell_a]_T \neq [\ell_{a'}]_T$ forall $a \neq a'$

Especially, $\boldsymbol{f}$ is a periodic flow with node mass balance $\vartheta_A = -T$ and $\vartheta_B = T$. The inequality

$$\pi_B - \pi_A - \boldsymbol{f}^t \boldsymbol{z} = \pi_B - \pi_A - \sum_{a=1}^{m} f_a z_a \geq f_0 := \ell_m \tag{8}$$

is a valid for $P^*(\boldsymbol{\ell})$. ■

Each of the considered arcs may be replaced by a chain of arcs, resulting in a system of paths between i and j. Consider a spanning tree. Then each tree arc $a : i \to j$ generates a cut and for each arc within this cut we find a path from i to j. Hence, there is a natural i, j path system, which can be used to generate cutting planes or equivalently rows of the matrix F.

3 The Modulo Simplex Method

For reasons of simplicity, in the following we only describe the case that the tension is restricted to be at its lower bound. This is no loss of generality, since upper bounds can be modelled as lower bounds on inverse directed arcs. Within the simplex method this means that the corresponding non-basic variable is set to the upper bound. Feasibility check and calculation of the modified cost of basis exchanges can be done straightforward.

We consider the periodic timetable slack problem

$$\min \left\{ \boldsymbol{\omega}^t \boldsymbol{y} \;\middle|\; \boldsymbol{y} \in \mathcal{Y} := \{ \boldsymbol{y} \in \mathbb{Z}^m \mid \exists \boldsymbol{z} \in \mathbb{Z}^m : \Gamma \boldsymbol{y} - T\boldsymbol{z} = \boldsymbol{b}; \mathbf{0} \leq \boldsymbol{y} - T\boldsymbol{z} \leq \boldsymbol{\delta} \} \right\}.$$

The integrality of the modulo parameter $\boldsymbol{z}$ makes the problem hard. For this reason we will eliminate those variables and keep them implicitly in the model by using modulo calculations. The modulo simplex method explores the extreme points of the polyhedron $conv.hull\,(\mathcal{Y})$.

The tree and co-tree arcs of the underlying spanning tree split the network matrix $\Gamma = [N_{\mathcal{T}}, E^{co}_{\mathcal{T}}]$ into its basic (= co-tree) and non-basic (= tree) components. Therefore a periodic basic solution is given by $\begin{pmatrix} \boldsymbol{y}_{\mathcal{T}} \\ \boldsymbol{y}^{co}_{\mathcal{T}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{b} \end{pmatrix}$, which is feasible if $\boldsymbol{b} \leq \boldsymbol{\delta}$. Any periodic tension $\boldsymbol{x}$ (with $\Gamma \boldsymbol{x} \equiv_T \mathbf{0}$) leads to a new solution $\boldsymbol{y}' := [\boldsymbol{y} + \boldsymbol{x}]_T = \boldsymbol{y} + \boldsymbol{x} - \boldsymbol{z}'T$ of $\Gamma \boldsymbol{y}' \equiv_T \boldsymbol{b}$ and stays feasible, if $\boldsymbol{y}' \leq \boldsymbol{\delta} := \boldsymbol{u} - \boldsymbol{\ell}$. In the following we will describe the problem by the use of a simplex tableau like structure. Consider the network matrix $\Gamma = [N, E]$ with respect to a spanning tree. The tree arcs are denoted by $a_1, ..., a_{r-1}$ and the co-tree arcs are given by $a_r, ..., a_m$. Then the slack space is given by the modulo equations

The resulting objective is given by

$$\boldsymbol{\omega} = \sum_{i=r}^{m} \boldsymbol{\omega}_i b_i$$

$$\begin{array}{l|l|l}
\gamma_{r1}y_1 + ... + \gamma_{r\,r-1}y_{r-1} & + y_r & \equiv_T b_r \\
\vdots \qquad\qquad \vdots & \quad \ddots & \vdots \\
\gamma_{i1}y_1 + ... + \gamma_{i\,r-1}y_{r-1} & + \quad y_i & \equiv_T b_i \\
\vdots \qquad\qquad \vdots & \quad \ddots & \vdots \\
\gamma_{m1}y_1 + ... + \gamma_{m\,r-1}y_{r-1} & + \quad y_m & \equiv_T b_m
\end{array}$$

	a_1 .... a_j ...	a_r ... a_i ... a_m	rhs
a_r	γ_{r1} ... γ_{rj} ...	1 ... 0 ... 0	b_r
$\vdots$	$\vdots$ ... $\vdots$	$\ddots$	$\vdots$
a_i	γ_{i1} ... $\boxed{\gamma_{ij}}$ ...	0 ... 1 ... 0	b_i
$\vdots$	$\vdots$ ... $\vdots$	$\ddots$	$\vdots$
a_m	γ_{m1} ... γ_{mj}	0 ... 0 ... 1	b_m
obj.			$\sum_{i=r}^{m} \omega_i b_i$

A basis exchange can be described by exchanging a leaving co-tree arc a_i with an entering tree arc a_j, which belongs to the uniquely determined co-tree cycle of the actual tree. The resulting cut $\boldsymbol{\eta}^{(a_i,a_j)}$ is given by adjoining the leaving tree component to the a_i-associated column of N. Each $\alpha \in \mathbb{Z}$ with $\alpha\boldsymbol{\eta}^{(a_i,a_j)} \leq \boldsymbol{\delta} = \boldsymbol{u} - \boldsymbol{\ell}$ defines by $\boldsymbol{y}' := \boldsymbol{y} + \alpha\boldsymbol{\eta}^{(a_i,a_j)}$ a new solution.

Exchanging co-tree arc i with tree arc j leads to the new solution

	a_1 .... a_i ...	a_r ... a_j ... a_n	rhs
a_r	$\gamma_{rl} - \frac{\gamma_{i1}\gamma_{rj}}{\gamma_{ij}}$ ... 0 ...	1 ... $-\frac{\gamma_{rj}}{\gamma_{ij}}$ ... 0	$\left[b_r - \frac{\gamma_{rj}}{\gamma_{ij}} b_i\right]_T$
$\vdots$	$\vdots$ ... $\vdots$	$\ddots$	$\vdots$
a_j	$\frac{\gamma_{i1}}{\gamma_{ij}}$ ... 1 ...	0 ... $\frac{1}{\gamma_{ij}}$ ... 0	$\left[\frac{b_i}{\gamma_{ij}}\right]_T$
$\vdots$	$\vdots$ ... $\vdots$	$\ddots$	$\vdots$
a_m	$\gamma_{m1} - \frac{\gamma_{i1}\gamma_{mj}}{\gamma_{ij}}$ ... 0	0 ... $-\frac{\gamma_{mj}}{\gamma_{ij}}$ ... 1	$\left[b_m - \frac{\gamma_{mj}}{\gamma_{ij}} b_i\right]_T$
obj.			$\tilde{\omega}_{ij} = \omega + \Delta\omega_{ij}$

.

The modified solution has cost

$$\tilde{\omega}_{ij} := \sum_{k=1}^{i-1} \omega_k \left[b_k - \frac{\gamma_{kj}}{\gamma_{ij}} b_i\right]_T + \omega_j \left[\frac{b_i}{\gamma_{ij}}\right]_T + \sum_{k=i+1}^{r} \omega_k \left[b_k - \frac{\gamma_{kj}}{\gamma_{ij}} b_i\right]_T$$

The cost difference can therefore be calculated by

$$\Delta\omega_{ij} = \tilde{\omega}_{ij} - \omega$$

$$= \sum_{k \neq i} \omega_k \left(b_k - \left[b_k - \frac{\gamma_{kj}}{\gamma_{ij}} b_i \right]_T \right) + \omega_i b_i - \omega_j \left[\frac{b_i}{\gamma_{ij}} \right]_T \tag{9}$$

The following example illustrates these considerations.

3.1 Example

Consider a problem with period $T = 20$ and underlying event network shown in Figure 2.

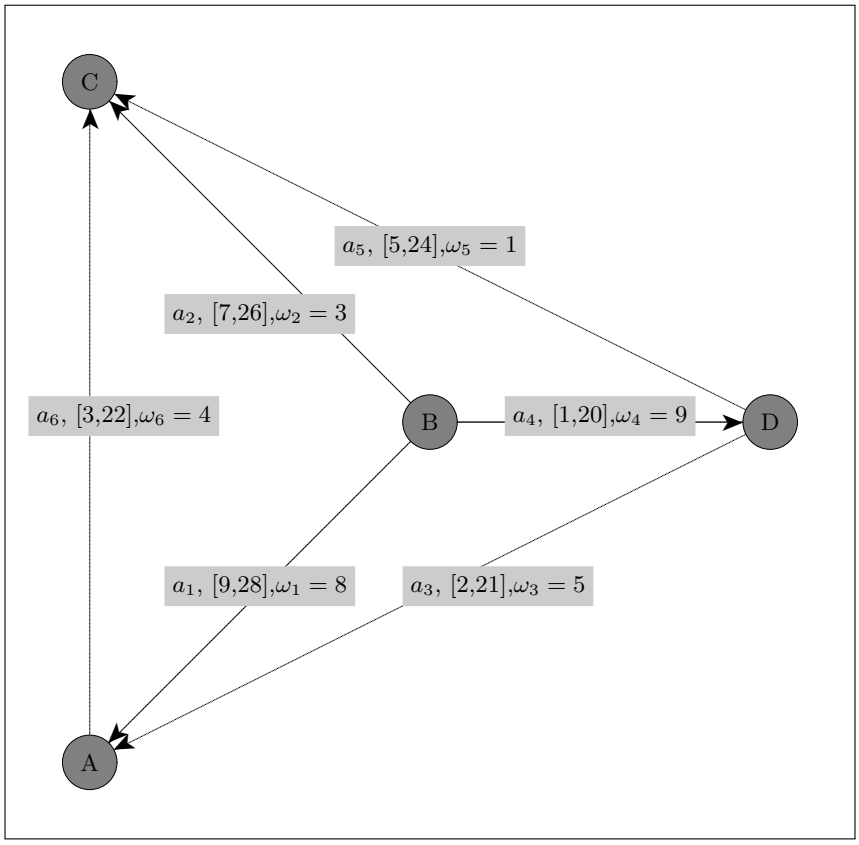


Fig. 2. Event Network

The initial spanning tree $\mathcal{T} = \mathcal{T}^\ell + \mathcal{T}^u$ with $\mathcal{T}^\ell = \{a_2, a_3, a_5\}$ and the resulting potential is given by Figure 3. This initial spanning tree structure induces the following *modulo simplex tableau* with total cost $\omega = 129$.
The following table contains for each possible basis exchange the resulting cost difference. This can be calculated by formula (9). The best gain will be received

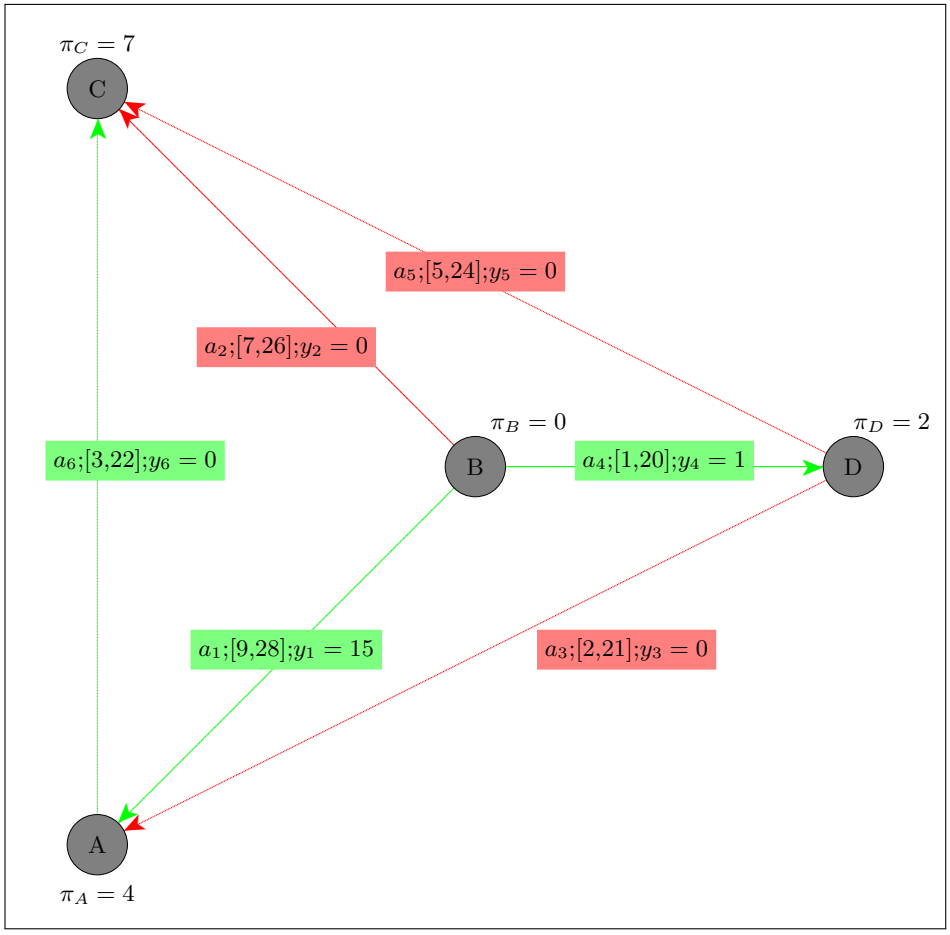


Fig. 3. Spanning tree structure (= red arcs), associated potential and slack times $y_a = [\pi_j - \pi_i - \ell_a]_T$.

by exchanging tree arc a_2 with co-tree arc a_1. Table 3 shows the new modulo simplex tableau after pivot step. The new objective is $\omega = 129 - 60 = 69$.

Table 3. Modulo Simplex Tableau of Step 1.

	a_1	a_3	a_5	a_4	a_2	a_6	b	ω
a_4	-1	1	0	1	0	0	6	9
a_2	-1	1	-1	0	1	0	5	3
a_6	0	1	-1	0	0	1	0	4
ω							69	

Table 1. Initial Modulo Simplex Tableau.

	a_2	a_3	a_5	a_4	a_1	a_6	b	ω
a_4	-1	0	1	1	0	0	1	9
a_1	-1	-1	1	0	1	0	15	8
a_6	0	1	-1	0	0	1	0	4
ω							129	

Table 2. Cost difference Δ for all possible basis exchanges.

	a_2	a_3	a_5
a_4	40	-	-12
a_1	$\boxed{-60}$	-35	0
a_6	-	0	20

The algorithm performs such modulo simplex pivot steps as long as a basis exchange will generate an improvement of the solution. Clearly, this only leads to a local minimum. Each periodic tension $\boldsymbol{\eta}$ with $\Gamma\boldsymbol{\eta} \equiv_T \mathbf{0}$ and $\boldsymbol{\eta} \leq \boldsymbol{\delta}$ defines by $\boldsymbol{y}' := \boldsymbol{y} + \boldsymbol{\eta}$ a new solution of the problem. It improves the old solution, if the new objective value gets better. In case of an improvement the modulo simplex pivoting will be applied again. This requires a basic solution, which can be simply received by solving the **non-periodic** minimum cost flow with fixed modulo parameter by the classical network simplex method.

In order to improve the local optimum after modulo simplex pivoting we apply a special class of cuts: For each node i the set of all leaving or entering arcs is a cut $\boldsymbol{\eta}^{(i)}$. Modifying the potential value of node i by $\pi_i' := \pi_i + \delta$, equals with the solution $\boldsymbol{y} + \delta\boldsymbol{\eta}^{(i)}$ after applying the $\delta-$multiple of the cut. For the class of those *single node cuts* it is obviously easy to check the improvement by enumerating all possible values for δ.

The modulo network simplex method can be summarized by

3.2 Modulo Network Simplex Algorithm

Initialisation: Determine an initial feasible tree structure $\mathcal{T} = \mathcal{T}^\ell + \mathcal{T}^u$ with feasible solution y

Single node improvement: WHILE (there exists an improving single node cut $\boldsymbol{\eta}$) **DO**

1. Apply this cut by transforming the solution $\boldsymbol{y'} := \boldsymbol{y} + \boldsymbol{\eta}$.
2. Fix the modulo parameter of this solution $\boldsymbol{y'}$ and solve the non-periodic minimum cost flow problem (see (3)) by the classical network simplex method. Then, the optimal solution becomes a tree solution.
3. Modulo-Simplex-Pivoting:
   (a) For each basis exchange pair (i, j) with $\gamma_{ij} \neq 0$ calculate the cost difference $\Delta\omega_{ij}$.
   (b) If $\Delta\omega_{ij} < 0$ and $\boldsymbol{\eta}^{(a_i,a_j)} \leq \boldsymbol{\delta} = \boldsymbol{u} - \boldsymbol{\ell}$, then improve the solution by exchanging co-tree arc a_i with tree arc a_j and continue with step (a). Otherwise terminate Modulo-Simplex-Pivoting.

The non-periodic simplex algorithm terminates, if the well known complementary slackness conditions are fullfilled. For the periodic case such a strong optimality condition cannot be given. However, sometimes it is possible to transform the periodic basic solution of a modulo simplex step into a primal feasible basic solution of a relaxation

$$\tilde{\boldsymbol{y}} \in \left\{ \boldsymbol{y} \mid \tilde{F}\boldsymbol{y} \geq \boldsymbol{r} \right\} \supseteq \mathit{conv.hull}\left(\{ \boldsymbol{y} \in \mathbb{Z}^m \mid \exists \boldsymbol{z} \in \mathbb{Z}^m : \Gamma\boldsymbol{y} - T\boldsymbol{z} = \boldsymbol{b}; \boldsymbol{0} \leq \boldsymbol{y} - T\boldsymbol{z} \leq \boldsymbol{\delta}\}\right)$$

If $\tilde{\boldsymbol{y}}$ is already optimal, i. e. $\boldsymbol{\omega}^t\tilde{\boldsymbol{y}} = \min\left\{ \boldsymbol{\omega}^t\boldsymbol{y} \mid \tilde{F}\boldsymbol{y} \geq \boldsymbol{r} \right\}$, then we found the optimal solution of the overall problem. Otherwise, the basis representation of $\tilde{\boldsymbol{y}}$ has **negative** reduced costs. A basis transformation of $\tilde{F}$ will exchange a tree and a co-tree arc, which then, also done for the modulo simplex, will possibly improve the solution.

.

4 Computational Results For a Real World Scenario

4.1 The Traffic Sample

We applied the described algorithm to a real world traffic sample, which was derived from the south-west area of the German Railway Network (see Figure 4).

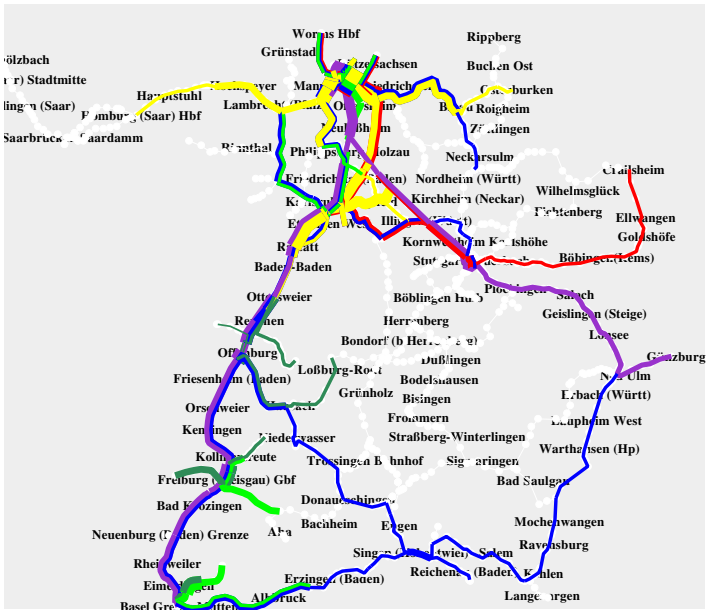


Fig. 4. The Traffic Sample contains 92 lines from the south-west area of the German railway network.

The timetabling problem contains 92 different railway lines with periods of 20, 30, 60 and 120 minutes, which results in an overall period of

$$T = lcm(20, 30, 60, 120) = 120\,\text{minutes}.$$

The resulting periodic event scheduling problem contains 669 event nodes and in total 3831 (with 3287 headway) constraints.

To solve the feasibility problem without any passenger connection constraints, we used a constraint programming approach, which finds a feasible solution within approximately one minute computation time. Next, for an origin destination matrix we applied a traffic assignment, by routing passengers on best paths. In this way we obtained for each possible connection between different lines a weight for the number of passengers using this change activity. The origin destination matrix contains only values given in percent of the total (unknown) traffic volume. For this reason, the change activity weight is primary that percentage of total volume which uses this connection. Due to the huge amount of approximately 1200 change activities with positive passenger weight, we only pick out the most important ones.

Table 4. Computational Results for the Modulo-Simplex-Algorithm

iteration	**objective**	**description**
	620952.00	initial solution from constraint propagation
	462111.00	min cost flow with fixed modulo parameter z
1	436881.00	modulo-network simplex
2	415182.00	modulo-network simplex
...	...	...
35	327113.00	modulo-network simplex
36	319874.00	single node cut improvement + min cost flow
37	312342.00	modulo-network simplex
...	...	...
56	294567.00	modulo-network simplex
57	286122.00	single node cut improvement + min cost flow
58	273789.00	modulo-network simplex
...	...	...
67	254988.00	modulo-network simplex
68	254711.00	single node cut improvement + min cost flow
69	254711.00	modulo-network simplex
68	**254711.00**	**final solution**

To do this and to get integer valued weights, the percentage was multiplied by a factor 200, which results into 570 connection constrains with weights in the range between 1 and 280. The results of the modulo network method are given by table 4. In total, the method needs approximately 20 minutes computation time.

5 Acknowledgement

This project was supported by the German Railway Company DB Netz AG.

References

1. R.K. Ahuja, T.L. Magnati, and J.B. Orlin. *Network Flows.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.
2. M. Kolonko, K. Nachtigall, and S. Voget. Exponat der Universität Hildesheim auf der CeBit 96: Optimierung von integralen Taktfahrplänen mit genetischen Algorithmen. *Hildesheimer Informatik-Berichte*, 8/96, 1996.
3. Christian Liebchen. *Periodic Timetable Optimization in Public Transport.* dissertation.de - Verlag im Internet GmbH, Berlin, 2006. Dissertation, TU Berlin, Institut für Mathematik.
4. Thomas Lindner. *Train Schedule Optimization in Public Rail Transport.* Dissertation, TUBRSW, 2000.

5. K. Nachtigall. Periodic Network Optimization with different Arc Frequencies. *Discrete Applied Mathematics*, (69):1–17, 1996.
6. Karl Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables.* Habilitationsschrift, Universität Hildesheim, 1998. auch als Institutsbericht (IB) 112-99/02 des Deutschen Instituts für Luft- und Raumfahrt, Braunschweig 1999.
7. M. Odijk. Construction of Periodic Timetables - Part I: A Cutting Plane Algorithm. Technical Report 94-61, Department of Mathematics and Computer Science. Delft University of Technology. Delft, The Netherlands, 1994.
8. Leon W.P. Peeters. *Cyclic Railway Timetable Optimization.* PhD thesis, Erasmus University Rotterdam, 2003.
9. A. Schrijver. *Theory of Linear and Integer Programming.* J. Wiley and Sons, Chichester New York Brisbane Toronto Singapore, 1986.
10. A. Schrijver and A. Steenbeek. Dienstregelingontwikkeling voor Railned. Technical report, Centrum voor Wiskunde en Informatica., 1994.
11. P. Serafini and W. Ukovich. A Mathematical Model for Periodic Scheduling Problems. *SIAM J. Discrete Math.*, 2(4):550–581, 1989.
12. W. Weigand. The Man-Maschine Dialogue and Timetable Planning. *Rail International*, 3:8–25, 1983.

The Second Chvátal Closure Can Yield Better Railway Timetables*

Christian Liebchen and Elmar Swarat

Technische Universität Berlin, Institute of Mathematics,
Straße des 17. Juni 136, D-10623 Berlin, Germany
{liebchen,swarat}@math.tu-berlin.de

Abstract. We investigate the polyhedral structure of the Periodic Event Scheduling Problem (PESP), which is commonly used in periodic railway timetable optimization. This is the first investigation of Chvátal closures and of the Chvátal rank of PESP instances.
In most detail, we first provide a PESP instance on only two events, whose Chvátal rank is *very* large. Second, we identify an instance for which we prove that it is feasible over the first Chvátal closure, and also feasible for another prominent class of known valid inequalities, which we reveal to live in much larger Chvátal closures. In contrast, this instance turns out to be infeasible already over the second Chvátal closure. We obtain the latter result by introducing new valid inequalities for the PESP, the multi-circuit cuts.
In the past, for other classes of valid inequalities for the PESP, it had been observed that these do not have any effect in practical computations. In contrast, the new multi-circuit cuts that we are introducing here indeed show some effect in the computations that we perform on several real-world instances – a positive effect, in most of the cases.

1 Introduction

It has been only recently that combinatorial optimization entered the practice of service design in public transport. The 2005 timetable of Berlin Underground is the first optimized timetable that was put into service [9]. It had been computed with integer programming techniques, namely profiting from several different classes of valid inequalities. Today, also the Dutch railways are operating a timetable that was designed with the help of techniques from combinatorial optimization and constraint programming [7]. Both projects build upon the Periodic Event Scheduling Problem (PESP).

The PESP, in its pure formulation of a feasibility problem, had been introduced by Serafini and Ukovich [18] and it generalizes the vertex coloring problem. In particular, for the two most natural optimization problems that are investigated on top of the PESP, MAXSNP-hardness has been established [8, 9]. In practice, this results in the following typical behavior of MIP solvers on medium to large sized instances. Known valid inequalities are able to close 60–90% of the initial gap between the integer optimum value and the optimum value of the LP relaxation. Still, solving this tightened IP risks to take several hours, if it is solvable at all.

* Supported by the DFG Research Center MATHEON in Berlin.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1580

There are of course much larger transportation networks in practice, which are beyond the computational limits of the methods that were used so far. As a consequence, at present there are several other research groups trying to tackle the periodic railway timetabling problem, and they are sharing the PESP as their model of choice [2, 17, 19]. For instance, Villumsen put the polyhedral approach that was suggested by Lindner [14] into practical computations for the commuter train network of Copenhagen. Unfortunately, he had to make the observation that

> "the chain cuts [14] have no effect on the solution" [19].

This is one motivation for us to have a closer look at the polyhedral structure of the feasible region of PESP instances. We do so by following the methodology that has been suggested recently by Fischetti and Lodi [6] for optimizing over the first Chvátal closure. Notice that one of the first instances to which they applied their method was the "hard MIPLIB instance `timtab1`", which is in fact a PESP model [10].

As a motivation, we first generalize an infeasible PESP instance – which is due to Lindner [14] – to a family of instances that are defined on wheel graphs. In Section 6 we will prove that these instances are feasible over the first Chvátal closure. Still worse, even the change-cycle inequalities that have been introduced by Nachtigall [15], of which in Section 4 we prove that, in general, they lie in much larger Chvátal closures, are not suited to certify infeasibility. Nevertheless, the techniques of Fischetti and Lodi suggested that these particular instances might be infeasible already over the *second* Chvátal closure. Indeed, by exploiting problem-specific insight, in the second Chvátal closure we identify general new valid inequalities for the PESP (Section 5) by which we prove that these particular instances are infeasible. We call these new inequalities the *multi-circuit cuts*.

In Section 7 we add multi-circuit cuts to the IP formulations of several timetabling instances that we took from practice. Although we have to admit that the results are not fully striking, on many instances we observe a perceptible speed-up in the solution time. In turn, on more complex instances, for which up to now no optimal solution has been found, our new cuts from the second Chvátal closure might indeed yield better railway timetables.

2 An IP for PESP

Initially, the Periodic Event Scheduling Problem (PESP, [18]) has been stated as a pure feasibility problem. We are given a directed graph $D = (V, A)$, which may feature (anti-) parallel arcs. For each arc a, there are defined some lower bound ℓ_a and some upper bound u_a. The PESP then asks whether for the given fixed period time T, the instance admits a *(periodically) feasible node potential* $\pi \in [0, T)^V$, i.e.,

$$(\pi_j - \pi_i - \ell_a) \bmod T \leq u_a - \ell_a, \ \forall a = (i, j) \in A. \tag{1}$$

In a railway timetabling context, the value T is the period time of the railway system, e.g., 60 minutes. A node i represents an arrival or departure of some specific directed line in the network, and we must assign a time value π_i to this event. For instance,

in the current timetable, the direct ICE trains from Berlin to Karlsruhe leave Berlin main station 33 minutes past the hour. Finally, in the constraint parameters ℓ and u one may encode lower and upper bounds on time durations to ensure safety requirements, transfer quality requirements, as well as many other features [11].

In a mixed-integer linear programming formulation, the modulo-operator in (1) is resolved by introducing integer variables p_a for the arcs, which we denote *periodical offsets*. Furthermore, we penalize any slack on the lower bounds ℓ_a in a linear objective function,

$$\begin{array}{lll} \min \sum_{a=(i,j)\in A} w_a(\pi_j - \pi_i + Tp_a) & & \\ \text{s.t.} \quad \pi_j - \pi_i + Tp_a \geq \ell_a, & \forall a = (i,j) \in A & \\ \pi_j - \pi_i + Tp_a \leq u_a, & \forall a = (i,j) \in A & \\ \pi_i \in [0,T), & \forall i \in V & \\ p_a \in \mathbb{Z}, & \forall a \in A. & \end{array} \tag{2}$$

Other formulations for this problem had been stated in terms of so-called tension variables $y_a = \pi_j - \pi_i$, or even *periodic tension variables* $x_a = \pi_j - \pi_i + Tp_a$, see e.g. [4, 11]. Observe that we always have $\ell_a \leq x_a$. In particular, the resulting MIPs, in which we can make the node potential variables π redundant, already perform considerably better [13]. Yet, their performance can even be enhanced—and it has to!—by adding valid inequalities. In this spirit, in the remainder of the paper we illustrate the limits of known valid inequalities, and introduce new classes of valid inequalities, which let us go beyond.

In Section 4, when we provide a relatively large lower bound on the Chvátal rank of PESP polyhedra, we will also find it most convenient to make use of the periodic tension variables x_a. Throughout the other parts of this article, however, we stay with (2). This is because we consider this formulation being more accessible, in particular for the newcomer, and it is a straightforward computation to adapt the classes of valid inequalities that we identify there to other equivalent mixed-integer programming formulations of the PESP.

The following lemma reveals that we are in fact dealing with pure integer programs.

Lemma 1 ([16]). *If ℓ, u, and T are integers, then in* (2) *w.l.o.g. we may replace $\pi_i \in [0,T)$ with $\pi_i \in \{0,\dots,T-1\}$.*

Proof. Consider an optimum solution (π^*, p^*) of (2). Now, fix the vector p^*. The resulting problem is a linear optimization problem with twice the node-arc incidence matrix of the constraint graph D as constraint matrix, which is thus totally unimodular. Since the right-hand side is integer, the LP has some integer optimum solution π°, and (π°, p^*) is feasible for (2) and not worse than the optimum solution (π^*, p^*). □

Note that the periodical offset variables p_a are either binary, or may in addition take the value two, provided that $u_a > \left\lceil \frac{\ell_a + \varepsilon}{T} \right\rceil T$. Nevertheless, w.l.o.g. we forget about any explicit bound on any of the variables in (2), and just keep their integrality requirements.

3 Chvátal Closures

Let M be an $m \times n$ matrix and consider the general rational polyhedron

$$P = \{x \mid Mx \leq b\}.$$

The *(first) Chvátal closure* P' of P is characterized by

$$P' = \{x \mid \lambda^\mathsf{T} M x \leq \lfloor \lambda^\mathsf{T} b \rfloor, \text{ for all } \lambda \geq \mathbf{0} \text{ with } \lambda^\mathsf{T} M \text{ integer}\}.$$

Also, set $P^{(0)} := P$ and recursively define $P^{(i+1)} = (P^{(i)})'$. In integer programming, we are interested in the *integer hull* P_I of P,

$$P_I := \text{conv}(\{x \in \mathbb{Z}^n \mid Mx \leq b\}).$$

The following is a key theorem in integer programming.

Theorem 1 ([3]). *For each rational polytope P there exists some integer t such that $P^{(t)} = P_I$.*

Note that in the sequel, we will switch back to $n = |V|$ and $m = |A|$, of course.

Now, denote by B the node-arc incidence matrix of a PESP constraint graph D. Then, consider the matrix

$$M := \begin{bmatrix} -B^\mathsf{T} & -T \cdot I_m \\ B^\mathsf{T} & T \cdot I_m \end{bmatrix}, \tag{3}$$

where I_m refers to the m-dimensional unit matrix. Together with the right-hand side vector

$$b := \begin{bmatrix} -\ell \\ u \end{bmatrix}, \tag{4}$$

the convex hull of the feasible solutions of (2) is nothing but P_I.

Also for the PESP, several specific studies of its polyhedral structure have been conducted [14–16]. In the sequel, we summarize some of their results and relate them to the general concept of Chvátal closures. To this end, define an *oriented circuit* $C = C^+ \dot\cup C^-$ as a subset of the arcs of D such that reorienting the elements of C^- would result in a directed circuit. The arcs in C^+ are called the *forward arcs*, and the arcs in C^- are the *backward arcs*. In particular, we distinguish the two oriented circuits that map onto the same circuit in the underlying undirected graph.

The following valid inequalities for PESP have been identified by Odijk [16].

Theorem 2 ([16]). *Let D be the constraint graph of a PESP instance and consider some oriented circuit C in D. Then the* cycle inequality

$$\sum_{a \in C^+} p_a - \sum_{a \in C^-} p_a \leq \left\lfloor \sum_{a \in C^+} \frac{u_a}{T} - \sum_{a \in C^-} \frac{\ell_a}{T} \right\rfloor \tag{5}$$

is valid for (2)*. More precisely, the cycle inequalities show up as early as in the first Chvátal closure $P^{(1)}$ of the LP-relaxation P of a PESP-polytope P_I.*

Proof. We combine these inequalities from the ones in (2). To this end, for each forward arc in C, multiply the less-than inequality of its upper bound u_a with $\frac{1}{T}$. Similarly, for each backward arc in C, multiply the greater-than inequality of its lower bound ℓ_a with $-\frac{1}{T}$, which translates into a positive coefficient in the vector λ. It is a simple observation that the node variables π all cancel out in a telescope sum. Finally, we round down the right-hand side and obtain (5). □

Both the potential strength of the cycle inequalities and the key role of the periodical offset variables p are reflected by the following theorem.

Theorem 3 ([16]). *An instance of PESP is feasible if and only if there exists an* integer *vector p such that p satisfies* all *the cycle inequalities.*

This is why we are seeking stronger valid inequalities in terms of the periodical offset variables p. In the next theorem we show that doing so we need to investigate the second Chvátal closure. This will be the main topic from Section 5 on. There, we start by highlighting that there exist some oriented circuits C in which the upper bound in (5) can even be decreased, still being valid for P_I, of course. In fact, Lindner [14] proved that the coefficients of *any* valid inequality for the PESP that only features periodical offset variables p, have to constitute a circulation in the constraint graph. Let us already mention that in Section 4 we provide an explicit proof that the Chvátal rank of a PESP instance may be at least $\frac{T}{2}$.

Denote by Q the polyhedron that is defined by taking all the inequalities from $P^{(1)}$ that do not feature any of the node variables π. Observe that formally the support of these inequalities may differ from circuits, as they are required in (5).

Theorem 4. *The cycle inequalities* (5) *constitute the complete description of Q, i.e.,*

$$Q = \{p \mid p \text{ satisfies all cycle inequalities } (5)\}.$$

Proof (idea). Basically, the proof makes use of the decomposition of an integer circulation into oriented circuits. However, due to space limitations we have to omit further details here. □

Notice that we are aware of instances on which Q does *not* equal the projection of $P^{(1)}$ onto the periodical offset variables p. In particular, there the p-part of some reversed-arc cut, which is defined in the next section, is necessary to certify the emptiness of $P^{(1)}$, while $Q \neq \emptyset$.

4 A Lower Bound on the Chvátal Rank of PESP

In this section we present the change-cycle inequalities, which were introduced by Nachtigall [15]. We provide a PESP-instance on two vertices, on which the change-cycle inequalities appear first in the $\frac{T}{2}$-th Chvátal closure, where T denotes the period time. To the best of our knowledge, this is the strongest explicit lower bound on the Chvátal rank of PESP. Unfortunately, due to space limitations we have to omit details of the proof here.

Before formulating the change-cycle inequalities, we introduce a few notation. Let C be some oriented circuit in the constraint graph of a PESP-instance. We sum the periodic tension values of the forward arcs in x^+ and the periodic tension values of the backward arcs in x^-, i.e.,

$$x^+ := \sum_{a \in C^+} x_a \qquad \text{and} \qquad x^- := \sum_{a \in C^-} x_a.$$

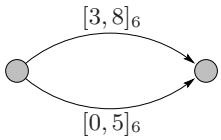


Fig. 1. A feasible PESP instance on the 2-circuit C_2 with $T = 6$

Analogously, we define

$$\ell^+ := \sum_{a \in C^+} \ell_a, \quad \text{and} \quad \ell^- := \sum_{a \in C^-} \ell_a.$$

Last, we define the slope μ and the ordinate intercept ν of the line that induces the change-cycle inequality as

$$\mu := 1 - \frac{T}{\ell^- - \ell^+ + T\tilde{z}} \qquad \text{and} \qquad \nu := (1-\mu)\ell^+ - T(\tilde{z}-1), \tag{6}$$

where $\tilde{z} := \left\lceil \frac{1}{T}(\ell^+ - \ell^-) \right\rceil$.

Theorem 5 ([15]). *The following* change-cycle inequalities

$$x^- \geq \mu x^+ + \nu \tag{7}$$

are valid for feasible instances of (2).

Notice that a similar inequality, which involves the upper bounds u_a of the arcs, is valid, too. Moreover, it had been observed in [12, Fig. 5.1] that change-cycle inequalities (7) are in a sense complementary to cycle inequalities (5).

In the remainder of this section we provide a two vertices instance of PESP, of which we prove that its Chvátal rank is $\frac{T}{2}$. In particular, the change-cycle inequality (7) of this instance does only appear in the $\frac{T}{2}$-th Chvátal closure. To this end, let T be a fixed period time and consider the following PESP-instance on two vertices: Let a_1 and a_2 be two parallel arcs, where $\ell_{a_1} = \frac{T}{2}$, $u_{a_1} = \left(\frac{3}{2}T\right) - 1$, $\ell_{a_2} = 0$, and $u_{a_2} = T - 1$. See Figure 1 for the example that corresponds to the period time $T = 6$.

In particular, in terms of periodic tension variables x_a we are dealing with the following polytope

$$P = \{(x_{a_1}, x_{a_2}, z)^\intercal \mid \frac{T}{2} \leq x_{a_1} \leq \left(\frac{3}{2}T\right) - 1,\ 0 \leq x_{a_2} \leq T-1,\ x_{a_1} - x_{a_2} = Tz\}, \tag{8}$$

where the variable z is in fact a shorthand for $p_{a_1} - p_{a_2}$. Observe that P_I corresponds to the convex hull of this PESP instance's solutions.

Proposition 1. *Consider the point* $Q_i = (\frac{T}{2} + i \cdot \frac{1}{2},\ i \cdot \frac{1}{2},\ \frac{1}{2})$. *Then* $Q_i \in P^{(i)} \setminus P^{(i-1)}$, *for all* $i \in \{1, \ldots, \frac{T}{2}\}$. *Moreover, for* $i < \frac{T}{2}$ *the points* Q_i *violate the change-cycle inequality* (7). *In particular, the change-cycle inequality* (7) *cannot be generated prior to the* $\frac{T}{2}$*-th Chvátal closure.*

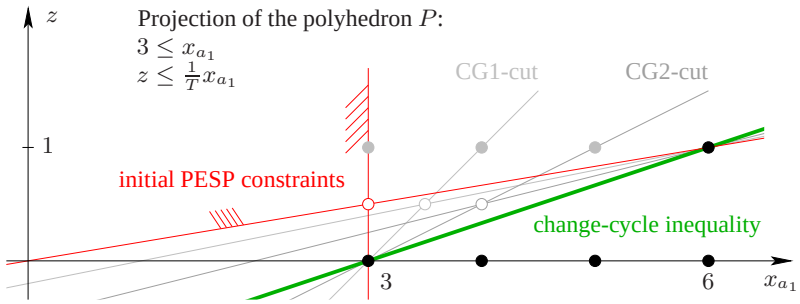


Fig. 2. A visualization of a change-cycle inequality for PESP, and its relation to Chvátal closures, here $T = 6$

Proof (sketch). In this context, the situation can be inspected best by exploiting the redundancy of the equation $x_{a_1} - x_{a_2} = Tz$ to only consider the projection into the $x_{a_1}z$-plane. In this space, the relevant inequalities of P are the initial inequality $x_{a_1} \geq \frac{T}{2}$ as well as $z \leq \frac{1}{T}x_{a_1}$, which is obtained by plugging $0 \leq x_{a_2}$ into $x_{a_1} - x_{a_2} = Tz$. Observe that the point $(x_{a_1}, z)^\mathsf{T} = (\frac{T}{2}, 0)^\mathsf{T}$ makes the former inequality tight, while $(x_{a_1}, z)^\mathsf{T} = (T, 1)^\mathsf{T}$ makes the latter inequality tight. In Figure 2, the corresponding half-spaces are drawn in red, while our ultimate goal, the change-cycle inequality (7), is drawn in green.

Then, here we can only summarize that by going from one Chvátal closure $P^{(i-1)}$ to the subsequent one $P^{(i)}$, both these inequalities are "rotated" around the points $(\frac{T}{2}, 0)^\mathsf{T}$ and $(T, 1)^\mathsf{T}$, respectively, such that the point Q_i become tight. □

Corollary 1. *The Chvátal rank of PESP is at least* $\frac{T}{2}$.

5 New Valid Inequalities for the PESP

The next section will reveal the need for new valid inequalities for the PESP: There, we present an instance for which all cycle inequalities (5) *and* change-cycle inequalities (7) are valid, although the instance is infeasible. Also, in practical computations adding these two types of valid inequalities we typically close no more than 60-90% of the initial gap between the IP optimum and its LP relaxation, and the resulting refined IPs still risk to be hard to solve. This is why here, we identify two new types of valid inequalities for the PESP polyhedron.

The first one is defined exclusively on the periodical offset variables p. By Theorem 4 we know that these cannot stem from the first Chvátal closure of the feasible region P of the LP relaxation of (2). In more detail, we specify situations in which we may decrease the right-hand side of the cycle inequalities (5). And with these new inequalities, we can easily prove the infeasibility of the instance that we discuss in depth in the next Section 6. In Section 7, we complement this analysis with promising empirical computations.

The second type of valid inequalities lives in the first Chvátal closure, and hence may now contain both types of variables, π and p. Unfortunately, due to space limitations we cannot illustrate in-depth their respective contribution here.

5.1 Multi-circuit Cuts

We start by presenting new PESP cuts from the second Chvátal closure $P^{(2)}$ of P.

Theorem 6. *Let $C_0, \ldots, C_k$ be oriented circuits with incidence vectors γ_i. Let $\lambda_i \in (0,1)$ such that $\gamma_0 = \lambda_1\gamma_1 + \cdots + \lambda_k\gamma_k$. Finally, let β_i be the right-hand sides in the cycle inequalities* (5) *of $C_1, \ldots, C_k$. Then*

$$\gamma_0^T p \leq \lfloor \lambda_1\beta_1 + ... + \lambda_k\beta_k \rfloor \tag{9}$$

is a valid inequality for $P^{(2)}$.

The proof follows immediately from Theorem 2 together with the definition of the second Chvátal closure. For some oriented circuits we may not be lucky at all, and (9) is the same as (5). However, for other cycles, the right-hand side in (9) may be much smaller than the one in (5), see Remark 1 on Page 14 for one such example. Since these cuts are obtained by representing an oriented circuit as the fractional sum of multiple other circuits, we refer to (9) as *multi-circuit cuts*.

Despite the fact that these inequalities are somehow straightforward, they are indeed useful. We will illustrate this in a detailed example in the next section, where in particular we find that

$$P^{(1)} \neq \emptyset \qquad \text{but} \qquad P^{(2)} = \emptyset.$$

5.2 Reversed-Arc Cuts

Here, we introduce one further new class of valid inequalities for the PESP, which stems from the first Chvátal closure. These inequalities were inspired by the results that we obtained by applying the methods of Fischetti and Lodi [6].

Theorem 7. *Let C be an oriented circuit, and take some backward arc $a_0 = (i,j) \in C^-$. The following inequality is valid for $P^{(1)}$*

$$\begin{aligned} \pi_j - \pi_i + (T-1)p_{a_0} + \sum_{a \in C^+} p_a - \sum_{a \in C^- \setminus a_0} p_a \\ \leq \left\lfloor \frac{1}{T} \left((T-1)u_{a_0} + \sum_{a \in C^+} u_a - \sum_{a \in C^- \setminus a_0} \ell_a \right) \right\rfloor . \end{aligned} \tag{10}$$

Proof. We provide the vector λ that combines (10) for some circuit C out of the initial matrix M. To this end, for $k \in \{0, \ldots, m\}$ consider the arc $a_k = (v,w) \in C$. Then, the rows k and $m+k$ of the matrix M correspond to the following two PESP inequalities

$$\begin{aligned} -\pi_w + \pi_v - Tp_{a_k} &\leq -\ell_{a_k}, \\ \pi_w - \pi_v + Tp_{a_k} &\leq u_{a_k}. \end{aligned}$$

Finally, choosing the components of the coefficient vector λ as

$$\lambda_k = \begin{cases} \frac{T-1}{T}, & k = m + c, \text{ where } a_c = a_0, \\ \frac{1}{T}, & k = c, \text{ where } a_c \in C^- \setminus \{a_0\}, \\ \frac{1}{T}, & k = m + c, \text{ where } a_c \in C^+, \text{ and} \\ 0, & \text{otherwise} \end{cases}$$

yields (10). □

In fact, these inequalities emerge from cycle inequalities by reversing one of their backward arcs. Hence, we refer to (10) as *reversed-arc cuts*. Observe that in some special cases, these inequalities can coincide with what Lindner [14] called *chain cutting planes*. However, for the latter Villumsen [19] had to observe in practical computations that these have "no effect" on the solution of his PESP instances. In addition to Theorem 3, this is another motivation for us to focus in our exposition on the multi-circuit cuts.

6 PESP Instances on Wheel Graphs

We introduce a family of infeasible PESP instances, for which the first Chvátal closure is still nonempty. Since the pioneering work of Edmonds [5], we are not aware of too many explicit such results. Here, even adding the change-cycle inequalities (7) does not change this status. Only adding two appropriate multi-circuit cuts (9) provides a certificate for the infeasibility of these instances. Let us annotate that these instances were inspired by an infeasible PESP instance which was studied by Lindner [14] and whose constraint graph is the wheel graph W_4 on four vertices.

We consider one fixed period time $T \geq 6$ for any of the instances that are defined below. Let $n \geq 4$ be some even number and consider the wheel graph W_n, see Figure 3 for an example with $n = 6$. We set the feasible intervals of the spoke arcs to $[0, 1]_T$, while we require $[1, T-1]_T$ for the remaining outer arcs.

We start investigating this class of instances by first giving a simple proof for the infeasibility of these instances. Hereafter, we establish that $P^{(1)} \neq \emptyset$, but $P^{(2)} = \emptyset$.

Lemma 2. *Let $T \geq 2$ and $n \geq 4$ be an even number. The PESP instance that is defined on the wheel graph W_n with feasible intervals $[0, 1]_T$ on the spokes and $[1, T-1]_T$ on the arcs of the outer circuit is infeasible.*

Proof. We may assume w.l.o.g. that $\pi_h = 0$, where h is the hub vertex in W_n. The constraints on the spokes restrict the π values of the other vertices to $\{0, 1\}$. The constraints on the remaining arcs require these two values to be used alternatingly around the outer circuit of W_n. Since we chose n to be even, the outer circuit has an odd number of vertices. But this is not compatible with the π values of all the vertices on the outer circuit taking the values zero and one alternatingly. □

The next lemma slightly simplifies the argumentation in the proof of the main theorem of this section, namely that $P^{(1)}$ is not empty.

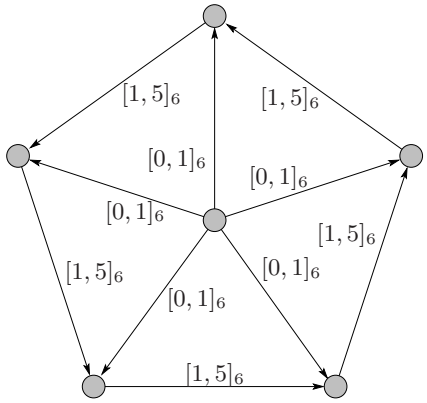


Fig. 3. An infeasible PESP instance on the wheel graph W_6 with $T = 6$

Lemma 3. *Consider some coefficient vector $\lambda \geq 0$. Let λ_a and $\lambda_{a^{-1}}$ correspond to two components whose PESP inequalities refer to the very same arc a and define $c := \min\{\lambda_a, \lambda_{a^{-1}}\}$. Derive λ' from λ by subtracting c from the components of both, a and a^{-1}. Now, if $\lambda'^{\mathsf{T}} M \leq \left\lfloor \lambda'^{\mathsf{T}} b \right\rfloor$ then $\lambda^{\mathsf{T}} M \leq \lfloor \lambda^{\mathsf{T}} b \rfloor$.*

Proof. First, observe that $(\lambda - \lambda')^{\mathsf{T}} M = 0$. Second, $(\lambda - \lambda')^{\mathsf{T}} b = c \cdot (-\ell_a + u_a) \geq 0$. Thus, rounding down cannot provide any negative value. Finally, because of $\lfloor a \rfloor + \lfloor b \rfloor \leq \lfloor a + b \rfloor$ we may add $(\lambda - \lambda')$ to λ' while keeping any valid inequality valid. □

As a consequence, for investigating $P^{(1)}$ we may assume w.l.o.g. that in any (relevant) valid inequality for $P^{(1)}$ none of the arcs shows up with both its inequalities for its respective lower and upper bounds.

Theorem 8. *$P^{(1)} \neq \emptyset$. In particular, all the cycle inequalities* (5) *and reversed-arc cuts* (10) *are valid for the same particular vector, in the case of $T \geq 6$.*

Proof. Before starting, in the vector p we distinguish the components that correspond to the $n-1$ spoke arcs from the components that correspond to the $n-1$ arcs of the outer circuit, $p^{\mathsf{T}} = (p_s^{\mathsf{T}}, p_c^{\mathsf{T}})$. Moreover, with $\mathbf{1}$ we denote the all-one vector of appropriate dimension. Our goal is to establish that

$$y_1 := (\pi^{\mathsf{T}}, p_s^{\mathsf{T}}, p_c^{\mathsf{T}}) = (\mathbf{0}^{\mathsf{T}}, \frac{1}{2T} \cdot \mathbf{1}^{\mathsf{T}}, \frac{1}{2} \cdot \mathbf{1}^{\mathsf{T}}) \in P^{(1)}. \tag{11}$$

To this end, let $\lambda^{\mathsf{T}} M x \leq \left\lfloor \lambda^{\mathsf{T}} b \right\rfloor$ be an arbitrary valid inequality of $P^{(1)}$, where M and b are as defined in (3) and (4), respectively. We have to check y_1 against this general inequality.

For ease of notation we rewrite the coefficient vector λ as $\lambda^{\mathsf{T}} = (\lambda_1^{\mathsf{T}}, \lambda_2^{\mathsf{T}}, \lambda_3^{\mathsf{T}}, \lambda_4^{\mathsf{T}})$, where λ_1 and λ_3 refer to the rows that correspond to the spokes, while λ_2 and λ_4 refer to the rows that correspond to the outer circuit of the wheel graph W_n. Moreover, λ_3 and λ_4 refer to the initial PESP-inequalities that define the upper bounds u_a, but λ_1 and

λ_2 refer to the initial PESP-inequalities that define the lower bounds ℓ_a, after having multiplied these with minus one.

Using these definitions, we find that

$$\begin{aligned}\lambda^\mathsf{T} M y_1 &= (\lambda_1^\mathsf{T}, \lambda_2^\mathsf{T}, \lambda_3^\mathsf{T}, \lambda_4^\mathsf{T}) \cdot (-\frac{1}{2} \cdot \mathbf{1}^\mathsf{T}, -\frac{T}{2} \cdot \mathbf{1}^\mathsf{T}, \frac{1}{2} \cdot \mathbf{1}^\mathsf{T}, \frac{T}{2} \cdot \mathbf{1}^\mathsf{T})^\mathsf{T} \\ &= -\frac{1}{2}||\lambda_1||_1 - \frac{T}{2}||\lambda_2||_1 + \frac{1}{2}||\lambda_3||_1 + \frac{T}{2}||\lambda_4||_1\end{aligned}$$

and

$$\begin{aligned}\left\lfloor \lambda^\mathsf{T} b \right\rfloor &= \left\lfloor (\lambda_1^\mathsf{T}, \lambda_2^\mathsf{T}, \lambda_3^\mathsf{T}, \lambda_4^\mathsf{T}) \cdot (\mathbf{0}^\mathsf{T}, -1 \cdot \mathbf{1}^\mathsf{T}, 1 \cdot \mathbf{1}^\mathsf{T}, (T-1) \cdot \mathbf{1}^\mathsf{T})^\mathsf{T} \right\rfloor \\ &= \left\lfloor -||\lambda_2||_1 + ||\lambda_3||_1 + (T-1)||\lambda_4||_1 \right\rfloor .\end{aligned}$$

In particular, for the point y_1 the initial inequality $\lambda^\mathsf{T} M y_1 \leq \left\lfloor \lambda^\mathsf{T} b \right\rfloor$ is equivalent to

$$-||\lambda_2||_1 + ||\lambda_3||_1 + (T-1)||\lambda_4||_1 - \left\lfloor -||\lambda_2||_1 + ||\lambda_3||_1 + (T-1)||\lambda_4||_1 \right\rfloor \quad (12)$$

$$\leq \frac{1}{2}||\lambda_1||_1 + \left(\frac{T}{2} - 1\right)||\lambda_2||_1 + \frac{1}{2}||\lambda_3||_1 + \left(\frac{T}{2} - 1\right)||\lambda_4||_1 \quad (13)$$

$$= \frac{1}{2}(||\lambda_1||_1 + ||\lambda_3||_1) + \left(\frac{T}{2} - 1\right)(||\lambda_2||_1 + ||\lambda_4||_1). \quad (14)$$

In order to prove that (12-13) is valid, observe first that the left-hand side (12) has values in the interval $[0, 1)$. So, we first identify some coefficient vectors λ for which (14) is at least one. Hereafter, we investigate the remaining vectors λ.

From Lemma 3, $\lambda \geq \mathbf{0}$, $\lambda^\mathsf{T} M$ being integer, and the coefficients of the periodical offsets p having value $|T|$, we conclude that for each component i of λ we have $\lambda_i = \frac{k}{T}$, with $k = 0, 1, 2, \ldots$.

Case "$||\lambda_2||_1 + ||\lambda_4||_1 \geq \frac{3}{T}$". We find immediately that (14) is at least as large as $\frac{3}{2} - \frac{3}{T}$. Now, recall that we chose the period time $T \geq 6$, and in particular (14) is at least one, establishing the theorem in this case.

Case "$||\lambda_2||_1 + ||\lambda_4||_1 = \frac{1}{T}$". In other words, the Chvátal-Gomory coefficient vector λ does only involve exactly one inequality of one arc $a = (i, j)$ of the outer circuit of W_n. In this case we are not aiming at showing that (12-13) was indeed valid. Rather, we enumerate all the eight relevant valid inequalities of $P^{(1)}$ that involve the arc a as the only arc of the outer circuit.

For that the requirement of $\lambda^\mathsf{T} M$ being integer is fulfilled, in particular for the node variables π, some of the initial PESP constraints in which π_i or π_j appear must have non-zero components in the coefficient vector λ. Because of $||\lambda_2||_1 + ||\lambda_4||_1 = \frac{1}{T}$, these must correspond to the spokes (h, i) and (h, j), where h denotes the hub of the wheel graph W_n, see Figure 4 for an illustration.

Depending on whether we use the lower bound or the upper bound inequalities of the spokes, w.l.o.g. the CG-multipliers are either $\frac{1}{T}$ or $\frac{T-1}{T}$.

First, if we choose twice the $\frac{1}{T}$, we end with the two standard cycle inequalities (5) for this triangle,

$$0 \leq p_a - p_{(h,j)} + p_{(h,i)} \leq 1. \quad (15)$$

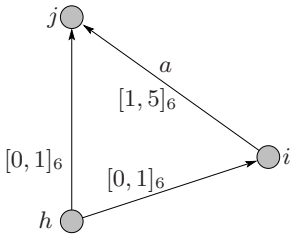


Fig. 4. A triangle in W_n with $T = 6$

For the values $p_s \equiv \frac{1}{2T}$ and $p_c \equiv \frac{1}{2}$ that we chose in our particular vector y_1, these inequalities are of course valid, because $0 \leq \frac{1}{2} \leq 1$.

Second, if for the spokes we chose once the value $\frac{1}{T}$ and once the value $\frac{T-1}{T}$, we obtain the following four reversed-arc cuts,

$$1 \leq \pi_j - \pi_h + p_a + (T-1)p_{(h,j)} + p_{(h,i)} \leq 1 \tag{16}$$
$$0 \leq \pi_i - \pi_h - p_a + p_{(h,j)} + (T-1)p_{(h,i)} \leq 0, \tag{17}$$

which are valid for our choice of y_1, too.

Last, taking $\frac{T-1}{T}$ as the coefficient for both spokes yields

$$0 \leq \pi_j - \pi_i + p_a + (T-1)p_{(h,j)} - (T-1)p_{(h,i)} \leq 1. \tag{18}$$

Also these two inequalities are valid for the vector y_1 as defined, $0 \leq \frac{1}{2} \leq 1$.

To summarize, in the case of $||\lambda_2||_1 + ||\lambda_4||_1 = \frac{1}{T}$ we considered all the eight relevant valid inequalities of $P^{(1)}$ and verified that the vector $(\pi^\mathsf{T}, p_s^\mathsf{T}, p_c^\mathsf{T}) = (\mathbf{0}^\mathsf{T}, \frac{1}{2T} \cdot \mathbf{1}^\mathsf{T}, \frac{1}{2} \cdot \mathbf{1}^\mathsf{T})$ is valid for any of them.

Case "$||\lambda_2||_1 + ||\lambda_4||_1 = \frac{2}{T}$". We distinguish between several subcases. First, we may have two non-incident arcs a_1 and a_2 of the outer circuit being involved in the cut that is defined by the coefficient vector λ. But then we are done, because we are in fact twice in the case of $||\lambda_2||_1 + ||\lambda_4||_1 = \frac{1}{T}$.

Second, we may have just one arc of the outer circuit being involved. The two cycle inequalities (5) that emerge from multiplying all its three initial constraints with $\frac{2}{T}$ are in fact nothing but just scaled versions of (15). Hence, here we need to consider valid inequalities in which some of the initial constraints are multiplied with $\frac{2}{T}$, while others are multiplied with $\frac{T-2}{T}$. The counterparts of (16) and (17) read

$$1 \leq \pi_j - \pi_h + 2p_a + (T-2)p_{(h,j)} + 2p_{(h,i)} \leq 2$$
$$-1 \leq \pi_i - \pi_h - 2p_a + 2p_{(h,j)} + (T-2)p_{(h,i)} \leq 0.$$

For the particular point y_1 these terms evaluate to $\frac{3}{2}$ and $-\frac{1}{2}$, respectively, and all the four inequalities are thus feasible. The same holds for the counterpart of (18), where y_1 yields one, which is feasible in

$$0 \leq \pi_j - \pi_i + 2p_a + (T-2)p_{(h,j)} - (T-2)p_{(h,i)} \leq 2.$$

Last, what we still have to investigate is the case in that two consecutive arcs a_1 and a_2 of the outer circuit are activated by the coefficient vector λ. Due to their orientation in W_n, in the valid inequality that is induced by λ, both arcs contribute either with their PESP inequalities that define their lower bounds, or both contribute with their PESP inequalities that define their upper bounds. In particular, the π variable of their common vertex has coefficient zero in the cut.

Hence, we are in a situation that is quite similar to the one that we already discussed in the case of $||\lambda_2||_1 + ||\lambda_4||_1 = \frac{1}{T}$. The only difference is that for the outer arcs we are now summing *twice* their lower or upper bounds in the inequalities. We summarize the relevant computations by providing the eight resulting valid inequalities – using the same notation as in the previous case – in which the reader will have no difficulty to verify that y_1 is indeed feasible,

$$\begin{aligned}
1 \leq \quad & p_{a_1} + p_{a_2} - p_{(h,j)} + p_{(h,i)} && \leq 1,\\
1 \leq \quad & \pi_j - \pi_h + p_{a_1} + p_{a_2} + (T-1)p_{(h,j)} + p_{(h,i)} && \leq 2,\\
-1 \leq \quad & \pi_i - \pi_h - p_{a_1} - p_{a_2} + p_{(h,j)} + (T-1)p_{(h,i)} && \leq 0, \text{ and}\\
0 \leq \pi_j - \pi_i + p_{a_1} + p_{a_2} + (T-1)p_{(h,j)} - (T-1)p_{(h,i)} & && \leq 2.
\end{aligned}$$

This concludes the last case for the coefficient vector λ and thus establishes (11). □

Proposition 2. *The change-cycle inequalities* (7) *are valid for the infeasible PESP instance that we consider on the wheel graphs* W_n.

Proof (sketch). We must omit the full proof due to space limitations. Nevertheless, let us compute the relevant quantities of the particular fractional solution

$$y_1 = (\pi^\mathsf{T}, p_s^\mathsf{T}, p_c^\mathsf{T}) = (\mathbf{0}^\mathsf{T}, \frac{1}{2T} \cdot \mathbf{1}^\mathsf{T}, \frac{1}{2} \cdot \mathbf{1}^\mathsf{T}) :$$

For a spoke arc a, here, the periodic tension variable is $x_a = \frac{1}{2}$, and for any other arc a, its periodic tension variable is $x_a = \frac{T}{2}$. In the most interesting case, namely the case of a triangle, cf. Figure 4 for an illustration in the case of $T = 6$, the integer variable z of this triangle evaluates to $\frac{1}{2}$. And with these values, the reader might not have any difficulties to compute the slope $\mu = -\frac{1}{T-1}$ and ordinate intersect $\nu = \frac{T}{T-1}$, and thus verify that the corresponding change-cycle inequality (7) is tight. For longer circuits, there is even some positive slack. □

Theorem 9. $P^{(2)} = \emptyset$. *In particular, two multi-circuit cuts* (9) *certify the emptiness of* $P^{(2)}$.

Proof. We apply Theorem 6 to the outer circuit C of the wheel graph W_n. We combine it linearly by summing over all the $|C|$ oriented 4-circuits that contain two consecutive edges of C.

Let C_i be one of these 4-circuits. Consider the cycle inequalities (5) of C_i and of its opposite counterpart C_i^{-1},

$$p_1 + p_2 + p_3 - p_4 \leq \left\lfloor \frac{1}{T}(1 + (T-1) + (T-1) - 0) \right\rfloor = \left\lfloor \tfrac{2T-1}{T} \right\rfloor = 1, \quad (19)$$

$$-p_1 - p_2 - p_3 + p_4 \leq \left\lfloor \frac{1}{T}(0 - 1 - 1 + 1) \right\rfloor = \left\lfloor \tfrac{-1}{T} \right\rfloor = -1, \quad (20)$$

where p_1 and p_4 are the periodical offset variables that we introduced for the two spokes of C_i. In other words, $p_1 + p_2 + p_3 - p_4 = 1$.

For that the oriented circuits C_i linearly combine C, we have to multiply each of them with $\frac{1}{2}$. Recall that we selected n to be even, thus $|C| = n - 1$ being odd. Doing so for their initial orientation, using (19) we find that

$$\sum_{a \in C} p_a \leq \left\lfloor |C| \cdot \frac{1}{2} \cdot 1 \right\rfloor = \left\lfloor \frac{n-1}{2} \right\rfloor \overset{n \text{ odd}}{=} \frac{n}{2} - 1, \tag{21}$$

because the periodical offset variables p of all the spokes cancel out. Similarly, summing (20) for all their opposite counterparts C_i^{-1} yields

$$\sum_{a \in C} -p_a \leq \left\lfloor |C| \cdot \frac{1}{2} \cdot (-1) \right\rfloor = \left\lfloor \frac{-n+1}{2} \right\rfloor \overset{n \text{ odd}}{=} -\frac{n}{2}. \tag{22}$$

Finally, multiplying (22) with minus one and comparing it to (21) yields $\frac{n}{2} \leq \frac{n}{2} - 1$ and thus reveals that indeed $P^{(2)} = \emptyset$. □

Remark 1. It is highly interesting to compare the resulting pair of inequalities (9) to their initial counterparts (5) in $P^{(1)}$:

$$\begin{aligned} P^{(1)}: \quad & \left\lceil (n-1)\tfrac{1}{T} \right\rceil \leq \sum_{a \in C} p_a \leq \left\lfloor (n-1)\tfrac{T-1}{T} \right\rfloor \quad \text{vs.} \\ P^{(2)}: \quad & \tfrac{n}{2} \leq \sum_{a \in C} p_a \leq \tfrac{n}{2} - 1. \end{aligned}$$

Hence, in a sense on the wheel graph instances the multi-circuit cuts propagate to $P^{(2)}$ the rounding benefit that particular cycle inequalities achieved already in $P^{(1)}$. □

This is our main motivation for the separation heuristic that we apply in the next section.

7 Computational Results

For the PESP, we investigate the change in the solution behavior of CPLEX 11, when adding multi-circuit cuts (9) to its IP models. To this end, we need to separate these cuts. In Remark (1) we observed that if we combine valid inequalities (5) of the first Chvátal closure in which the rounding was strong, i.e., $b - \lfloor b \rfloor \approx 1 - \varepsilon$, then, in the second Chvátal closure we can achieve much stronger multi-circuit cuts (9) than their corresponding cycle inequalities (5) in the first Chvátal closure.

In most detail, we generate multi-circuit cuts (9) in the following way.

1. Build an initial IP model of an optimization instance of PESP.
   Actually, instead of immediately using (2) we are using a purely tension-based formulation here, because in [13] it was reported that these performed best.
2. Generate valid inequalities for this IP.
   These are cycle inequalities (5) and change-cycle inequalities (7). For the separation heuristic we made the same experience as Nachtigall, namely that considering the fundamental circuits subject to a minimum spanning tree with the periodic tension values of the current LP relaxation as weights, empirically is the most efficient deterministic solution heuristic. Denote the resulting LP by LP_1.

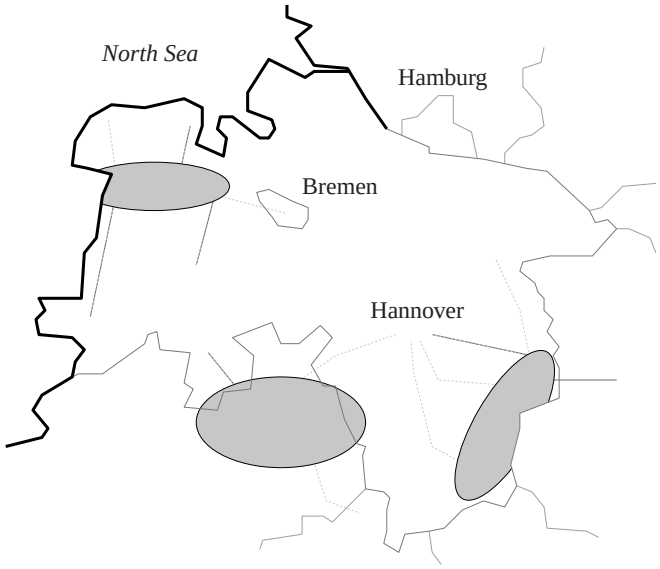


Fig. 5. The subregions of Lower Saxony and Westfalia (north-western part of Germany) of which we distill our three test instances

3. Store "strong" cycle inequalities in a pool $\mathcal{P}$.
   While computing LP_1, we record for each cycle inequality (5) that we generate its rounding benefit $\beta := b - \lfloor b \rfloor$, no matter whether it is added to LP_1 or not. If β is larger than some threshold value – we used $\beta \geq 0.7$ – then this cycle inequality is added to a pool $\mathcal{P}$ of "strong" cycle inequalities.
4. Add multi-circuit cuts (9) to LP_1.
   After Steps 2 and 3 have been accomplished, denote by x^* the optimum fractional solution of the final LP relaxation LP_1. To cut this point x^* off with some multi-circuit cut (9), we formulate the Chvátal-Gomory IP, that Fischetti and Lodi proposed in [6], for the cycle inequalities (5) in $\mathcal{P}$. Since the cycle inequalities already live in the first Chvátal closure, this way we are exploring parts of the second Chvátal closure. We iterate this CG-procedure until for some subsequent linear program LP_2 (LP_1 plus some multi-circuit cuts) its optimal solution can no more be separated by this procedure, or a time limit applies.
5. Solve the IP.
   In LP_2, switch on the integrality requirements on the periodical offset variables p and let CPLEX 11 solve this (mixed) integer linear program.

Data. We investigate the performance of the multi-circuit cuts (9) on several real-world data sets. Unfortunately, there is still not available any public library of real-world periodic railway timetabling instances. Hence, we need to resort to instances that have been available at our institute, e.g., some that had already been used in [8, 10]. In particular, all are subnetworks of the German passenger railway network.

More precisely, we consider three regions within Lower Saxony and Westfalia: Harz (H), Ostfriesland (O), and Ostwestfalen-Lippe (L), see Figure 5. All these net-

Table 1. Size of our test instances. Here, ν is the cyclomatic number $|A| - |V| + 1$, i.e., the number of integer variables in the tension-based IP models that we apply [13].

Instance name	service lines	$\|V\|$	$\|A\|$	ν	tight arcs	width
Harz 1 (H1)	17	54	309	256	26	10^{120}
Harz 2 (H2)	16	30	308	279	7	10^{149}
Harz 3 (H3)	12	43	226	184	23	10^{93}
Harz 4 (H4)	22	58	432	375	26	10^{183}
Harz 5 (H5)	15	55	332	278	29	10^{153}
Ostfriesland 1 (O1)	10	77	281	205	58	10^{99}
Ostfriesland 2 (O2)	13	107	380	274	86	10^{128}
Ostwestfalen-Lippe 1 (L1)	12	60	295	236	45	10^{108}
Ostwestfalen-Lippe 2 (L2)	12	65	289	225	48	10^{112}
Ostwestfalen-Lippe 3 (L3)	13	66	357	292	49	10^{145}

works are operated at a period time of two hours. Together with the standard time precision that is used by Deutsche Bahn AG, and which is 0.1 minutes, in our models this yields $T = 1200$. It is a general observation that cycle inequalities (5) tend to be stronger if the spans $u_a - \ell_a$ of the PESP constraints are smaller. Obviously, multi-circuit cuts (9) inherit this property. Hence, if these new valid inequalities bear any computational benefit, we hope to reveal it on instances where railway capacity is rather scarce. This is done by modeling the complete passenger traffic in the respective regions (regional and long-distance trains), and by considering single tracks. The sizes of the resulting PESP instances, after eliminating redundancies such as contracting fixed arcs with zero span, are reported in Table 1. There, in the column "tight arcs" we counted the number of arcs a with relatively small span, i.e., $u_a - \ell_a \leq \frac{T}{10}$. In the column "width", we provide a (rough) upper bound on the size of the Branch-and-Bound tree that had already been considered in [13], which is the product of the possible number of values over all the integer variables z.

Results. We summarize our computational results in Table 2. There, we compare three different policies for solving PESP instances. First, take the pure initial model as is, with no problem-specific valid inequalities being added. Its LP relaxation admits a trivial optimal solution: simply take $\pi \equiv 0$ and $p_a := \frac{\ell_a}{T}$. When reporting on values of refined LP relaxations, we scale the values such that this trivial solution has value zero, and the optimum value is 100.[1] Second, we add the problem-specific cycle inequalities (5) plus some change-cycle inequalities (7), as described above. Last, we also add multi-circuit cuts (9).

We start by giving the optimum solutions of the respective (refined) LP relaxations in the columns "LP bound". Next to this, we put the solution time under standard settings of CPLEX 11 on an Intel Core2 with 2.13 GHz and a 2GB RAM running Linux. In the last but third column we report how many multi-circuit cuts (9) could be found by the separation heuristic that we sketched above, and which was based on [6].

[1] In the tension-based IP (see [13]) we add cycle inequalities (5) as bounds on the integer variables, which typically yields values slightly larger than zero, e.g, 5–25%.

Table 2. Computational Results of adding multi-circuit cuts (9) to PESP IP models. A **boldface** entry indicates that the shortest solution time is achieved by adding multi-circuit cuts (9) (LP bounds indexed to "intopt$\widehat{=}100$", time in seconds)

	pure IP model		IP + (5) + (7)		IP + (5) + (7) + (9)		
Instance	LP bound	opt time	LP bound	opt time	# cuts (9)	LP bound	opt time
H1	4.4	325	86.0	75	2	86.0	**42**
H2	35.6	850	83.0	263	15	83.0	349
H3	4.3	64	77.8	13	64	81.0	**12**
H4	40.8	3059	86.8	2255	1	86.8	2727
H5	4.1	2921	56.7	1221	17	58.9	1663
O1	12.3	216	84.8	197	18	85.3	**79**
O2	16.7	338	84.4	365	25	85.0	**187**
L1	27.2	141	89.0	94	25	89.2	**69**
L2	11.2	203	94.7	71	22	94.7	**56**
L3	19.0	2652	90.3	1010	20	90.7	1226

To summarize, in contrast to what Villumsen [19] had to observe for the chain-cutting planes, which were due to Lindner [14], multi-circuit cuts (9) indeed have an effect on the solution behavior of CPLEX 11 on PESP instances. First of all, on each instance, CPLEX is (still, see below) better off when fed with the full machinery of additional valid inequalities, compared to not adding any cuts at all. Unfortunately, there are some instances, on which adding multi-circuit cuts (9) cause longer solution times, compared to the (5)+(7) setting. Nevertheless, in the majority of the cases, multi-circuit cuts (9) yield an improved solution behavior. In several cases, the solution time drops by more than 40%.

Additional Comments. Let us close by commenting on two interesting effects. First, in Table 2, we voluntarily decided to consider the pure LP bounds instead of the dual bound that CPLEX is able to achieve in its root node preprocessing. This is mainly motivated by the fact that the LP bounds are conceptually better accessible, compared to the result of a powerful "black box". Yet, consider the instance O2. For this, Table 2 contains entries of 16.7% and 85.0% for the LP bounds with and without cuts, respectively. But after the root node preprocessing of CPLEX 11, the respective values get together as close as 82.0% and 85.4%. Now, compare these values to the root node preprocessing of CPLEX 8.1, which is the version that had been used in an extensive computational study on other railway timetabling instances [13]: 28.6% and 85.3%. Similar observations can be made for the respective solution times.

This illustrates the improvements that more recent versions of CPLEX are able to achieve in the preprocessing of PESP IP models. Could this be a consequence of the fact that pure PESP IP models have been included in the MIPLIB [1, 10], in combination with new general IP insight, e.g., the one reported in [6]? Here, it might be interesting to recall that Fischetti and Lodi called the PESP IP models in the MIPLIB "very hard"...

Nevertheless, although the preprocessed dual bounds get closer to each other, problem-specific insight, e.g., in form of the new multi-circuit cuts (9) that we just introduced here, may still cut the solution time by roughly one half.

Second, and last but not least, we point out the high sensitivity that the models show with respect to certain specific multi-circuit cuts (9). As an example, on the instance H2 we had to make the following observation. With just inequalities (5) and (7) being added, a solution time of 263s can be observed, cf. Table 2. Then, adding just the first *two* multi-circuit cuts (9) that our separation heuristic found, the solution time is cut by more than 73% to less than 70s. But adding the next two such cuts, we end with a solution time of even 392s. In other words, if we just added the first two cuts, instead of all the 25 that we were able to separate, in Table 2 we could have replaced the value 349s in the H2 row with only **70**s. . .

On the one hand, this underlines that multi-circuit cuts (9) indeed have some effect. On the other hand, this asks for an understanding on which particular ones of these cuts are the "right" ones.

8 Conclusions

We introduced multi-circuit cuts as new valid inequalities for the Periodic Event Scheduling Problem (PESP). These live in its second Chvátal closure. For a particular family of infeasible PESP instances, we managed to prove that its first Chvátal closure is nonempty. And even adding all change-cycle inequalities, of which we further proved that in general they appear only in much larger closures, does not turn the status to infeasible. Hence, it is a first theoretical merit of the multi-circuit cuts to certify infeasibility of these particular instances. Complementary to this, in our computational study, we observed that multi-circuit cuts are likely to reduce the solution time of CPLEX 11 on PESP IP models.

We admit that up to now, our separation has not really been tuned. More theoretical insight is needed to distinguish between helpful multi-circuit cuts, and unproductive ones. We are very much confident that with such an additional insight, adding just the helpful multi-circuit cuts will *always* improve on the two other settings that we considered in Table 2. In addition, practically efficient separation heuristics for multi-circuit cuts are required, in particular if we want to use these cuts in a branch-and-cut context, too. But also any further new classes of valid inequalities from whichever Chvátal closure will be equally welcome – given that they have some (positive) effect on the solution behavior of CPLEX 11.

To summarize, of course multi-circuit cuts are not the end of the story in the solution of PESP instances. However, we feel that these are one step forward into a promising direction.

References

1. T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Oper. Res. Lett.*, 34(4):361–372, 2006.
2. G. C. Caimi, M. Fuchsberger, M. Laumanns, and K. Schüpbach. Periodic railway timetabling with event flexibility. In C. Liebchen, R. K. Ahuja, and J. A. Mesa, editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

3. V. Chvátal. Edmond's polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:205–337, 1973.
4. W. Dauscha, H. D. Modrow, and A. Neumann. On cyclic sequence types for constructing cyclic schedules. *Zeitschrift fur Operations Research*, 29(1):1–30, 1985.
5. J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research National Bureau of Standards Section B*, 69:125–130, 1965.
6. M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.
7. L. G. Kroon. Mathematics for railway timetabling. *ERCIM News*, 68:22–23, January 2007.
8. C. Liebchen. A cut-based heuristic to produce almost feasible periodic railway timetables. In S. E. Nikoletseas, editor, *WEA*, volume 3503 of *Lecture Notes in Computer Science*, pages 354–366. Springer, 2005.
9. C. Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 2008. accepted for publication.
10. C. Liebchen and R. H. Möhring. Information on MIPLIB's timetab-instances. Preprint 049/2003, TU Berlin, Mathematical Institute, 2003.
11. C. Liebchen and R. H. Möhring. The modeling power of the periodic event scheduling problem: Railway timetables - and beyond. In F. Geraets, L. G. Kroon, A. Schöbel, D. Wagner, and C. D. Zaroliagis, editors, *ATMOS*, volume 4359 of *Lecture Notes in Computer Science*, pages 3–40. Springer, 2004.
12. C. Liebchen and L. W. Peeters. On cyclic timetabling and cycles in graphs. Technical Report 761-2002, TU Berlin, Mathematical Institute, 2002.
13. C. Liebchen, M. Proksch, and F. H. Wagner. Performance of algorithms for periodic timetable optimization. In M. Hickman, P. Mirchandani, and S. Voß, editors, *Computer-Aided Systems in Public Transport (CASPT 2004)*, volume 600 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 2008.
14. T. Lindner. *Train Schedule Optimization in Public Rail Transport*. Ph.D. thesis, Technische Universität Braunschweig, 2000.
15. K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. Habilitation thesis, Universität Hildesheim, 1998.
16. M. A. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research B*, 30(6):455–464, 1996.
17. J. Opitz and K. Nachtigall. A modulo network simplex method for solving periodic timetable, 2007. A german description of their software system TAKT can be found in *Der Eisenbahningenieur*, 58(7/2007):50–55.
18. P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.
19. J. C. Villumsen. *Construction of Timetables Based on Periodic Event Scheduling*. Master's thesis, Danish Technical University, Copenhagen, 2006.

Towards Solving Very Large Scale Train Timetabling Problems by Lagrangian Relaxation*

Frank Fischer[1], Christoph Helmberg[1],
Jürgen Janßen[2], Boris Krostitz[2]

[1] Technical University of Chemnitz, Department of Mathematics, 09107 Chemnitz
[2] Deutsche Bahn AG, Konzernentwicklung, GSU 1, 60326 Frankfurt, Stephensonstr. 1

Abstract. The train timetabling problem considered is to find conflict free routes for a set of trains in a given railway network so that certain time window conditions are satisfied. We deal with the very large scale problem of constructing such timetables for the German railway network. A number of restrictions on different train types like freight trains or passenger trains have to be observed, e.g., sequence dependent headway times, station capacities, and stopping times. In order to handle the enormous number of variables and constraints we employ Lagrangian relaxation of the conflict constraints combined with a cutting plane approach. The model is solved by a bundle method; its primal aggregate is used for separation and as starting point for rounding heuristics. We present some promising results towards handling a test instance comprising ten percent of the entire network.

1 Introduction

One of the main tasks in strategic railway planning is timetable construction, i.e., to find feasible arrival and departure times for a set of trains with predefined routes. The generated timetables should satisfy a number of different constraints like headway times and station capacities, passenger train stops should lie in given time windows.

This problem is known in the literature as *Train Timetabling Problem* (TTP) and has received considerable attention in the last decades. The TTP is related to the so called *Periodic Event Scheduling Problem* introduced in [1], where periodic timetables are considered, e.g., for subway or fast-train networks, see [2] for a detailed survey on this topic.

Most approaches to the (non-periodic) TTP are based on formulations in the form of *Integer Linear Programs* (ILP) representing train routes by time discretised networks, see [3, 4, 5, 6]. This helps to deal with headway restrictions. Some authors have shown how other types of constraints like station capacities or prescribed timetables can be handled, see, e.g., [7]. The solution methods include

* This work was supported by the *Bundesministerium für Bildung und Forschung* under grant 03HEPAG4. Responsibility for the content rests with the authors.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1585

heuristic and exact branch-and-bound based approaches using LP relaxations and Lagrangian relaxations of the ILP, see [4, 8].

In this paper we deal with very large scale real world instances of the German railway company Deutsche Bahn AG (DB) as they arise in the strategic long term planning process of DB. The largest test instance comprises roughly ten percent of the entire German network with approximately 3000 trains to be scheduled for a time period of six hours.

For instances of this size the (in general exponential) number of constraints ensuring sufficient headway time between successive trains on each track necessitates the use of a primal cutting plane approach. A column generation approach for generating the single train schedules would therefore have to include dynamically the effect of the constraints separated so far, e.g., by solving a shortest path problem on a time discretised network showing the effect of the current constraints on the new variables in each time period. In essence, this is exactly what one obtains by classical Lagrangian relaxation of the conflict constraints, which decomposes the problem into shortest path problems on time discretised networks for each train. In this setting, optimizing the dual Lagrange multipliers by simple subgradient methods is not an option, as we need good approximations to the primal solutions for separating the headway constraints in the primal cutting plane approach. The less classical bundle cutting plane approach of [9] offers exactly what we need: it is a bundle method for optimizing the Lagrange multipliers of the dual and generates at the same time a primal approximate solution, the *primal aggregate*, which we use for primal separation of the headway constraints and station capacities. In our instances we deal with different train types and predefined timetable conditions for some of the trains. The Lagrangian relaxation of the model is solved using the ConicBundle package [10].

Our paper is structured as follows. In the next section we give a formal description of the TTP and introduce our model. Section 3 describes the solution methods and finally in section 4 we present preliminary computational results of our approach on our real world test instances.

2 The Train Timetabling Problem

Our TTP can be formally described as follows. We are given an *infrastructure digraph* $D = (V, A)$ representing the railway network, where V is a finite set of nodes (e.g. stations, track switches, ...) and A is the set of directed arcs representing a direction in which the corresponding track can be used. If the locations corresponding to two nodes $u, v \in V$ are connected by two tracks, one for each direction, both arcs uv and vu are in A. We also have two arcs if they are connected by a single track that is used in both directions, i.e., if these two arcs belong to the same physical track. All arcs of the latter kind are collected in the set $A_S \subseteq A$. Let T be the set of trains and $m(j) \in M$ be the *train-type* of $j \in T$ where $M = M_p \dot{\cup} M_f$ is the union of *passenger train types* M_p and *freight train types* M_f. The train-type classifies the speed, length and other properties of the train as needed for strategic planning. A train may

stop and possibly wait at a node $u \in V$ or may *pass* through the node without stopping. Let $B_S = \{wait, pass\}$ denote the *stopping behaviours* of a train. For each arc the train may stop or pass at the start or at the end node and we collect these *acceleration modes* in the set $B_R = B_S \times B_S$. Then for each arc $uv \in A$ we are given a mapping $t^R_{uv} \colon M \times B_R \to \mathbb{Z}_+$, where $t^R_{uv}(m, b)$ denotes the *running time* of a train of type m over the arc uv with respect to its acceleration mode b in minutes, and a mapping $t^H_{uv} \colon M \times B_R \times M \times B_R \to \mathbb{Z}_+$, where $t^H_{uv}(m_1, b_1, m_2, b_2)$ denotes the *minimal headway time* of a train of type m_1 with acceleration mode b_1 followed by a train of type m_2 with acceleration mode b_2 in minutes. If $uv \in A_S$, we have additional headway times for two trains passing the arc in opposite directions $t^{HS}_{uv} \colon M \times B_R \times M \times B_R \to \mathbb{Z}_+$ with the same interpretation as above.

For each train $j \in T$ the predefined route is given by the *ordered sequence of nodes* $U(j) = (u^j_1, \ldots, u^j_{n_j})$ with $n_j \in \mathbb{N}$, no other nodes are visited. The timetable for a train may be restricted in the following way. For each node u^j_i we have a *stopping-interval* $I^j_i = [t^{S,j}_i, t^{E,j}_i] \subseteq \mathbb{Z} \cup \{\pm\infty\}$ and a *minimal stopping time* $d^j_i \in \mathbb{Z}_+$. The train j has to arrive at node u^j_i before the end of its stopping interval $t^{E,j}_i$, is not allowed to leave the node before $t^{S,j}_i + d^j_i$ and has to wait at the node for at least d^j_i minutes. A waiting time $d^j_i = 0$ signals that train j does not need to stop at node u^j_i. For freight trains there are no stopping restrictions on the nodes except for the first node u^j_1. Here the interval has the form $I^j_1 = [t^{S,j}_1, \infty]$ specifying the train's starting time.

Important constraints on the timetables arise from the capacity of stations. We denote the *absolute capacity* of a node $v \in V$ by $c_v \in \mathbb{N}$, it specifies the maximal number of trains allowed to visit node v at the same time. In many stations the capacity also depends on the direction from which the trains enter the node. For an arc $uv \in A$ the *directional capacity* of the node v is $c_{uv} \in \mathbb{N}$ and describes the maximal number of trains that may stop at or pass through node v at the same time when arriving over arc uv. Clearly, $c_{uv} \leq c_v$ for all $uv \in A$.

We model the problem in a classical way via time discretised networks for the single train routes and by using coupling constraints for the capacity and headway restrictions. Let $S = \{1, \ldots, N\}$ denote the discretised time steps corresponding to minutes. For each train $j \in T$ we have a network $G^j = (V^j, A^j)$ defined as follows. The node set V^j is a subset of $\{\sigma^j, \tau^j\} \cup (B_S \times \{1, \ldots, n_j\} \times S)$, where σ^j is an *artificial source node* and τ^j an *artificial terminal node*, while, e.g., a node $(wait, i, t)^j$ has to be interpreted as train j stops in node u^j_i at time t.

The arc set A^j is built of the following subsets:

- a set of *waiting arcs* $((wait, i, t)^j, (wait, i, t+1)^j)$, for each $i \in \{1, \ldots, n_j\}$ and $t \in \{1, \ldots, N-1\}$ for nodes where the train may stop;
- a set of *running arcs* $\big((b_1, i, t)^j, (b_2, i+1, t+t^R)^j\big)$ connecting two successive nodes for $i \in \{1, \ldots, n_j - 1\}$, where $t^R = t^R_{u^j_i, u^j_{i+1}}(m(j), (b_1, b_2))$ gives the running time with respect to the stopping behaviours b_1 and b_2;

– a set of *starting arcs* of the form $(\sigma^j, (wait, 1, t)^j)$ corresponding to feasible starting times at the first node;
– a set of *ending arcs* of the form $((wait, n_j, t)^j, \tau^j)$ collecting all possible arrivals at the last node;
– a set of *infeasible arcs* $((b, i, t)^j, \tau^j)$ allowing the train to go from each intermediate node directly to the terminal node.

Of course, the graph G^j contains only those arcs that are valid for the train j with respect to the stopping intervals and minimal stopping times.

Now we introduce for each arc $a \in \mathcal{A} = \bigcup_{j \in T} A^j$ a binary variable $x_a \in \{0, 1\}$ equal to one if and only if the path associated with train j contains arc a. Let $\delta^+(v)$ and $\delta^-(v)$ denote the (possibly empty) sets of arcs leaving and entering node $v \in \mathcal{V} = \bigcup_{j \in T} V^j$. Likewise we define for $v \in V, t \in S$

$$\delta^-(v, t) = \left\{ ((b', i', t')^j, (b, i, t)^j) \in \mathcal{A} \colon u_i^j = v \right\}$$

which is the set of all train arcs arriving at the infrastructure node v at time t, and for $uv \in A, t \in S$

$$\delta^-(uv, t) = \left\{ ((b', i', t')^j, (b, i, t)^j) \in \mathcal{A} \colon u_{i-1}^j u_i^j = uv \right\}$$

which is the set of train arcs arriving at the infrastructure node v at time t over the arc uv. With appropriate arc costs $w_a, a \in \mathcal{A}$ (see Section 4), the ILP formulation reads (later we prefer the dual to be a minimization problem, so we use maximization here)

$$\text{maximize} \sum_{a \in \mathcal{A}} x_a w_a \tag{1}$$

subject to

$$\sum_{a \in \delta^+(\sigma^j)} x_a = 1, \qquad j \in T, \tag{2}$$

$$\sum_{a \in \delta^+(v)} x_a = \sum_{a \in \delta^-(v)} x_a, \qquad j \in T, v \in V^j \setminus \{\sigma^j, \tau^j\}, \tag{3}$$

$$\sum_{a \in \delta^-(v,t)} x_a \le c_v, \qquad v \in V, t \in S, \tag{4}$$

$$\sum_{a \in \delta^-(uv,t)} x_a \le c_{uv}, \qquad uv \in A, t \in S, \tag{5}$$

$$\sum_{a \in C} x_a \le 1, \qquad C \in \mathcal{C}, \tag{6}$$

$$x_a \in \{0, 1\}, \qquad a \in \mathcal{A}. \tag{7}$$

The set $\mathcal{C}$ contains the (in general exponentially large) family of maximal sets $C \subseteq \mathcal{A}$ that hold pairwise conflicting arcs. We say two arcs

$$((b_1, i_1, t_1)^j, (b_2, i_2, t_2)^j) \in A^j \quad \text{and} \quad ((b_1', i_1', t_1')^{j'}, (b_2', i_2', t_2')^{j'}) \in A^{j'}$$

with $t_1 \leq t_1'$ conflict if either

- $u^j_{i_1} u^j_{i_2} = u^{j'}_{i_1'} u^{j'}_{i_2'} = uv \in A$ and $t_1 + t^H_{uv}(m(j),(b_1,b_2),m(j'),(b_1',b_2')) > t_1'$, or
- $u^j_{i_1} u^j_{i_2} = u^{j'}_{i_2'} u^{j'}_{i_1'} = uv \in A_S$ and $t_1 + t^{HS}_{uv}(m(j),(b_1,b_2),m(j'),(b_1',b_2')) > t_1'$,

i.e., they violate the headway times.

In the objective function (1) the infeasible arcs should have costs with a sufficiently penalizing effect. Constraints (2) ensure that exactly one path per train will be used. Constraints (3) are the flow conservation constraints. The node capacities are imposed by (4) for the absolute capacities and by (5) for the directional capacities. Finally the *clique constraints* (6) forbid the use of conflicting arcs.

3 Solution Methods

Our solution method is based on the *Lagrangian dual* of the model (1)-(7) obtained by relaxing the coupling constraints (4)-(6). In order to explain the decomposition approach, we collect the coupling constraints (4)-(6) in the system $Dx \leq d$ and denote by $D^j, j \in T$, the columns corresponding to the $x_a, a \in A^j$. Furthermore for $j \in T$

$$\mathcal{X}^j := \left\{ x \in \mathbb{R}^{A^j} : x \text{ fulfills (2), (3) and (7) for fixed } j \right\}$$

represents the set of all feasible flows in Graph G^j. The Lagrangian dual problem reads

$$\min_{y \geq 0} \varphi(y)$$

where

$$\varphi(y) := d^T y + \sum_{j \in T} \varphi_j(y),$$

with

$$\varphi_j(y) := \max_{x \in \mathcal{X}^j} \sum_{a \in A^j} x_a w_a - y^T D^j x. \tag{8}$$

Obviously, the φ_j are convex functions because they are maxima over affine functions. For each y the evaluation of $\varphi(y)$ requires the solution of $|T|$ independent shortest path problems (8). Let $x(y)$ be the optimal solution of the shortest path problems for given y, then $g(y) = d - Dx(y)$ is a subgradient of φ at y.

The ConicBundle library [10] implements a bundle method to solve problems of type

$$\min_{y \geq 0} f(y)$$

where $f(y)$ is a convex function given by a first-order oracle, i.e., for given y the oracle returns $f(y)$ and a subgradient $g(y)$ of f at the point y.

The method generates a sequence $(x_k)_{k\in\mathbb{N}}$ of *primal aggregates* that are convex combinations of the primal optimizers giving rise to the subgradients of the φ_j. For an appropriate subsequence $L \subseteq \mathbb{N}$ each cluster point of $(x_l)_{l\in L}$ lies in the set of optimal solutions of the LP relaxation (if such solutions exist), see [9, 11] for technical aspects. Note, the x_k are in general not feasible for our primal problem because they violate the coupling constraints, but they yield successively better approximations to primal optimal solutions.

Since the number of constraints (6) is exponential, we separate the constraints based on the primal aggregates x_k generated by the bundle method, i.e., we add constraints to the model that we find violated by x_k and that are not yet present in the relaxation, see [9, 12] for more information on separation and convergence aspects in bundle methods.

The capacity constraints (4)-(5) are separated by complete enumeration.

The separation of the maximal clique constraints (6) is not trivial. This is because the headway times t^H_{uv} and t^{HS}_{uv} may be different for each train-type and for each stopping behaviour. [13] gives an extensive analysis of the structure of clique constraints arising from headway times in TPP and proves that the time window of interest is bounded by twice the maximum headway time. In our case this may be quite large. Therefore, we use a greedy heuristic to find large violated cliques as described in Algorithm 1. For any arc $a \in \mathcal{A}$ with positive flow value we find all arcs in conflict with a and sort them non-increasingly with respect to their flow value. Starting with the single clique $\{a\}$ we successively try to add the next arc in the sequence to all existing cliques. If the new arc does not enlarge the clique, we add the largest subclique containing it. If an upper limit N_C on the number of cliques is exceeded, we eliminate the cliques of minimal weight. The maximal cliques of each a are added as cutting planes if their weight is greater than one and if they are not yet contained in the problem. The routine is called after each descent step of the bundle method.

The last step is the computation of an integral solution. Let x be a primal aggregate returned by the bundle method. In our current approach we create an ordering of the incoming and outgoing trains for each node based on x as follows. Let $u \in V$ be an arbitrary node and let $j \in T$ be a train with $u = u^j_i$ for some $i \in \{2, \dots, n_j\}$. The *average arrival time* of j at u is given by

$$t^{j,-}_u := \frac{\displaystyle\sum_{\substack{a=((b',i-1,t'),(b,i,t))\in A^j \\ u=u^j_i}} t \cdot x_a}{\displaystyle\sum_{\substack{a=((b',i-1,t'),(b,i,t))\in A^j \\ u=u^j_i}} x_a}.$$

These average arrival times define an *arrival order* on the arriving trains visiting node u, i.e., we say for two trains $j, j' \in T$ visiting node u

$$j \preceq^-_u j' \iff t^{j,-}_u \le t^{j',-}_u.$$

Algorithm 1 Separation of clique constraints on primal aggregate x

$\mathcal{C}_{all} \leftarrow \emptyset$
for $a \in \mathcal{A} : x_a > 0$ **do**
  $A_C \leftarrow \{b \in \mathcal{A} : a \text{ is in conflict with } b\}$
  $\mathcal{C} \leftarrow \{\{a\}\}$ *"$\mathcal{C}$ is the set of new cliques."*
  while $A_C \neq \emptyset$ **do**
    Find $c \in \text{Argmax}\{x_b : b \in A_C\}$ *"Find conflicting arc c so that x_c maximal."*
    for $C \in \mathcal{C}$ **do**
      "Try to add c to each clique C."
      if $C \cup \{c\}$ is a clique **then**
        $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C\}) \cup \{C \cup \{c\}\}$
      else
        "If not possible, find (weight-)maximal subclique of C containing c."
        $\bar{C} \leftarrow \{d \in C : d \text{ conflicts with } c\}$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\bar{C}\}$
        if $|\mathcal{C}| > N_C$ **then**
          "Only keep a bounded number of cliques."
          $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\text{argmin}\{\sum_{e \in \bar{C}} x_e : \bar{C} \in \mathcal{C}\}\}$
        end if
      end if
    end for
  end while
  $\mathcal{C}_{all} \leftarrow \mathcal{C}_{all} \cup \mathcal{C}$
end for
return $\{C \in \mathcal{C}_{all} : \sum_{e \in C} x_e > 1\}$

Similarly one can define *average departure times* and a *departure order* $\preceq_u^+$ on the departing trains. Then we run a simulation on the trains, letting the trains arrive and leave as early as possible with respect to the orderings, headway times, stopping intervals and stopping times. Furthermore the data of the running times is a tight upper bound on the fastest possible traversal of the trains, so in the simulation trains may go slower over an arc. In our first experiments those constraints were never violated, but we observed some unexpected delays of passenger trains, so the approach certainly needs further improvement.

The rounding heuristic is called several times to generate timetables for a group of trains. In particular, first only long-distance trains are simulated and the corresponding arcs in the train graphs are fixed to 1. Then a new relaxation is computed for the non-fixed trains and the rounding heuristic is used to generate a timetable for the short-distance trains. Finally in a third iteration the freight trains are handled in the same way.

Since the rounding heuristic had yielded bad results for some test instance, we tried another simple heuristic. We fix successively those arcs to 1, whose values in the relaxation are above 0.8 or, if no such arc exists, the arc with the largest value. When 95% of the arcs have been fixed, we call the rounding heuristic above to generate an integral solution.

4 Numerical Results

We implemented our model in C++ using CPLEX 9.1 [14] and the ConicBundle library [10]. All computations were done on an Intel Xeon Dual Core, 3 GHz, 16 GB RAM. The test data is the south-west subnet of the network of DB (roughly Baden-Wuerttemberg). This subnet has about 10% of the size of the whole German network. We considered three test cases of different size:

1. A small part of the network containing the five most frequently used arcs.
2. The main long-distance and freight traffic route along the river Rhine.
3. The whole subnet.

All tests searched for a timetable of six hours. Table 1 shows the instance sizes (the columns Nodes and Arcs refer to the infrastructure network) as well as the number of variables in the resulting model.

Table 1. Test instances.

Instance	Nodes	Arcs	Passenger	Freight	Variables
1	104	193	242	9	317336
2	656	1210	50	67	2448842
3	2103	4681	2501	659	8990060

The cost coefficients $w_a, a \in \mathcal{A}$, have been chosen so that all trains profit from travelling as early as possible, but our tests were focused on the construction of *feasible* and not necessarily *optimal* timetables.

$$w_a = \begin{cases} -(10000 - i) & a = ((b,i,t)^j, \tau^j) \in \mathcal{A}, i \in \{1,\dots,n_j - 1\} \text{ (infeasible arc)}, \\ -(n_j - i) & a = ((b,i,t)^j, (b',i+1,t')^j) \in A^j, i \in \{1,\dots,n_j - 1\} \\ 0 & \text{otherwise.} \end{cases}$$

Because of the large amount of memory that the model requires, we were only able to solve instances 1 and 2 with CPLEX. All instances could be solved by the bundle method mentioned above. Table 2 shows the time and the memory required to solve the models.

Table 2. Solution times of the relaxation.

Instance	CPLEX	ConicBundle	Size
1	33s	12s	160 MB
2	1945s	341s	1 GB
3	–	2512s	6 GB

In order to illustrate the development and progress of the bundle cutting plane approach we present the development of the objective function in Figure

1, the development of the number of constraints and their violation in Figure 2 and Figure 3.

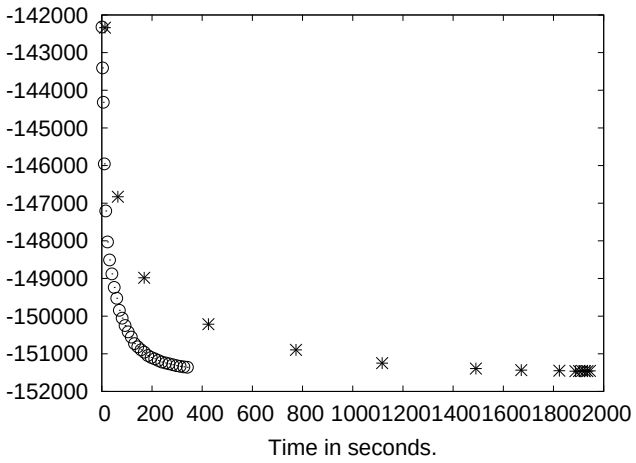


Fig. 1. Development of the objective function for Cplex (∗) and ConicBundle (∘) of instance 2.

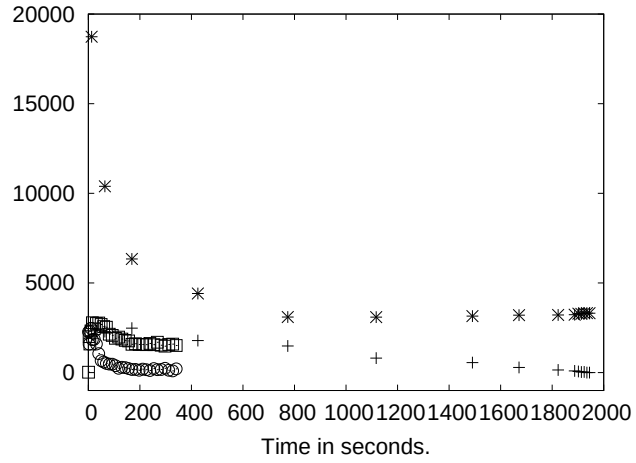


Fig. 2. Development of the number of newly separated (+) and active (∗) constraints for Cplex and the number of newly separated (∘) and active (□) constraints for ConicBundle of instance 2.

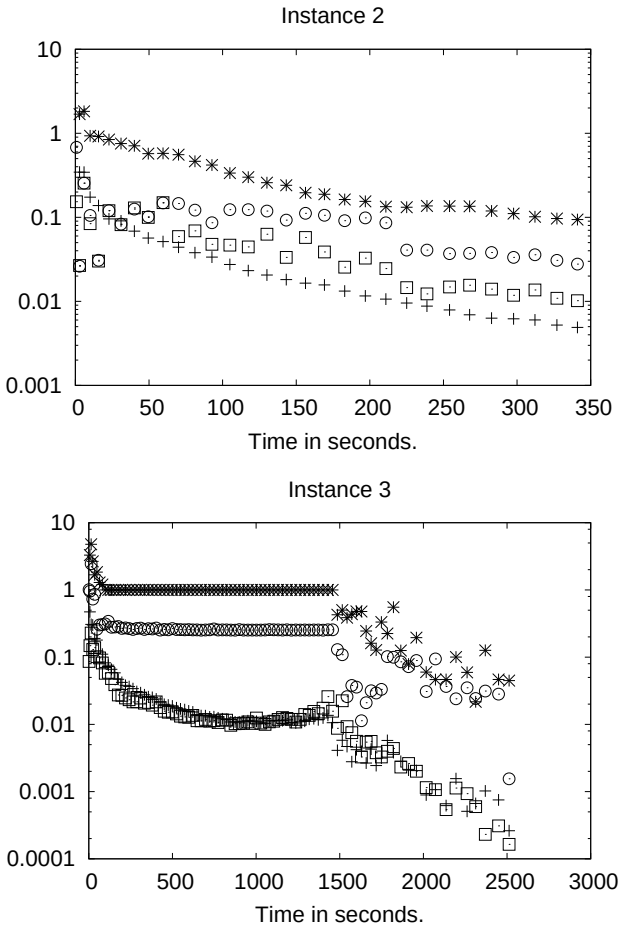


Fig. 3. Development of the maximum (○) and average (□) violation of capacity constraints and the maximum (∗) and average (+) violation of clique constraints for ConicBundle of instance 2 and 3.

As expected, the objective value shows a strong tailing-off effect which results from the combination of the respective effects for bundle and cutting plane methods. In contrast to the simplex method, the violation of active constraints that are already present in the model stays relatively high over a long time but finally tends to zero in accordance with theory.

Using the rounding heuristic described above, we generated integer solutions for all instances. For instances 1 and 2 the resulting timetable seems to be rather good with almost no delays for the passenger trains (compared with the predefined timetables). Unfortunately, the results for instance 3 are quite bad. In this case, many trains are infeasible (i.e. they use an infeasible arc) and

many passenger trains have significant delays. Therefore we used the successive fixing heuristic on that instance. This approach yielded better but not really good results (see table 3). In view of the many further possibilities to exploit structural properties and dual sensitivity information, we are confident that we shall be able to improve in the near future.

Table 3. Results of the rounding heuristic (instances 1, 2, 3) and successive fixing heuristic (instance 3b).

Instance	Time	Infeasible trains	Late trains	Average delays
1	39s	0	0	0
2	697s	0	0	0
3	3182s	40	906	865s
3b	10h	9	778	603s

Remark:

- *Infeasible trains* are those who use infeasible arcs.
- *Late trains* are passenger trains arriving more than 5 minutes later compared with the predefined timetable at at least one station.
- *Average delays* shows the average number of seconds those trains arrive later at their stations compared with the predefined timetable.

5 Conclusion

Relaxations of real world timetabling problems seem to lead to very large scale instances that are not easily solvable by commercial state-of-the-art software, but can be successfully attacked by Lagrangian relaxation combined with bundle methods. For the whole network of DB, however, more work has to be done to reduce the model size on the one hand and to separate the clique constraints more efficiently on the other hand. The solutions of the relaxation show that almost all trains use arcs only in a small time interval, so it should be possible to omit large parts of the train networks at the beginning and generate new arcs dynamically when they are required. More work needs to be invested into developing better rounding heuristics.

Bibliography

[1] Serafini, P., Ukovich, W.: A mathematical for periodic scheduling problems. SIAM J. Discret. Math. **2** (1989) 550–581

[2] Liebchen, C.: Periodic Timetable Optimization in Public Transport. PhD thesis, Technical University Berlin (2006)

[3] Caprara, A., Fischetti, M., Toth, P.: Modeling and solving the train timetabling problem. Oper. Res. **50** (2002) 851–861

[4] Brännlund, U., Lindberg, P.O., Nou, A., Nilsson, J.E.: Railway timetabling using lagrangian relaxation. Transportation Science **32** (1998) 358–369

[5] Caprara, A., Fischetti, M., Guida, P., Monaci, M., Sacco, G., Toth, P.: Solution of real-world train timetabling problems. In: HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3, Washington, DC, USA, IEEE Computer Society (2001) 3030

[6] Borndörfer, R., Schlechte, T.: Models for railway track allocation. In Liebchen, C., Ahuja, R.K., Mesa, J.A., eds.: ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, Dagstuhl, Germany, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)

[7] Cacchiani, V., Caprara, A., Toth, P.: A column generation approach to train timetabling on a corridor. 4OR: A Quarterly Journal of Operations Research **6** (2008) 125–142

[8] Caprara, A., Monaci, M., Toth, P., Guida, P.L.: A lagrangian heuristic algorithm for a real-world train timetabling problem. Discrete Appl. Math. **154** (2006) 738–753

[9] Helmberg, C.: A cutting plane algorithm for large scale semidefinite relaxations. In Grötschel, M., ed.: The Sharpest Cut. MPS-SIAM Series on Optimization. SIAM/MPS (2004) 233–256

[10] Helmberg, C.: ConicBundle 0.2d. Fakultät für Mathematik, Technische Universität Chemnitz. (2008) `http://www.tu-chemnitz.de/~helmberg/ConicBundle`.

[11] Feltenmark, S., Kiwiel, K.C.: Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems. SIAM J. Optim. **10** (2000) 697–721

[12] Belloni, A., Sagastizábal, C.: Dynamic bundle methods. Mathematical Programming (2008)

[13] Lukac, S.G.: Holes, antiholes and maximal cliques in a railway model for a single track. ZIB-Report ZR 04-18, Zuse-Institut Berlin, Takustraße 7, 14195 Berlin, Germany (2004)

[14] ILOG CPLEX Division 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA: Using the CPLEX Callable Library. (1997) Information available at `http://www.cplex.com`.