6th Workshop on Algorithmic Methods and Models for Optimization of Railways

ATMOS 2006, September 14, 2006, Zuerich, Switzerland

Edited by

Riko Jacob

Matthias Müller-Hannemann

OASIcs – Vol. 5 – ATMOS 2006 www.dagstuhl.de/oasics

Editors

Riko Jacob
ETH Zürich
Institute of Theoretical Computer Science
ETH Zentrum CAB J21.2
8092 Zürich, Switzerland
rjacob@inf.ethz.ch

Matthias Müller-Hannemann
Technische Universität Darmstadt
Department of Computer Science
Hochschulstr. 10
64289 Darmstadt, Germany
muellerh@algo.informatik.tu-darmstadt.de

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 978-3-939897-01-9

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

Publication date
August, 2006.

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.2006.i

ISBN 978-3-939897-01-9 **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 2006 Preface: Algorithmic Methods and Models for Optimization of Railways

Riko Jacob[1] and Matthias Müller-Hannemann[2]

[1] ETH Zürich, Institute of Theoretical Computer Science
ETH Zentrum CAB J21.2, CH - 8092 Zürich, Switzerland
rjacob@inf.ethz.ch
[2] Technische Universität Darmstadt, Department of Computer Science
Hochschulstr. 10, 64289 Darmstadt, Germany
muellerh@algo.informatik.tu-darmstadt.de

The 6th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 06) is held on September 14, 2006 in Zürich, Switzerland (http://algo06.inf.ethz.ch/atmos), as part of the ALGO meeting. Previous ATMOS workshops were held in Heraklion, Crete, Greece (2001), in Malaga, Spain (2002), in Budapest, Hungary (2003), in Bergen, Norway (2004), and in Palma de Mallorca, Spain (2005).

Solving railway optimization problems requires a coordinated interdisciplinary effort from various areas in mathematical optimization and theoretical computer science, including graph and network algorithms, theory of computation, approximation algorithms, combinatorial optimization, and algorithm engineering. The goal of the ATMOS workshop series is to provide a forum for the exchange and dissemination of new ideas, techniques, and research in the field of railway optimization. In particular, the workshop is meant to bring together researchers from the above areas interested in all aspects of algorithmic methods and models for railway optimization, including the development of algorithms, experimental studies, and useful prototype implementations.

The program committee received 14 submissions of full papers. After a peer-reviewing process 8 contribution were selected for presentation at the workshop, 7 of these papers are collected in this issue, the contribution of Dennis Huisman is already accepted for publication in the European Journal of Operational Research. The contributed papers are representative for several areas of research within the scope of ATMOS: locomotive and wagon scheduling, crew scheduling, line planning, quality of service aware transportation planning, periodic timetabling, and simulation studies on robustness and recovery.

In addition, the workshop includes an invited lecture by Ralf Borndörfer (Zuse-Institute Berlin, Germany) on "Directions in Railway and Public Transport Optimization", and an invited tutorial by Ravindra K. Ahuja (Univ. of Florida and Innovative Scheduling, Inc., USA) on "Next Generation Decision Support Systems in Railroad Scheduling".

ATMOS 2006 is partially supported by the ARRIVAL project a Specific Targeted Research Project funded by the Future and Emerging Technologies Unit of the EC within the 6th Framework Programme of the European Commission, under contract no. FP6-021235-2.

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/690

We would like to take this opportunity to thank the other program committee members for their timely and professional work:

- Ravindra K. Ahuja (Univ. of Florida and Innovative Scheduling Inc., USA)
- Elias Dahlhaus (DB Systems, Frankfurt a.M., Germany)
- Camil Demetrescu (Univ. of Rome La Sapienza, Italy)
- Dick Middelkoop (ProRail, Utrecht, The Netherlands)
- Martin Skutella (Univ. Dortmund, Germany)
- Paolo Toth (Univ. Bologna, Italy)

We also thank all external referees who helped in the paper selection. Finally, we would like to thank the editors of the Dagstuhl Seminar Proceedings for the opportunity to publish these proceedings within DROPS. For the upcoming workshop we wish for many nice talks and constructive discussions.

Zürich and Darmstadt, August 2006

Riko Jacob and Matthias Müller-Hannemann
PC co-chairs of ATMOS 2006

ATMOS 2006 — Abstracts Collection

6th Workshop on Algorithmic Methods and Models for Optimization of Railways

September 14, 2006 — Zürich

INVITED TALK:
Directions in Railway and Public Transport Optimization

Ralf Borndörfer (Zuse-Institute Berlin, Germany)

Optimization methods for rail and public transport have reached a high mathematical standard and can make significant contributions to the solution of the planning problems of the area. Particularly rolling stock, crew, and roster optimizers are nowadays routinely used in many companies to solve scheduling problems of unprecedented size and complexity. Such success stories often create a demand for more, such that extensions, new problems, and applications emerge directly out of practice. This demand, combined with changes in technology and regulations, result in a dynamic environment with interesting optimization challenges. The talk illustrates three of these developments from a mathematical as well as a practical point of view, namely, the more and more integrated treatment of scheduling problems, approaches to service design, and some developments related to deregulation.

TUTORIAL:
Next Generation Decision Support Systems in Railroad Scheduling

Ravindra K. Ahuja (University of Florida and Innovative Scheduling, Inc., USA)

The past few decades have witnessed numerous applications of combinatorial optimization in industry and these applications have resulted in substantial cost savings. However, the US railroad industry has been benefited from the advances and most of the planning and scheduling processes do not use modeling and optimization. Indeed, most of the planning and scheduling problems arising in railroads, which involve billions of dollars of resources annually, are currently being solved manually. The main reason for not using optimization models and methodologies is the mathematical difficulty of these problems which prevented the development of decision tools that railroads can use to obtain implementable solutions. However, this situation is now gradually changing. We are now developing cutting-edge discrete optimization and network flow based algorithms that railroads have already started using and have started deriving benefits from them. This talk will give an overview of important railroad planning and scheduling problems including blocking, train scheduling, locomotive and crew scheduling; describe new algorithms to solve some of these problems; and how these algorithms are packaged into highly interactive web-based decision support systems. We will present computational results of these algorithms on the data provided by several US railroads, demonstrating potential benefits from tens of millions of dollars annually.

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/689

A Column Generation Approach for the Rail Crew Re-Scheduling Problem

Dennis Huisman (Erasmus University Rotterdam, Netherlands)

When tracks are out of service for maintenance during a certain period, trains cannot be operated on those tracks. This leads to a modified timetable, and results in infeasible rolling stock and crew schedules. Therefore, these schedules need to be repaired. The topic of this paper is the rescheduling of crew.

In this paper, we define the Crew Re-Scheduling Problem (CRSP). Furthermore, we show that it can be formulated as a large-scale set covering problem. The problem is solved with a column generation based algorithm. The performance of the algorithm is tested on real-world instances of NS, the largest passenger railway operator in the Netherlands. Finally, we discuss some benefits of the proposed methodology for the company.

Keywords: Column generation, crew re-scheduling, large-scale optimization, railways, transportation

An Efficient MIP Model for Locomotive Scheduling with Time Windows

Martin Aronsson, Per Kreuger (Swedish Institute of Computer Science, Kista, Sweden), and Jonata Gjerdrum (Green Cargo AB, Sweden)

This paper presents an IP model for a vehicle routing and scheduling problem from the domain of freight railways. The problem is non-capacitated but allows non-binary integer flows of vehicles between transports with departure times variable within fixed intervals. The model has been developed with and has found practical use at Green Cargo, the largest freight rail operator in Sweden.

Keywords: Vehicle routing and scheduling, rail traffic resource management, resource levelling

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/683

Locomotive and Wagon Scheduling in Freight Transport

Armin Fügenschuh, Henning Homfeld (TU Darmstadt, Germany), Andreas Huck (Deutsche Bahn AG, Germany), and Alexander Martin (TU Darmstadt, Germany)

We present a new model for a strategic locomotive scheduling problem arising at the Deutsche Bahn AG. The model is based on a multi-commodity min-cost flow formulation that is also used for public bus scheduling problems. However, several new aspects have to be additionally taken into account, such as cyclic departures of the trains, time windows on starting and arrival times, network-load dependend travel times, and a transfer of wagons between trains. The model is formulated as an integer programming problem, and solutions are obtained using commercial standard software. Computational results for several test instances are presented.

Keywords: Freight transport, vehicle scheduling, time windows, Integer Programming.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/686

Periodic Metro Scheduling

Evangelos Bampas, Georgia Kaouri, Michael Lampis, and Aris Pagourtzis (National Technical University of Athens, Greece)

We introduce the Periodic Metro Scheduling (PMS) problem, which aims in generating a periodic timetable for a given set of routes and a given time period, in such a way that the minimum time distance between any two successive trains that pass from the same point of the network is maximized. This can be particularly useful in cases where trains use the same rail segment quite often, as happens in metropolitan rail networks.

We present exact algorithms for (PMS) in chain and spider networks, and constant ratio approximation algorithms for ring networks and for a special class of tree networks. Some of our algorithms are based on a reduction to the Path Coloring problem, while others rely on techniques specially designed for the new problem.

Keywords: Train scheduling, path coloring, delay-tolerant scheduling, periodic timetabling

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/684

A Game-Theoretic Approach to Line Planning

Anita Schöbel and Silvia Schwarze (Georg-August Universität Göttingen, Germany)

We present a game-theoretic model for the line planning problem in public transportation, in which each line acts as player and aims to minimize a cost function which is related to the traffic along its edges.

We analyze the model and in particular show that a potential function exists.

Based on this result, we present a method for calculating equilibria and present first numerical results using the railway network of *Deutsche Bahn*.

Keywords: Line planning, network game, equilibrium

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/688

QoS-aware Multicommodity Flows and Transportation Planning

George Tsaggouris and Christos Zaroliagis (CTI & University of Patras, Greece)

We consider the *QoS-aware Multicommodity Flow* problem, a natural generalization of the weighted multicommodity flow problem where the demands and commodity values are elastic to the Quality-of-Service characteristics of the underlying network. The problem is fundamental in transportation planning and also has important applications beyond the transportation domain. We provide a FPTAS for the QoS-aware Multicommodity Flow problem by building upon a Lagrangian relaxation method and a recent FPTAS for the non-additive shortest path problem.

Keywords: Quality of service, multicommodity flows, fully polynomial approximation scheme, transportation planning

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/682

Freight Service Design for the Italian Railways Company

Marco Campetella (Trenitalia Spa, Italy), Guglielmo Lulli (Università di Milano "Bicocca", Italy), Ugo Pietropaoli, and Nicoletta Ricciardi (Università di Roma "La Sapienza", Italy)

In this paper, we present a mathematical model to design the service network, that is the set of origin-destination connections. The resulting model considers both full and empty freight car movements, and takes into account handling costs. More specifically, the model suggests the services to provide, as well as the number of trains and the number and type of cars traveling on each connection. Quality of service, which is measured as total travel time, is established by minimizing the waiting time of cars at intermediate stations.

Our approach yields a multi-commodity network design problem with concave arc cost functions. To solve this problem, we implement a tabu search procedure which adopts "perturbing" mechanisms to force the algorithm to explore a larger portion of the feasible region. Computational results on realistic instances show a significant improvement over current practice.

Keywords: Railways transportation, service network design, tabu search

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/685

Robustness and Recovery in Train Scheduling - a Case Study from DSB S-tog a/s

Mads Hofman, Line Madsen, Julie Jespersen Groth, Jens Clausen, Jesper Larsen (Technical University of Denmark)

This paper presents a simulation model to study the robustness of timetables of DSB S-tog a/s, the city rail of Copenhagen. Dealing with rush hour scenarios only, the simulation model investigates the effects of disturbances on the S-tog network. Several timetables are analyzed with respect to robustness. Some of these are used in operation and some are generated for the purpose of investigating timetables with specific alternative characteristics.

Keywords: Train scheduling, simulation model, robustness, recovery

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/687

A Game-Theoretic Approach to Line Planning*

Anita Schöbel[1], Silvia Schwarze[2]

[1] University of Göttingen, Institute for Numerical and Applied Mathematics
37083 Göttingen, Lotzestr. 16 – 18, Germany
schoebel@math.uni-goettingen.de

[2] University of Göttingen, Institute for Numerical and Applied Mathematics
37083 Göttingen, Lotzestr. 16 – 18, Germany
schwarze@math.uni-goettingen.de

Abstract. We present a game-theoretic model for the line planning problem in public transportation, in which each line acts as player and aims to minimize a cost function which is related to the traffic along its edges. We analyze the model and in particular show that a potential function exists. Based on this result, we present a method for calculating equilibria and present first numerical results using the railway network of *Deutsche Bahn*.

Keywords. Line Planning, Network Game, Equilibrium

1 Introduction

In line planning, a *public transportation network (PTN)* is modeled by vertices for each stop (or train station) and edges for each direct connection between stops (or tracks between stations). A *line* is given as a path in the PTN and the *frequency* indicates, how often the bus or train goes within a certain time interval. The goal is to choose lines from a given *line pool* that satisfy certain criteria and minimize an objective function. The usual restrictions consider that the demand of the passengers is satisfied, i.e. that enough resources are provided to transport the customers that want to travel in the PTN. Furthermore, the amount of traffic may be limited e.g. by safety regulations. Problems of this kind have been treated with different objective functions: In [1] the lines are chosen with respect to the cost of operating the lines, but also customer-oriented objectives have been considered (see [2] for maximizing the number of direct travelers and [3,4,5,6] for recent approaches minimizing traveling times).

In our approach, we present a new model for line planning, namely from a game theoretic point of view. The lines act as players, the strategies of the players correspond to the frequencies of the lines. The payoff of the game represents the objective of the players which is to minimize the expected delay. This delay is dependent on the overall traffic and hence on the frequencies of all lines in the

* 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).

network. The remainder of the paper is structured as follows. In order to keep the notation clear, we first present the model for lines between only a single origin and a single destination (see Section 2) and show in Section 3 that this model is a special case of generalized Nash equilibrium games with a polyhedron as feasible set. In particular this allows us to prove the existence of an exact potential function. In Section 4 we extend our results to multiple origin-destination pairs. First numerical results within a real-world application are presented in Section 5. The paper is concluded by suggestions for further research.

2 The Line Planning Game Model

We consider a network $G = (V, E)$ with vertices $v \in V$ and edges $e \in E$, where V and E are nonempty and finite. A *line* P in G is given by a finite path of edges $e \in E$: $P = (e_1, ..., e_k)$. We denote the *line pool* $\mathcal{P}$ as a set of lines P in G from a single *origin* s to a single *destination* t. Multiple origin-destination pairs will be considered in Section 4.

The *frequency* of a line P is denoted by f_P. The frequencies in the complete network are represented by the *frequency vector*, given by $f \in \mathbb{R}_+^{|\mathcal{P}|}$. Furthermore, the frequency (or load) on an edge $e \in E$ is given by the sum of the frequencies on lines that are containing e,

$$f_e = \sum_{P:e\in P} f_P \ . \tag{1}$$

As common in the literature about line planning, we consider the following two restrictions. First, a *minimal frequency* $f^{min} \geq 0$ from s to t has to be covered to meet the demand of the customers, i.e. we require

$$\sum_{P\in\mathcal{P}} f_P \geq f^{min} \ . \tag{2}$$

If this condition is not satisfied all lines receive a payoff M, with M being a large number working as a penalty. The second bound is the real-valued *maximal frequency* $0 \leq f_e^{max} < \infty$ that is assigned to each edge $e \in E$, i.e. it has to hold

$$f_e \leq f_e^{max} \ \forall \ e \in E \ . \tag{3}$$

The maximal frequency establishes a capacity constraint usually given by security issues. If $f_e > f_e^{max}$ for an edge e, all lines that contain e receive a payoff of $N < M$. We allow N to be any real value smaller than M, nevertheless in the line planning problem it makes sense to choose N being a large number to punish if constraints (3 are exceeded.
Further, we will call a frequency vector f *feasible* if the both the constraints (2) and (3) are satisfied, i.e. if both bounds f^{min} and $f_e^{max}, e \in E$, are respected. The *set of feasible frequency vectors* is given by

$$\mathbb{F}^{LPG} = \left\{ f \in \mathbb{R}_+^{|\mathcal{P}|} : \sum_{P\in\mathcal{P}} f_P \geq f^{min} \ \wedge \ \sum_{P:e\in P} f_P \leq f_e^{max} \ \forall \ e \in E \right\} \ .$$

Finally, we have to specify the payoff function of the game. To this end, we first define the cost of a line P as the sum of costs on the edges belonging to that line,

$$c_P(f) = \sum_{e \in P} c_e(f_e) \ ,$$

where the cost functions $c_e(\cdot)$ describe the expected average delay on edge e, which depends on the frequency or load on e. We assume the cost functions c_e to be continuous and nonnegative for nonnegative loads, i.e. $c_e(x) \geq 0$ for $x \geq 0$. We need no further assumption on the cost functions, although in line planning the costs are usually nondecreasing.
The *payoff function* (or benefit) of a line P is for nonnegative frequency vectors f given by

$$b_P(f) = \begin{cases} c_P(f) & if \sum_{P_k \in \mathcal{P}} f_{P_k} \geq f^{min} \wedge \forall\ e \in P : f_e \leq f_e^{max} \\ N & if \sum_{P_k \in \mathcal{P}} f_{P_k} \geq f^{min} \wedge \exists\ e \in P : f_e > f_e^{max} \\ M & if \sum_{P_k \in \mathcal{P}} f_{P_k} < f^{min} \end{cases} \ .$$

Summarizing, the line planning game Γ is given by the tuple

$$\Gamma = (G, \mathcal{P}, f^{min}, f^{max}, c, N, M) \ .$$

To illustrate the payoff function of one single player (or line) P, we fix the frequencies f_{P_k} of all other players $P_k \neq P$. We obtain the frequency vector f_{-P}, by deleting the P^{th} component in the frequency vector f. The payoff function depending just on f_P is illustrated in Figure 1. The payoff consists of three

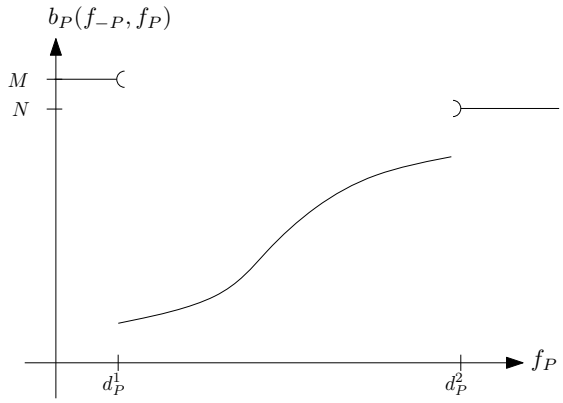


Fig. 1. Payoff of line P for a fixed frequency vector f_{-P}

continuous intervals. The left part is described by the (penalty) payment M in

case of not satisfying the minimal frequency f^{min}, the right part by the (penalty) payment N in case of exceeding a maximal frequency. The middle part is given by the sum of costs on the edges belonging to P. It is nondecreasing, if we have nondecreasing cost functions $c_e(f_e)$ on the respective edges of the path. The values that mark the boundaries of the intervals will be important later:
The *lower decision limit* of player P is given by

$$d_P^1(f_{-P}) = f^{min} - \sum_{P_k \in \mathcal{P} \setminus \{P\}} f_{P_k} \ . \tag{4}$$

The *upper decision limit* of player P is denoted by

$$d_P^2(f_{-P}) = \min_{e \in P} \{f_e^{max} - \sum_{P_k \in \mathcal{P} \setminus \{P\}} f_{P_k}\} \ . \tag{5}$$

If no confusion regarding the strategies f_{-P} of the other players arises, we denote the lower and the upper decision limit by d_P^1 and d_P^2, respectively. We obtain that

$$b_p(f) = c_P(f) \text{ if and only if } d_P^1 \leq f_P \leq d_P^2,$$

i.e. whenever $f_P \in [d_P^1, d_P^2]$, the constraints (2) and (3) are satisfied. In this case, f is feasible, if it is nonnegative. Note that it may happen that $[d_P^1, d_P^2] \cap \mathbb{R}_+$ is empty for a player P, even if $\mathbb{F}^{LPG}$ is nonempty.

As usual in game theory, we are interested in finding the equilibria of the game, which in our case represent line plans with equally distributed probability for delays. In a line planning game, a frequency vector f^* is an *equilibrium* if and only if for all lines $P \in \mathcal{P}$ and for all $f_P \geq 0$ it holds that

$$b_P(f_{-P}^*, f_P^*) \leq b_P(f_{-P}^*, f_P) \ ,$$

i.e. no player $P \in \mathcal{P}$ is able to improve its payoff by changing only his strategy. Equilibria in line planning games may be feasible or infeasible, which can be observed in the following example. As we are interested in implementable solutions, we analyze feasible frequencies in the following.

Example 1. We consider a line planning game with a line pool containing two lines. Let f_1 and f_2 be the frequencies of these lines. The minimal frequency $f^{min} = 1$ has to be covered from s to t. The game network consists of three edges, as illustrated in Figure 2. The maximal frequencies of the edges are given by $f_{e_1}^{max} = f_{e_2}^{max} = 2$ and $f_{e_3}^{max} = 3$. Furthermore, the following costs are assigned to the edges: $c_{e_1}(x) = x$, $c_{e_2}(x) = 2x$ and $c_{e_3}(x) = x^2$. Thus, we obtain payoffs: $c_1(f) = f_1 + (f_1 + f_2)^2$ for the first player and $c_2(f) = 2f_2 + (f_1 + f_2)^2$ for the second player. See Figure 3 for an illustration of the set of feasible frequencies $\mathbb{F}^{LPG}$. This line planning game provides multiple equilibria. Feasible equilibria are e.g. $f^1 = (1, 0)$ and $f^2 = (0, 1)$, with payoffs $b(f^1) = (2, 1)$ and $b(f^2) = (1, 3)$. There are also infeasible equilibria, e.g. $f^3 = (4, 4)$, where no player is able to receive a smaller payoff than N. The frequency vector $f^4 = (3, 3)$ is no

equilibrium, although no player is able to reach the set of feasible frequencies within one step. It is a property of line planning games that outside the feasible region not necessarily each player gets punished. Here, e.g. player 1 could change his frequency to zero. The resulting frequency vector $\bar{f}^4 = (0, 3)$ is still infeasible, but player 1 is able to improve his payoff from $b_1(f^4) = N$ to $b_1(\bar{f}^4) = 9$.

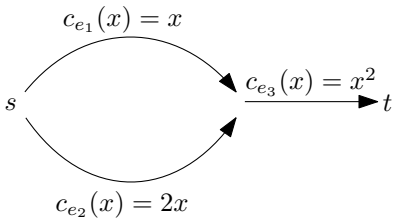


Fig. 2. Game network of Ex.1

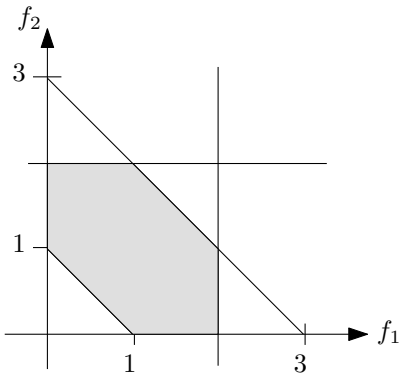


Fig. 3. Set of feasible frequencies $\mathbb{F}^{LPG}$

The above example illustrates that in line planning games, there may be areas of infeasible frequency vectors, where some players violate constraints and others do not. See Figure 4, the illustration of the two players game of Example 1 and consider the four infeasible regions A, B, C, and D. For frequency vectors that

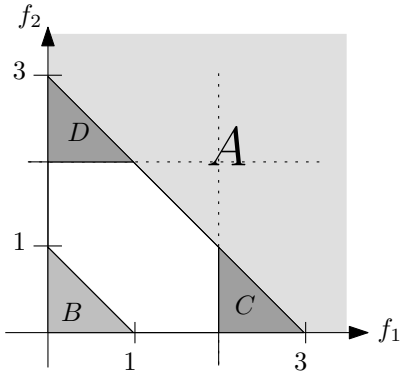


Fig. 4. Infeasible regions A, B, C, and D

lie in region A, both players get punished with payoff N, while in region B, both receive the payoff M. In region C, only player 1 gets punished with payoff N, while player 2 receives a payoff $c_2(f)$, as player 2 is satisfying the maximal frequencies f_e^{max} on his edges e_2 and e_3, but player 1 is violating $f_{e_1}^{max} = 2$. In region D the reverse situation occurs: player 2 gets punished and player 1 does not. Situations like in regions C and D happen since the players are not sharing the same set of constraints. A systematic investigation of such areas is a topic of future research (e.g. for for standard networks $G(n)$, see [7], which are a basic concept to represent all networks of a line planning game with n players, such that the set of equilibria remain unchanged).

3 Line Planning Games as Games on Polyhedra

Since we are not interested in solutions not satisfying the constraints (2) and (3) we now concentrate on feasible strategies f. First of all, note that the feasible region F^{LPG} of an LPG (see (2)) is a polyhedron. It can be represented as $S(A,b) = \{f : Af \leq b\}$ where the $(1 + |E| + |\mathcal{P}|) \times |\mathcal{P}|$-matrix A and the right-hand side vector $b \in \mathbb{R}^{(1+|E|+|\mathcal{P}|)}$ are given as

$$A = \begin{pmatrix} -\mathbb{1}_{|\mathcal{P}|} \\ \hline H \\ \hline -\mathbb{I}_{|\mathcal{P}|} \end{pmatrix} \qquad b = \begin{pmatrix} -f^{min} \\ \hline f^{max} \\ \hline \mathbb{0}_{|\mathcal{P}|} \end{pmatrix} .$$

In this formula, the $|E| \times |\mathcal{P}|$ matrix H is the edge-path incidence matrix of the underlying network with entries

$$h_{e,P} = \begin{cases} 1 & if \quad e \in P \\ 0 & else \end{cases} , \tag{6}$$

$\mathbb{1}_n = (1, \ldots, 1)$ is the vector containing n-times the entry 1, and $\mathbb{0}_n = (0, \ldots, 0)^T$ is the vector containing n-times the entry 0. Note that the polyhedron $S(A,b)$ is compact.

Example 2. In the line planning game of Example 1 the corresponding matrix H is given by

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} .$$

The polyhedron $S(A,b)$ is described by

$$A = \begin{pmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} , \qquad b = \begin{pmatrix} -1 \\ 2 \\ 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} .$$

Hence we can model LPG as a game on a polyhedron, as defined in [8]: In a game on a polyhedron with n players, each player chooses a coordinate x_n such that $(x_1, \ldots, x_n)$ lies within a given feasible set P, which is a polyhedron. Each of the players $i = 1, \ldots, n$ has a payoff function $\tilde{c}_i : P \to \mathbb{R}$ where the payoff $\tilde{c}_i(x)$ of player i depends on the complete vector x and hence also on the strategies of the other players. Since not only the payoff but also the feasibility of a strategy of player i depends on the decisions of the other players, games on polyhedra belong to the class of *generalized Nash equilibrium (GNE) games* (see [9]). Consequently, if we are only looking for feasible solutions $f \in \mathbb{F}^{LPG}$, we will call the line planning game *generalized line planning game* to stress that the feasible strategy set of each player depends on the strategies the other players. As usual for GNE games, we now require that the equilibria are feasible, i.e. in generalized line planning games, a feasible frequency vector f^* is a *generalized equilibrium* if and only if for all lines $P \in \mathcal{P}$ and for all

$$f_P \in [d_P^1(f^*_{-P}), d_P^2(f^*_{-P})] \ \cap \ \mathbb{R}_+ \ ,$$

it holds that

$$b_P(f^*_{-P}, f^*_P) \leq b_P(f^*_{-P}, f_P) \ .$$

Since only feasible solutions are considered, the payoff in the generalized line planning game is hence given by $b_P(f) = c_P(f)$. Furthermore, since the line planning game is a game on a polyhedron, we can transfer results from this type of games. One important property is the existence of a potential function.

A function $\Pi : f \to \mathbb{R}$ is an *exact restricted potential function* for a generalized line planning game Γ if for every $P \in \mathcal{P}$, for every f_{-P} with a nonempty set $[d_P^1(f_{-P}), d_P^2(f_{-P})] \cap \mathbb{R}_+$ and for every x, $z \in [d_P^1(f_{-P}), d_P^2(f_{-P})]$ it holds:

$$b_P(f_{-P}, x) - b_P(f_{-P}, z) = \Pi(f_{-P}, x) - \Pi(f_{-P}, z) \ . \tag{7}$$

A line planning game Γ is called an *exact restricted potential game* if it admits an exact restricted potential.

Exact potential functions have been introduced in [10]. The modification to a restricted version enables the investigation of GNE games and has been introduced in [7]. Although exact restricted potential functions do not exist in general for games on polyhedra, it can be shown that they exist if the cost structure of the game originates from a network. This property is called *path player game property* and has been introduced in [7]. To satisfy this property, it has to be possible to define for each subset of the set of players $\mathcal{P}$ a standard function that is dependent on the subset such that any player's payoff can be decomposed into these standard functions. In [7,8] it was shown that games on polyhedra have an exact restricted potential function whenever the path player game property holds. Fortunately, the line planning game has this property (see again [7]) such that the following holds:

Theorem 1. *A generalized line planning game is a game on a polyhedron with PPG-property. Hence, it has an exact restricted potential function.*

An exact restricted potential function is given as

$$\Pi(f) = \sum_{e \in E} [c_e(f_e) - c_e(0)].$$

To alternatively prove this result, consider the two feasible frequency vectors $f^x = (f_{-P}, x)$ and $f^z = (f_{-P}, z)$ and verify that equation (7) does hold.

$$\begin{aligned} \Pi(f^x) - \Pi(f^z) &= \sum_{e \in E} [c_e(f_e^x) - c_e(0)] - \sum_{e \in E} [c_e(f_e^z) - c_e(0)] \\ &= \sum_{e \in E} [c_e(f_e^x) - c_e(f_e^z)] = \sum_{e \in P} [c_e(f_e^x) - c_e(f_e^z)] \qquad (8) \\ &= c_P(f^x) - c_P(f^z) = b_P(f^x) - b_P(f^z) \ . \end{aligned}$$

Equation (8) is true as f^x and f^z are different only with respect to line P. Since it is a general result that for infinite potential games with continuous payoffs on compact feasible strategy sets, equilibria exist (see [10]) we directly obtain the following corollary.

Corollary 1. *In the line planning game, equilibria exist.*

Using the shape of the potential function, we furthermore obtain:

Theorem 2. *For a generalized line planning game with feasible region $\mathbb{F}^{LPG}$, a generalized equilibrium is given by an optimal solution of the following problem:*

$$\min \sum_{e \in E} c_e(f_e) \quad \textit{subject to} \quad f \in \mathbb{F}^{LPG} \ .$$

Theorem 2 provides a method for calculating equilibria in the line planning game, namely by solving the optimization problem mentioned. This method is valid for all types of continuous cost functions $c_e(f_e)$, which is the strength of this approach. On the other hand, not necessarily all equilibria are found by using Theorem 2. Other approaches which determine all equilibria for line planning games with linear costs or strictly increasing costs are presented in [7].

Example 3. Consider the line planning game analyzed in Example 1. By Theorem 2 an equilibrium can be found by solving the following problem:

$$\min f_1 + 2f_2 + (f_1 + f_2)^2 \quad \text{subject to} \quad f \in \mathbb{F}^{LPG} \ . \qquad (9)$$

The solution of the optimization problem, and thus an equilibrium is given by $f^* = (1, 0)$ with $b(f^*) = (2, 1)$. Note that f^* is the unique solution of (9), but not the unique equilibrium.

4 Multiple Origin-Destination-Pairs

In this section we consider a network $G = (V, E)$ with Q multiple origin-destination(OD)-pairs $\{s_q, t_q\}$, $q = 1, \ldots, Q$. For the q^{th} OD-pair, the pool of lines connecting s_q and t_q is given by $\mathcal{P}_q$. The paths are given as pairwise disjoint sets:

$$\mathcal{P}_{q_1} \cap \mathcal{P}_{q_2} = \emptyset \quad \forall\, q_1, q_2 = 1, \ldots, Q, q_1 \neq q_2 \ .$$

With $q(P)$ we denote the index of the OD-pair $\{s_q, t_q\}$ such that $P \in \mathcal{P}_q$. Since each line P is assigned to exactly one OD-pair, $q(P)$ is well-defined. Furthermore, the minimal frequency for the q^{th} OD-pair is given by f_q^{min}. We denote:

$$\mathcal{P} = \bigcup_{q=1,\ldots,Q} \mathcal{P}_q \quad \text{and} \quad f^{min} = \left(f_q^{min}\right)_{q=1,\ldots,Q} \ .$$

The maximal frequencies on edges f_e^{max} and the cost $c_e(f_e)$ assigned to the edges are defined as in the single origin-destination case. We call such a game *line planning game with multiple OD-pairs*.

The *payoff* for player $P \in \mathcal{P}_q$ and a nonnegative frequency vector f in an LPG with multiple OD-pairs is given by

$$b_P(f) = \begin{cases} c_P(f) & \text{if } \sum_{P_k \in \mathcal{P}_{q(P)}} f_{P_k} \geq f_{q(P)}^{min} \wedge \forall\, e \in P : f_e \leq f_e^{max} \\ N & \text{if } \sum_{P_k \in \mathcal{P}_{q(P)}} f_{P_k} \geq f_{q(P)}^{min} \wedge \exists\, e \in P : f_e > f_e^{max} \\ M & \text{if } \sum_{P_k \in \mathcal{P}_{q(P)}} f_{P_k} < f_{q(P)}^{min} \end{cases} \ .$$

Like in the single OD-pair case a frequency vector f is called *feasible* if the bounds $f_q^{min}, q = 1, \ldots Q$ and $f_e^{max}, e \in E$ are satisfied. The *set of feasible frequencies for line planning games with multiple OD-pairs* is given by

$$\mathbb{F}^{LPGMOD} = \left\{ f \in \mathbb{R}_+^{|\mathcal{P}|} : \sum_{P \in \mathcal{P}_q} f_P \geq f_q^{min} \ \forall\, q \in Q \ \wedge \sum_{P : e \in P} f_P \leq f_e^{max} \ \forall\, e \in E \right\} \ .$$

Finally, for a player $P \in \mathcal{P}_q$ we have to adjust the definition of the lower decision limit presented in (4):

$$d_P^1(f_{-P}) = f_{q(P)}^{min} - \sum_{\substack{P_k \in \mathcal{P}_q \\ P_k \neq P}} f_{P_k} \ ,$$

while the upper decision limit stays the same as in (5). If we just consider feasible frequencies $f \in \mathbb{F}^{LPGMOD}$, we obtain a *generalized line planning game with multiple OD-pairs*. Generalized equilibria are defined in such games similar to the single OD-pair case.

Recall the definition of the edge path incidence matrix H. A line planning game

with multiple OD-pairs is represented by a game on a polyhedron S(A,b) with:

$$A = \left(\begin{array}{ccccc} -\mathbb{1}_{|\mathcal{P}_1|} & 0 & \dots & 0 & 0 \\ 0 & -\mathbb{1}_{|\mathcal{P}_2|} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -\mathbb{1}_{|\mathcal{P}_{m-1}|} & 0 \\ 0 & 0 & \dots & 0 & -\mathbb{1}_{|\mathcal{P}_m|} \\ \hline \multicolumn{5}{c}{H} \\ \hline \multicolumn{5}{c}{-\mathbb{I}_{|\mathcal{P}|}} \end{array}\right) \qquad b = \left(\begin{array}{c} -f_1^{min} \\ -f_2^{min} \\ \vdots \\ -f_m^{min} \\ \hline f^{max} \\ \hline \mathbb{0}_{|\mathcal{P}|} \end{array}\right) .$$

Example 4. We consider a line planning game with four OD-pairs as illustrated in Figure 5. Let $f_q^{min} = 1 \; \forall \; q = 1, \dots, Q$ and $f_e^{max} = 4 \; \forall \; e \in E$. We denote the

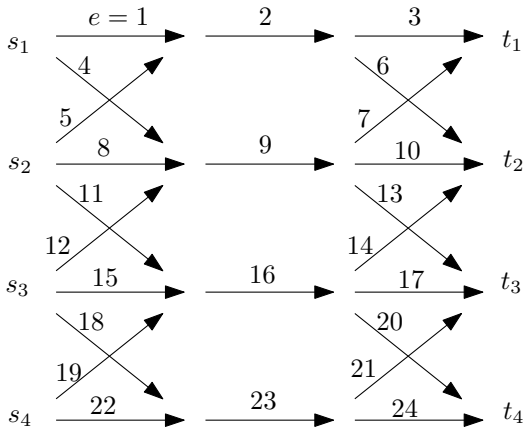


Fig. 5. Game network of Example 4

edges $e = 1, \dots, 24$ and the lines with $P^1, \dots, P^{10}$. The frequency of the lines is given by $f_1, \dots, f_{10}$. The line pools are given by

$$\begin{aligned} \mathcal{P}_1 &= \{P^1, P^2\} &&= \{(1,2,3),(4,9,7)\} \; , \\ \mathcal{P}_2 &= \{P^3, P^4, P^5\} &&= \{(5,2,6),(8,9,10),(11,16,14)\} \; , \\ \mathcal{P}_3 &= \{P^6, P^7, P^8\} &&= \{(12,9,13),(15,16,17),(18,23,21)\} \; , \\ \mathcal{P}_4 &= \{P^9, P^{10}\} &&= \{(19,16,20),(22,23,24)\} \; . \end{aligned}$$

We introduce cost functions $c_e(f_e) = f_e$ for all edges e in E. We apply Theorem 2 and solve

$$\begin{aligned}\min \sum_{e\in E} c_e(f_e) &= \min \sum_{e\in E} c_e \left(\sum_{P\in\mathcal{P}} h_{e,P} f_P \right) \\ &= \min \left(3f_1 + 3f_2 + 3f_3 + 3f_4 + 3f_5 + 3f_6 + 3f_7 + 3f_8 + 3f_9 + 3f_{10}\right) \\ &\quad \text{subject to} \quad f \in S(A,b) \ .\end{aligned}$$

As each frequency f_P has exactly the same coefficient in the objective function, each frequency that satisfies $\sum_{P\in\mathcal{P}_q} f_P = f_q^{min} = 1$, e.g. $f^1 = (1,0,1,0,0,1,0,0,1,0)$, is an optimal solution and thus also an equilibrium. The objective value is 4 for all these solutions. The payoff for f^1 is given by $b(f^1) = (4,1,4,1,1,3,1,0,3,0)$, while for $f^2 = (1,0,0,1,0,0,1,0,0,1)$ we have a payoff $b(f^2) = (3,1,1,3,1,1,3,1,1,3)$.

We can use this approach also for nonlinear cost functions. Set e.g. $c_e(f_e) = f_e^2$ for all edges e in E. The objective function $\min \sum_{e\in E} c_e(f_e)$ yields the optimal solution

$$f^3 = (0.538, 0.462, 0.385, 0.308, 0.308, 0.308, 0.308, 0.385, 0.462, 0.538)$$

with an objective value of 7.385. Solving this problem as an integer problem yields $f^4 = (0,1,1,0,0,0,0,1,1,0)$, with objective value 12.

5 Line Planning for Interregional Trains in Germany

We implemented our approach of Theorem 2 using real-world data, related to the German railway system of *Deutsche Bahn AG*. In particular we consider train stations connected by interregional trains, such as *InterCityExpress (ICE)*, *InterCity (IC)*, and *EuroCity (EC)*. The following studies are meant to test the possibility of implementing our method with realistic data and to obtain equilibria based on larger databases. Our numerical study is interesting due to the following two reasons:

- Although the investigation of line planning games is still in an early stadium, and the results are hence not ready for practical use yet, the study illustrates that further research in this field is worthwhile.
- Second, the numerical behavior of the method for finding equilibria in the line planning game is demonstrated.

The following data is at our disposal:

- OD-matrix describing 319 train stations and the minimal frequency f_q^{min} given for the OD-pairs.
- Three line databases of different size, containing 132 (S - small), 688 (M - medium) and 2770 (L - large) lines.

The line databases are not in a form suitable for our model. We will discuss later how line pools are created from this data. From theoretically $319 \times 318 = 101\,442$ OD-pairs, still 56 646 have a positive minimal frequency f_q^{min} and have to be considered. Thus, we have a line planning game with multiple OD-pairs. For those OD-pairs, $f_q^{min} \in [1, 4831]$ hold. Note that the values of f^{min} are to be interpreted as weights dependent on the number of passengers. From these weights, frequencies are obtained by a linear transformation.

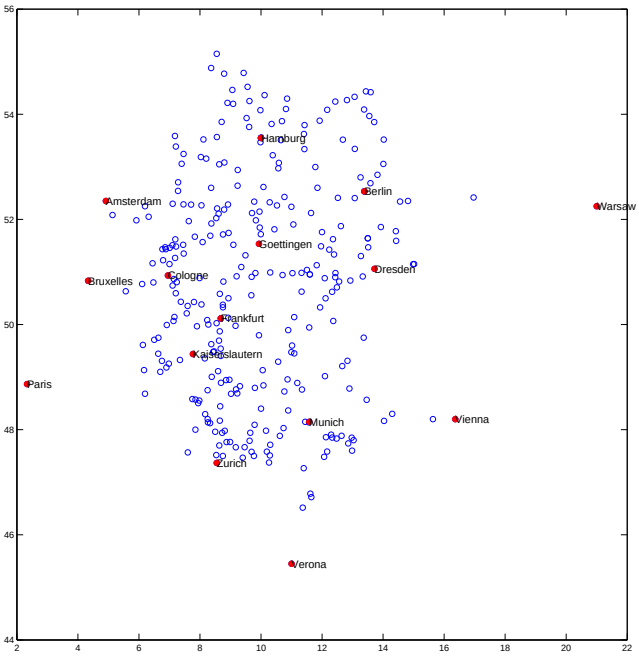


Fig. 6. Train stations under consideration

The train stations under consideration are located in Germany and neighboring countries. Figure 6 illustrates the locations of all 319 stations. The following information are needed for the line planning game, but are not given in the data. We do not have available the maximal frequency f_e^{max} on the edges, as well as we do not know the costs $c_e(f_e)$ assigned to the edges. Thus, we have to make assumptions for the implementation of our model. As there is no maximal frequency on the edges, we choose the value sufficiently large for each edge, such that the maximal frequency is satisfied for our problems. In particular, we set $f_e^{max} = 100\,000$. Regarding the cost function $c_e(f_e)$, Theorem 2 allows to use any continuous function. We implement the strictly increasing function $c_e(f_e) = f_e$

on all edges $e \in E$.

As the line planning game model considers only direct connections between stations, we neglect all OD-pairs where no direct connection exists in the line pool. For the future design of line databases, this should be taken into consideration. Furthermore, we introduce the bound U_q and consider only OD-pairs where $f_q^{min} > U_q$ does hold. This bound is used to consider just "important" OD-pairs with high minimal frequencies for our computations and it is a tool to control the size of the problem.

Furthermore, we have to construct a line pool from the line databases according to the definitions in our model. As we reduced the number of OD-pairs, we have to analyze only lines that are relevant for the OD-pairs under consideration. Thus, we generate the line pool by using these lines. On the other hand, one line may offer a direct connection for more than one OD-pair. In our model, we assume disjoint line pools, i.e. one line has to be assigned to exactly one OD-pair. According to this, we duplicate lines that provide a direct connection for more than one OD-pair. The lines have to be given such that we obtain a line pool $\mathcal{P} = \bigcup_{q=1,\dots,Q} \mathcal{P}_q$ consisting of disjoint subsets $\mathcal{P}_q$. Note that the frequencies of the original lines from the databases S, M and L are then given by the sum over the frequencies of its duplicates.

We study five scenarios with a different number of OD-pairs and use different line databases. In Studies 1,2 and 3, we consider the same set of OD-pairs, namely for $f_q^{min} > 599$, but we change the size of the line database. In Studies 2,4 and 5, the line database is invariant (we choose the medium sized one), but the set of OD-pairs is changed.

<table>
<tr><th></th><th>$f_q^{min} > 999$</th><th>$f_q^{min} > 599$</th><th>$f_q^{min} > 399$</th></tr>
<tr><td>small</td><td rowspan="3">Study 4</td><td>Study 1</td><td rowspan="3">Study 5</td></tr>
<tr><td>medium</td><td>Study 2</td></tr>
<tr><td>large</td><td>Study 3</td></tr>
</table>

Table 1 contains the computational results. We present a short explanation of the content in the following list:

Column 3 Number of OD-pairs which satisfy $f_q^{min} > U_q$
Column 5 Size of line databases
Column 6 Number of OD-pairs with direct connections and which satisfy $f_q^{min} > U_q$
Column 7 Size of line pool constructed from line database, including duplicates of lines
Column 8 Number of lines with positive frequency, i.e. that are established for the PTN (including duplicates)
Column 9 Objective function value of the optimization problem solved with Method 1
Column 10 Reference to Figure of PTN
Columns 12 – 14 Copied from the first part of the table, for easier reading
Columns 15 – 19 Statistical information about length of each line (number of stations)
Columns 20 – 24 Statistical information about number of lines (including duplicates) serving each train station

It can be observed from Studies 1 – 3 that for a larger database, more direct connections are available and thus more OD-pairs can be served. The number

Table 1. Computational results

1	2	3	4	5	6	7	8	9
Study	U_q	# OD-pairs	Line data-base	Size data base	# OD-pairs with direct connections	Size line pool	Lines with positive frequency	$c \times x$
1	599	251	S	132	87	262	88	1 402 494.001
2	599	251	M	688	117	1287	156	2 151 352.000
3	599	251	L	2770	157	5544	244	2 636 404.000
4	999	113	M	688	53	493	68	1 456 873, 000
5	399	499	M	688	132	2610	299	2 971 507.012

11	12	13	14	15	16	17	18	19	20	21	22	23
				# of stations per line					# of lines per station			
Study	Line data-base	# OD-pairs with direct connections	# of chosen lines	min	max	mean	var	histogram Figure	min	max	mean	var
1	S	87	88	6	33	15.15	25.94	7	1	43	4.18	52.76
2	M	117	156	9	20	15.06	10.87	8	1	73	7.36	131.97
3	L	157	244	6	37	14.18	23.83	9	1	121	10.84	323.17
4	M	53	68	9	20	15.21	11.66	10	1	34	3.24	30.59
5	M	132	299	9	20	15.34	10.44	11	1	129	14.38	435.38

of chosen lines hence also increases. Nevertheless, it is not growing in the same speed as the line pool, but about the same speed as the number of considered OD-pairs. Thus, in our examples, the number of established lines is not so much influenced by the size, but by the structure of the line pool, i.e. how much OD-pairs are served by the line pool. This should be taken into consideration for the design of line pools that are to be used for line planning games. In Studies 2,4 and 5, the lower bound U_q on minimal frequencies is small, hence we obtain a larger number of OD-pairs. It can also be observed that with increasing number of OD-pairs, the number of established lines is increasing, although the line data base is unchanged.

Considering the number of stations contained in the established lines, this example shows a relation to the chosen line database S, M or L. It can be observed that the results in the three scenarios using database M are similar.

In terms of lines per station, the station served by the highest number of lines in each study is Frankfurt(Main) Süd.

Histogram: Number of Stations per Line

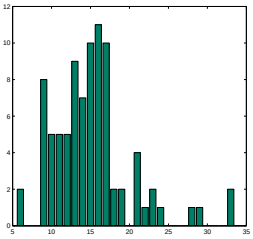


Fig. 7. $U_q = 599$, S

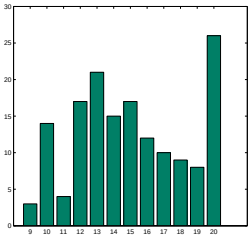


Fig. 8. $U_q = 599$, M

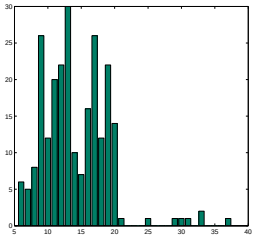


Fig. 9. $U_q = 599$, L

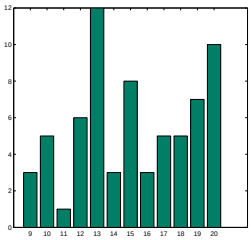


Fig. 10. $U_q = 999$, M

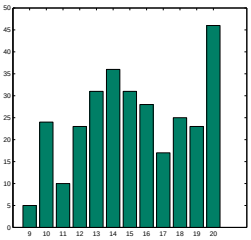


Fig. 11. $U_q = 399$, M

6 Conclusion and further research

The line planning game is a new model for analyzing line planning problems with game theoretical means. In particular it is a special case of a game on polyhedra in which an exact potential function exists. This result is the basis for an algorithm to calculate equilibria of the game. Numerical results have been presented.

Other methods for finding *all* equilibria in the path player games for linear or strictly increasing functions have been developed in [7]. The implementation of the second of these approaches, which seems to be realistic in line planning, is under research.

Although the resulting line plans seem to be suitable for practical applications, other aspects of line planning have been neglected and are topics for future research. Among these are setup costs for installing the lines (or fixed costs for operating a line) which can be approximated by bounding the maximal number of lines which may be installed. From the passengers' point of view, the travel time of their journeys should be considered; a first measure can be the length

of the lines. Another drawback of the basic model presented in this work is that frequencies f_P are real numbers, while in practice, only $f_P \in \mathbb{N}_0$ make sense. A first extension to an integer LPG has been considered in [7], where also an algorithmic approach has been developed to find integer equilibria. Finally, it will be a reasonable extension of the current model to take into account that passengers may want to change lines. More results and an implementation of this topic is under research.

References

1. Claessens, M., van Dijk, N., Zwaneveld, P.: Cost optimal allocation of rail passenger lines. European Journal on Operational Research **110** (1996)
2. Bussieck, M., Kreuzer, P., Zimmermann, U.: Optimal lines for railway systems. European Journal of Operational Research **96** (1996) 54–63
3. Schöbel, A., Scholl, S.: Line planning with minimal transfers. In: Proceedings of 5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS). (2005)
4. Scholl, S.: Customer-oriented line planning. PhD thesis, Technische Universität Kaiserslautern (2005)
5. Borndörfer, R., Grötschel, M., Pfetsch, M.E.: Models for line planning in public transport. Technical Report 04-10, ZIP Berlin (2004)
6. Borndörfer, R., Grötschel, M., Pfetsch, M.: A path-based model for line planning in public transport. Technical Report 05-18, ZIP Berlin (2004)
7. Schwarze, S.: Path Player Games: Analysis, Generalization and Application. PhD thesis, University of Göttingen (2006) available online: `http://www.num.math.uni-goettingen.de/schwarze/schwarze_ppg.pdf`.
8. Schöbel, A., Schwarze, S.: Games on polyhedra. Working Paper (2006)
9. Harker, P.: Generalized Nash games and quasi-variational inequalities. European Journal of Operational Research **54** (1991) 81–94
10. Monderer, D., Shapley, L.S.: Potential games. Games and Economic Behavior **14** (1996) 124–143

An Efficient MIP Model for Locomotive Scheduling with Time Windows

Martin Aronsson[1], Per Kreuger[1] and Jonatan Gjerdrum[2]

[1] Swedish Institute of Computer Science (SICS)
Box 1263, SE-164 29 Kista, Sweden
{martin.aronsson,per.kreuger}@sics.se
[2] Green Cargo AB
Box 39, SE-171 11 Solna, Sweden
Jonatan.Gjerdrum@greencargo.com

Abstract This paper presents an IP model for a vehicle routing and scheduling problem from the domain of freight railways. The problem is non-capacitated but allows non-binary integer flows of vehicles between transports with departure times variable within fixed intervals. The model has been developed with and has found practical use at Green Cargo, the largest freight rail operator in Sweden.

Keywords Vehicle routing and scheduling, rail traffic resource management, resource levelling

1 Introduction

The increasing competition within the railway transportation sector requires long-term sustainable and effective resource utilisation methods for companies such as Green Cargo, the largest rail freight operator in Sweden.

In many countries in Europe, railroads have traditionally been state-owned organisations with diverse interests in e.g. passenger traffic, freight traffic, infrastructure and real estate investments. The Swedish state railway was properly deregulated in all these aras around the millennium, creating separate companies with dedicated resources. Before the deregulation, locomotives were used for passenger traffic in the day-time and freight traffic at night. Today, vehicles are dedicated to one type of traffic, which has brought about utilisation patterns such as in figure 1. In 2005, Green Cargo is facing heavy reinvestments in its locomotive fleet.

If Green Cargo would be able to even out (*level*) the production resource requirements at peak times less investments would be needed. Green Cargo thus aims at a levelled week-day production.

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/683

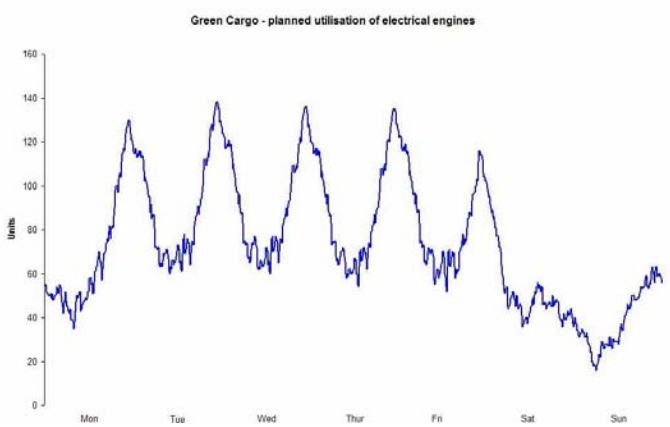


Fig. 1. Vehicle utilization pattern

1.1 Timetable

Swedish railway companies have to adhere to timetables partly designed by the government authorities. Green Cargo as well as other operators bids for allocation slots in the track network based on information about traffic patterns and customer requirements. If no slot conflicts arise, the operators receive their bids. However, the service potential is in some sense beyond the control of the individual operators.

1.2 Locomotive optimisation

The locomotive optimisation process determines the turn-round plan for all locomotives. In the fleet assignment process, a sequence of timetable slots is assigned to each locomotive by optimising planning tools based on network models. Furthermore, multiple locomotives and deadhead passive transports usage is determined, and options for maintenance is provided.

There are a number of rules or preconditions that a solution to this problem has to comply with: only specific points for switching locomotives exist, specific locomotives switching times are required, pulling power at specific lines are prespecified as is the type of locomotives that can be utilised.

Normally, the *timetable slots* are considered as *given* in this process. However, the flexibility of scheduling rail freight timetables is greater than that of passenger railways as long as customer requirements are catered for. If the slots were allowed to be shifted in time, this would enable several locomotive turns to take place which would otherwise be considered infeasible. I.e. it is not enough to only generally reduce the peaks of the traffic. To reduce the number of locomotives in the turn-round plans, the turns have to be made at the right moments as well.

Locomotives thus need to be at the right location at the right time considering that customer, resource sharing (e.g. crew and waggons) and turn-round requirements as well as input from operators sharing the track resources also have to be taken into account when determining the slots. Both duration (travelling time) and end points of the slot are determined by these factors.

There have been no commercially available tools to help the planners bridge the gap between timetable slot planning and locomotive turn-round planning. The idea presented here introduce the opportunity to modify the slot positioning in the early phase of the timetabling process so as to enable the locomotive optimisation process to achieve better results while retaining quality in terms of customer service level performance. A further optimised locomotive turn-round plan will also result in less reinvestment requirements making it an important incentive for conducting this research.

2 Problem description

Minimum cost network flow models have been extensively used (see e.g. [1]) to compute an optimal allocation of a set of scheduled transports to vehicles. The transports are normally represented as nodes in a network and the fact that a vehicle, used by one transport, can also be used by an other one, as a (directed) arc between their corresponding nodes. Classical network flow models of this kind usually have set partitioning structure and binary flow variables so that each transport is allocated to a unique vehicle. They also usually include one or more depot-nodes and are acyclic either in the plane or on a cylinder.

A straightforward generalisation of this type of flow model for cyclic schedules without depots, allows (small) integer values for the flows and have been used for engine routing in rail transportation (see e.g. [2]). In such models, additional integer variables are associated with each node to encode how many vehicles travel with each transport. Flow is conserved on each node without any depots giving cyclic schedules for each vehicle. Lower and upper bounds on the node variables variables capture the minimum and maximum number of vehicles required and usable by each transport. In addition, multiple commodities can be used to encode heterogeneous vehicle fleets.

Lower bounds on the node variables vary in known cases from 0 on (potential) passive transports (service trains) to 2 for heavy freight transports which require at least two engines. Upper bounds larger than the corresponding lower ones encode the possibility to relocate additional *accompanying* vehicles with a planned transport that is already served with the required number of vehicles. With a cost function penalising the total number of vehicles needed and any nonproductive relocation of the vehicles we get a straight forward and practical model which has seen several years of practical use in e.g. the Swedish rail industry.

Normally, the network is statically generated using temporal non-overlap and distance conditions on the transports. It would, however, be of great practical value if this kind of model could be generalised to allow for rescheduling of

the transports in cases where this would significantly reduce the cost of the vehicle usage. Using time windows for the departure times of the transports and an initial network with connections between any two transports which arrive and depart from the same location, breaks the locality (and hence, the network structure) of the model since a transfer of a vehicle (turn) from one transport to another may pose requirements on the placement of the transports within its time window which can be incompatible with requirements posed by another transfer.

Problems of this general type are variants of the "multiple Travelling Salesman Problem" (m-TSP). The case with time windows is normally referred to as a "multiple Travelling Salesman Problem with Time Windows" m-TSPTW. See e.g. [3,1,4,5,6,7]. This problems is normally (e.g. [8]) considered as a special case of the extensively studied class "Vehicle Routing Problems" (VRPs) [9,10]. One could also argue that the m-TSPTW is an uncapacitated variant of the "Vehicle Routing Problems with Time Windows" (VRPTW) which is also well studied, albeit using mainly other methods than the one proposed here. See e.g. [11,12,13,14,15,16,17,18]. They are also part of the larger problem of assigning engines to pre-scheduled transports based on more general transport and vehicle properties referred to as "Locomotive Scheduling Problems" (see e.g. [5]).

The problem under study here differs from most of those studied in this literature by allowing non binary (but small integer) flows between nodes in the network. This corresponds to multiple (required and/or optional) temporally overlapping visits in the m-TSPTW. The problem does not include any finite capacities on vehicles as in VRPTW but neither does it have the simple set partitioning structure of the m-TSPTW type of problem. The model presented here uses a single commodity but should be straight forward to generalise for heterogeneous vehicle fleets. However, the practicality of such a generalisation has not been investigated.

The paper presents an IP-model for this problem which can be used to efficiently and exactly solve practical problems up to the size of those occurring in real life transportation planning for moderate sizes ($<$ 3 hours) of departure time windows using a state-of-the-art commercial solver.

The model and its implementation for the solution of a fully operational large scale practical case is presented. The transports in this case are train transports with a fixed schedule whose departure times are relaxed from ± 15 up to ± 90 minutes and the vehicles considered are the engines used to pull the trains. Performance results for solving several version of the practical problem using CPLEX 9 [19] on a PC-type workstation are also reported.

3 Model parameters

The model is parametrised by a number of constants and variables with associated bounds which will be summarised here. The constraints and objective function will be presented in section 5 below. Note that we have chosen to present the variable bounds, which are, of course, also parameters to the model, in connec-

tion with the respective variables below. Note also that the problem is periodic, i.e. that the transport schedule is repeated after a fixed period CT. The individual vehicle schedules may, on the other hand, take several such periods before they are repeated.

3.1 Constants

n	The number of transports in the problem.
CT	Cycle Time (period after which the transport schedule is repeated).
t_i	Travel time for transport i. We require each t_i to be positive and strictly smaller than CT.
p_i	Penalty per vehicle *accompanying* a transport above that of its vehicle requirement.
lo_i, ld_i	Origin and destination locations for transport i.
r_{ij}	Setup time (turn-time) for the exchange of one or more vehicles between transport i and j. We require each r_{ij} to be positive and fulfil the inequality $t_i + r_{ij} < CT$. This requirement is quite natural in most cyclic transportation problems where travel times and setup times are typically small in comparison with the cycle period.
M	Any "sufficiently large" constant ("big M") used to linearise the model.

3.2 Decision variables (discrete)

X_{ij}	Integer variable, determining how many vehicles are *turned* (transferred) from transport i to transport j. In the train case considered here, the lower bound, $\underline{X_{i,j}}$ is normally 0 and the upper bound $\overline{X_{i,j}}$ either 1 or 2.
C_{ij}, C'_{ij}	Boolean variables, used to determine if a *turn* (if any) from transport i to transport j crosses the period limit CT one or more times.
Y_{ij}, Y'_{ij}	Integer variables, which for any optimal solution will have the values $Y_{ij} = C_{ij}X_{ij}$ and $Y'_{ij} = C'_{ij}X_{ij}$ respectively.
S_i	Integer variable used to encode the number of vehicles allocated to transport i. A lower bound $\underline{S_i}$ on this variable encodes the vehicle requirement of the transport while an upper bound $\overline{S_i}$ limits the number of vehicles usable/transportable by it. For the train case where only engine vehicles are considered, these limits are normally $\underline{S_i} = 1$ and $\overline{S_i} = 2$ unless the transport is a scheduled potential vehicle relocation transport, in which case the the lower bound may be $\underline{S_i} = 0$, indicating that the transport need not be performed unless needed to balance the vehicle flow of the model.
E_i	Integer variable used to encode the number of vehicles *accompanying* a transport in addition to the number required $\underline{S_i}$ by the transport itself.

3.3 Time point variables (continuous)

d_i Continuous variable denoting the departure time of train i. The departure time window of the transport i is encoded as the bounds $\underline{d_i}$ and $\overline{d_i}$ of d_i. For any i, we require that $0 \leq \underline{d_i} \leq \overline{d_i} < CT$.

This formulation does not guarantee that the arrival times $d_i + t_i$ will always be smaller than CT which influence the formulation of the constraints relating the arrival and departure events of the transports. The following section gives a case analysis of the situations that can occur and motivates the constraint formulation given in the section following it.

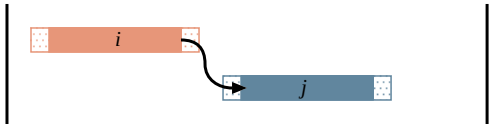
4 Turning over the cycle time border

The cases are illustrated by figures where a coloured vertical bar represents the transports. The length of the coloured bar is the travel time of the transport (the interval between scheduled departure time and arrival time). The surrounding transparent bar illustrates the departure time window of the transport so that the coloured bar may be placed anywhere within the transparent one.

There are four main cases for a turn to transport i to transport j to consider, each one described below.

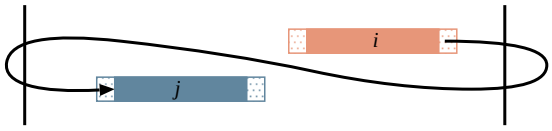
A_o The turn, if chosen, will never cross the cycle time boarder i.e.

$$\overline{d_i} + t_i + r_{ij} \leq \underline{d_j}$$



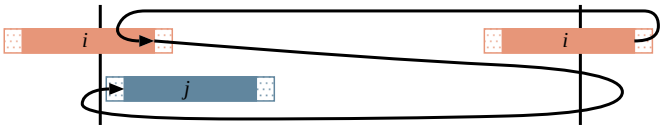
A_1 The turn, if chosen, is certain to cross the cycle time boarder exactly once. i.e:

$$\left(\underline{d_i} + t_i + r_{ij} > \overline{d_j}\right) \wedge \left(\overline{d_i} + t_i + r_{ij} - CT \leq \underline{d_j}\right)$$



A_2 A more rare case which has nevertheless to be taken into account is when the turn, if chosen, is certain to cover two periods. Note that in this case (as well as sometimes, in A_1), two instances of the transport that crosses the boarder have to be considered, one leaving the period and one entering the period, i.e:

$$\left(\underline{d_i} + t_i + r_{ij} - CT > \overline{d_j}\right)$$



This is hardly ever desirable, at least not if the period time is long in comparison with the longest travel time. In the Swedish rail freight problem the period time is a week and the longest transport travel time normally less than 24 hours.
In the model below we will penalise this case twice as hard as A_1 which in practise means that turns of this type are almost never found in an optimal solution. An alternative model could with some loss of generality instead explicitly forbid this type of turn by introducing constraints forcing $X_{ij} = 0$ whenever $(\underline{d_i} + t_i + r_{ij} - CT > \overline{d_j})$ which would in the general case reduce the number of booleans in the model to half. In practise, however, the gain is marginal since we introduce the booleans only where they are needed and this case is as already mentioned rare.
The case where $d_i + t_i + r_{ij} > 2CT$ can be safely ignored since we require the constants to fulfil $t_i + r_{ij} \leq CT$ and $t_i < CT$.

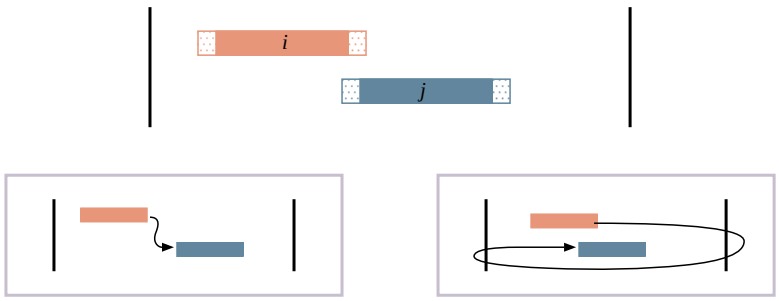
A more complex case occur when the time windows overlap so that the turn may or may not cross the cycle time boarder one or more times but the exact number depends on the assignment of the departure time variables.

For typical distributions of time windows in the rail freight case, we see mainly cases where the time window limits will place us in situations where we cannot determine if we are in case A_0 or A_1, only rarely whether we are in case A_1 or A_2 but almost never in ones where we cannot exclude at least one of the three. This fact can be used to reduce the number of boolean variables needed in the model significantly.

In the general case it is possible to distinguish the following sub-cases:

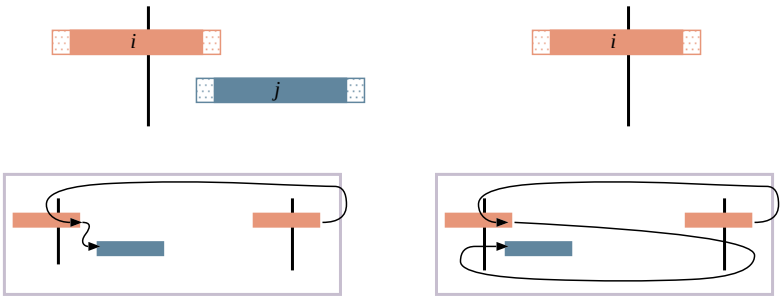
B_1 The turn may cross the cycle time limit once or not at all, i.e:

$$(\underline{d_i} + t_i + r_{ij} \leq \overline{d_j}) \wedge \left(\overline{d_i} + t_i + r_{ij} > \underline{d_j}\right) \wedge \left(\overline{d_i} + t_i + r_{ij} - CT \leq \underline{d_j}\right)$$



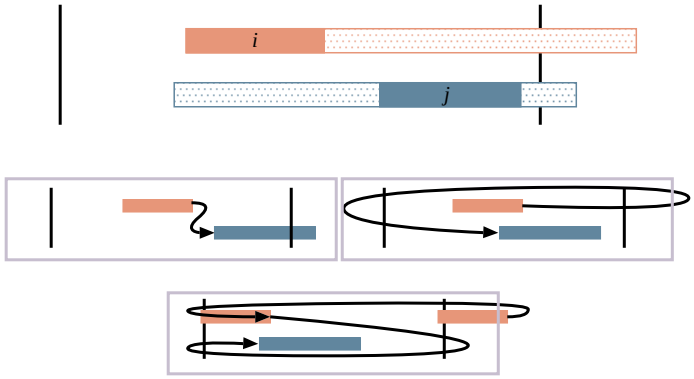
B_2 The turn may cross the cycle time limit twice but maybe only once. I.e:

$$(\underline{d_i} + t_i + r_{ij} > \overline{d_j}) \wedge (\underline{d_i} + t_i + r_{ij} - CT \leq \overline{d_j}) \wedge \left(\overline{d_i} + t_i + r_{ij} - CT > \underline{d_j}\right)$$



B_3 The turn may cross the cycle time border twice, once or not at all. I.e:

$$(\underline{d_i} + t_i + r_{ij} \leq \overline{d_j}) \wedge \left(\overline{d_i} + t_i + r_{ij} - CT > \underline{d_j}\right)$$



In the constraints given below we will *not* distinguish between these three subcases but treat them collectively as a single case $\mathcal{B}$ which will simplify the presentation of the model. In a practical implementation it *does* make sense to distinguish between them since we need to introduce two booleans per possible turn only in the B_3 case which is very rare.

5 Model constraints and objective

The cases labelled A_0 through A_2 above are all, if used as turns in a solution, determined to cross the cycle time limit either once, twice or not at all. The cases labelled B_i on the other hand are indeterminate and will be collectively encoded using the two boolean decision variables C_{ij} and C'_{ij}. To be able to treat the

A and B cases separately we will define four mutually exclusive subsets of the possible turns.

Let

$$\mathcal{A}_0 = \{\langle i,j\rangle \,|\, (0 < i,j \leq n) \wedge \overline{d_i} + t_i + r_{ij} \leq \underline{d_j}\}$$
$$\mathcal{A}_1 = \{\langle i,j\rangle \,|\, (0 < i,j \leq n) \wedge \left(\underline{d_i} + t_i + r_{ij} > \overline{d_j}\right) \wedge \left(\overline{d_i} + t_i + r_{ij} - CT \leq \underline{d_j}\right)\}$$
$$\mathcal{A}_2 = \{\langle i,j\rangle \,|\, (0 < i,j \leq n) \wedge \left(\underline{d_i} + t_i + r_{ij} - CT > \overline{d_j}\right)\}$$

and

$$\mathcal{B} = \{\langle i,j\rangle \mid 0 < i,j \leq n\} \setminus (\mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{A}_2)$$

We will need to explicitly represent the decision variables only for the case $\mathcal{B}$. Observe also that $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{B}$ have no elements in common.

Since the main objective of the model is to minimise the number of vehicles used by a solution and this corresponds exactly to the number of vehicles turned over the cycle time limit, the objective function will treat each of these cases (except $\mathcal{A}_0$ which can never contribute to the cost) separately. We also introduce a term in the cost function which penalises the use of additional vehicles for transports that do not need them. These are in most cases necessary to balance the flow of the model but should be avoided if possible. The penalty is weighted by the (temporal) length t_i of the transport and a factor p_i specific to each transport.

This factor should in most cases be smaller than 1 to give the number of vehicles the appropriate influence on the total cost. Potential *passive transports* (with vehicle demand $\underline{S_i} = 0$) will generally have a larger factor p_i than those in which a vehicle relocation is accompanying an existing transport.

Minimise

$$\Sigma_{\langle i,j\rangle \in \mathcal{A}_1} X_{ij} + \Sigma_{\langle i,j\rangle \in \mathcal{A}_2} 2X_{ij} + \Sigma_{\langle i,j\rangle \in \mathcal{B}} \left(Y_{ij} + Y'_{ij}\right) + \Sigma_{0<i\leq n} E_i p_i t_i$$

subject to

1. The number of vehicles turned *from* transport i is equal to the number used by it

$$\forall i((\Sigma_{j\in\{j|ld_i=lo_j\}} X_{ij}) - S_i = 0)$$

and the number of vehicles turned *to* transport j is equal to the number used by it

$$\forall j((\Sigma_{i\in\{ki|ld_i=lo_j\}} X_{ij}) - S_j = 0)$$

The candidate turns are chosen such that the destination ld_i of the source transport i and the origin lo_j of the sink j is identical. One way to relax this condition somewhat is described in section 6 below.

2. Turn time constraints.

$$\begin{cases} d_j - d_i + CT\ C_{ij} + CT\ C'_{ij} > t_i + r_{ij} \\ X_{ij} - Y_{ij} + M\ C_{ij} \leq M \\ X_{ij} - Y'_{ij} + M\ C'_{ij} \leq M \end{cases} \qquad \forall i, j\ (\langle i, j \rangle \in \mathcal{B})$$

and

$$S_i - E_i = \underline{S_i} \qquad \forall i$$

3. C_{ij}, C'_{ij} *boolean*, $C'_{ij} \leq C_{ij}$
4. S_i, X_{ij} (implicitly) *integer*
5. Variable bounds $\underline{d_i} \leq d_i \leq \overline{d_i}$, $\underline{S_i} \leq S_i \leq \overline{S_i}$, $\underline{X_{ij}} \leq X_{ij} \leq \overline{X_{ij}}$ for $\forall ij$

5.1 Constraint notes

The flow (conservation) constraints (1) ensure that each transport is supplied with as many vehicles as it needs and that the flow is balanced. To ensure that this is always possible we need to introduce a "sufficiently large" set of "potential" passive transports into the problem.

How this is done in general is not further discussed in this paper. However a straight forward heuristic to introduce additional such transports of a fixed maximum duration is outlined in section 6 below.

The turn time constraints (2) and their use of the boolean variables (3) are the core of the model. Note that $C_{ij} = C'_{ij} = 0$ if and only if $d_i + t_i + r_{ij} \leq d_j$, that $C_{ij} = 1 > C'_{ij}$ if and only if $d_i + t_i + r_{ij} - CT \leq d_j$ and finally that $C_{ij} = C'_{ij} = 1$ if and only if $d_i + t_i + r_{ij} - CT > d_j$ corresponding exactly to the three A-cases above. Note also that unnecessarily assigning 1 to C_{ij} while $X_{ij} > 0$ will be penalised by forcing Y_{ij} to become equal to X_{ij} and similarly for C'_{ij} and Y'_{ij}.

E_i is defined by the equation $S_i - E_i = \underline{S_i}$ to be the *excess* number of vehicles travelling with transport i. The requirement that $C'_{ij} \leq C_{ij}$ removes an obvious symmetry from the model. In practise the effect of this constraint is minor since the case where both booleans are needed, almost never occurs.

A key feature of the model and the main reason that it scales relatively well in practise is that the integrality constraints on S_i, and X_{ij} (4) need not be enforced by the solver. In each leaf in the search tree branching on the boolean variables C_{ij} and C'_{ij} the part of the coefficient matrix involving these variables will be a pure minimal cost flow). The same is obviously not the case for the part of the coefficient matrix involving the departure time variables d_i but since these variables are related to the decision variables S_i and X_{ij} only through the booleans (C_{ij}, C'_{ij}), each assignment of the d_i that is consistent with a complete (integral) assignment of the booleans will also be consistent with the optimal assignment to the decision variables S_i and X_{ij}.

This means that the optimal solution to the problem obtained by relaxing the integrality constraints on S_i and X_{ij} (but not on C_{ij} and C'_{ij}) will also be an optimal solution to the original problem.

5.2 Minimising deviation from a given timetable

In a setting where we start with a given time table and relax the departure and arrival times to allow a reduction of the total vehicle requirement of the schedule we may also want to minimise the deviation from the original time table starting from one in which the vehicle cost has been minimised. The following supplementary optimisation step can then be used for a given solution in which the Y_{ij}, Y'_{ij}, C_{ij} and C'_{ij} variables have have been determined. Let d_i^{orig} denote the original (unrelaxed) departure times and let the corresponding d_i variables have the same bounds as in section 5. Then introduce the following additional variables:

w_i	the amount of time that train i is changed with respect to the original time table.
w_i^-	the amount of time that the departure of train i is moved earlier
w_i^+	the amount of time that the departure of train i is moved later

Minimise

$$\sum_{0<i\leq N} w_i$$

subject to

1. $\forall i(d_i + w_i^- \geq d_i^{orig})$ determining the amount of time that transport i is early
2. $\forall i(d_i - w_i^+ \leq d_i^{orig})$ determining the amount of time that transport i is delayed
3. $w_i^+ + w_i^- - w_i = 0$ relating positive, negative and absolute movement of departure of train i
4. $\forall i, j(d_i - d_j \leq 2M - M * Y_{ij} - M * Y'_{ij} + CT * C_{ij} + CT * C'_{ij} - t_i - r_{ij})$ enforcing the turns of a previously determined solution.

In principle it should be possible to combine these two models into a third one weighing departure time deviation against vehicle cost but doing so in the straightforward way breaks the clean separation of the discrete decision variables X_{ij} and S_i on the one hand and the continuous departure time variables d_i on the other. In practise such a model does not scale at all well. In addition it would probably be very difficult in practise to determine suitable weights for the deviation variables w_i in the combined cost function. In the empirical results reported below the result of minimising the deviation for each given solution to the main problem is given in the deviation column.

6 System generated service trains

By changing the flow equations (1) in the above model, we can capture the introduction of (additional) passive transports (service trains), to reduce the overall need of vehicles somewhat.

Let $\text{dist}(l_1, l_2)$ be a function defining geographical distance between locations l_1 and l_2 and m a limit on the distance traversed by an (additional) passive transport. Replace the flow equations in the model above with:

$$\forall i((\Sigma_{j\in\{j|\text{dist}(ld_i,lo_j)\leq m\}}X_{ij}) - S_i = 0)$$

$$\forall j((\Sigma_{i\in\{i|\text{dist}(ld_i,lo_j)\leq m\}}X_{ij}) - S_j = 0)$$

The turn time r_{ij} for such turns i, j must be adapted to reflect the additional time taken to relocate the vehicle and the cost function extended with a term that reflects the cost for driving a service train:

$$\Sigma_{\langle i,j\rangle\in C} d_{ij} X_{ij}$$

where $C = \{\langle i, j\rangle \,|\text{dist}(lo_i, ld_j) > 0\}$ and d_{ij} is a cost factor that may or may not reflect the actual distance between ld_i and lo_j. In the experiments below m was set to one hour and each d_{ij} to a factor corresponding to $p_i t_i$ for a preplanned passive transport of duration t_i of one hour which in turn was twice of that of a vehicle accompanying an active transport of that duration.

The rest of this extended model is identical to that of section 5.

7 Empirical results

The following performance results have all been produced using data extracted from production data of the largest Swedish rail freight company Green Cargo. The case contains almost all transports handled by the most common type of vehicle in use, the electrical RC locomotive, and covers a full week. The problems solved below were generated by introducing a fixed amount of slack for each departure time in the production plan.

The number of transports in each of the generated problems is 1304. This includes a small number of statically generated potential passive transports of vehicles used to balance the flow in the network. In general relocation of vehicles are performed either as *accompanying transports* travelling with "real" transports or as *passive transports* (deadheads) where the vehicle travels by itself through the network. In both these cases the number of vehicles travelling with a scheduled transport exceeds the minimum required to perform the transport.

In the solutions reported below, *accompanying transports* has been freely introduced (though penalised) and moved around between transports that allow them. Passive transports on the other hand are eliminated wherever that leads to an improved objective. Dynamically generated new passive transports (as section 6) are introduced only in a separate set of problems and even then they are limited to a maximum traversal time of 60 minutes (while the average transport time is about four hours). Allowing additional passive transports in this way typically reduces the number of vehicles somewhat but introduce the need to schedule additional tasks on the infrastructure resources.

Note that introducing slack uniformly is not completely realistic. In reality customer requirements or limits on infrastructure capacity may not allow free rescheduling of the transports within their time windows. To some extent this can be improved by introducing individual slacks for each transport and weighted binary relations between arrival and departure events that encode e.g. transfers of cars and cargo. In the performance results reported here, no such additional constraints were used. Nevertheless a production version the software used to generate these problems is currently in use at Green Cargo in their planning of locomotives.

The tables below reports for each slack size (in minutes) the number of booleans needed to encode the turn time constraints which should give a rough indication of the MIP size. Furthermore the tables reports properties of the optimal solution found in terms of the number of vehicles, the total amount of "accompanying" and (additional) "passive" time in minutes. Performance results include the number of nodes, iterations and runtime in seconds as reported by CPLEX 9 for each slack size. The runtime reported is those reported by CPLEX on an 2.4 GHz Pentium 4 processor using about 2 GB of main memory. For the larger cases caching the node tree to disk was used whenever it became larger than the main memory. The strategy used was the default branch and bound (cut) heuristic of CPLEX 9 [19].

Once the optimal solution for the locomotive turnrounds has been found a new timetable is generated minimising the sum of deviations from the original timetable. This problem is linear and no performance results of these runs are given. The resulting deviation (in minutes) is given in each table to give an indication of how much the original time table had to be changed to achieve the corresponding improvement of the main objective.

The result reported in table 1 is without additional passive transports as outlined in section 5 while those in table 2 were obtained using the method outlined in section 6 to generate additional passive transports where this improved the objective.

Table 1. Without additional passive transports

Slack minutes	Booleans	Vehicles	Accompanying minutes	Deviation minutes	Nodes	Iterations	Runtime h:mm:ss
±0	-	117	50835	-	-	6803 (0)	0:00:05
±15	1027	116	50206	5107	0	7208	0:00:07
±30	1995	112	51107	13763	50 (1)	9056	0:00:54
±45	2836	105	51177	20841	40	10158	0:01:19
±60	3913	99	49402	35651	714 (251)	28565	0:08:22
±75	4930	97	49411	48486	21101 (3547)	630926	1:53:40
±90	5876	90	50385	69067	156441 (74849)	9245253	23:22:43

In both cases the ±0 case is for network solving, not MIP. The result reported for the ±90 cases was obtained with a integrality gap of 0.05% which guaranties a

Table 2. Allowing short additional passive transports

Slack minutes	Booleans	Vehicles	Passive minutes	Accompanying minutes	Deviation minutes	Nodes	Iterations	Runtime h:mm:ss
±0	-	116	276	49006	-	-	7105 (0)	0:00:07
±15	1127	113	247	50459	5274	0	7182	0:00:08
±30	2178	110	247	50429	15067	50	9323	0:01:06
±45	3132	104	116	51354	23456	50 (40)	10624	0:01:36
±60	4346	98	218	49264	40258	647 (435)	31029	0:08:53
±75	5456	95	203	50185	5541	14506 (1818)	403299	1:15:32
±90	6538	88	612	52293	64147	182364 (60220)	9870645	26:04:30

solution with an optimal number of vehicles and a solution within approximately 3 days of passive transport time from the optimum. Using the 0.01% default gap of CPLEX 9 gave a runtime about four times as long and only a marginal improvement of the objective.

8 Conclusions

The model presented generalises the m-TSPTW problem to multiple required and upper bounded optional visits to each location. It is shown how it can be applied to an important practical problem in rail transportation and that cases of realistic size can be solved using a standard (though state-of-the-art) commercial solver.

Innovative features of the model include the use of boolean variables to separate the integer and continuous parts of the problem and maintaining the flow character of the integer part of the problem for each complete assignment of the booleans.

Application of the model produces a modified timetable which accommodates the requirements for an efficient locomotive turn-round plan. The practical usefulness of the model and its scalability is demonstrated on a set of problems derived from a real case in the Swedish rail freight industry.

Significant savings can be realised for a uniform fleet of locomotives, in terms of locomotives planned, by utilising the presented method.

Future work would include identifying methods to compute better lower bounds and possibly to investigate if decomposition methods traditionally used for VRPTW type problems can be used to further improve the performance of solvers using the model. Multicommodity variants of the model may also be of interest.

References

1. Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F.: Time Constrained Routing and Scheduling. In: Network Routing. Volume 8 of Handbooks in Operations Research and Management Science. North-Holland (1995) 35–139

2. Drott, J., Hasselberg, E., Kohl, N., Kremer, M.: A planning system for locomotive scheduling. Technical report, Swedish State Railways, Stab Tågplanering, Stockholm, Sweden, and Carmen Systems AB (1997)
3. Solomon, M., Desrosiers, J.: Time window constrained routing and scheduling problems. Transportation Science **22** (1988) 1–13
4. Zwaneveld, P., Kroon, L., Romeijn, H., Salomon, M., Dauzère-Pérès, S., van Hoesel, S., Ambergen, H.: Routing trains through railway stations: Model formulation and algorithms. Transportation Science **30** (1996) 181–194
5. Ahuja, R.K., Liu, J.N., Orlin, James B.and Sharma, D., Shughart, L.A.: Solving real-life locomotive scheduling problems. Working Paper 4389-02, MIT Sloan (2002)
6. Mitrovic-Minic, S., Krishnamurti, R.: The multiple traveling salesman problem with time windows: Bounds for the minimum number of vehicles. Technical report, School of Computing Science, Simon Fraser University (2002)
7. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega (2006) 209–219
8. Cordeau, J.F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F.: Chapter 7: The VRP with Time Windows. In: The Vehicle Routing Problem [10]. SIAM Monographs on Discrete Mathematics and Applications,SIAM, Philadelpia, Pa. (2002)
9. Ball, M., Magnanti, T., Monma, C., Nemhauser, G., eds.: Network Routing. In Ball, M., Magnanti, T., Monma, C., Nemhauser, G., eds.: Handbooks in Operations Research and Management Science. Volume 8., North-Holland (1995)
10. Toth, P., Vigo, D., eds.: The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematis and Applications. SIAM (2002)
11. Savelsbergh, M.: Local search in routing problems with time windows. Annals of Operations Research **4** (1985) 285–305
12. Solomon, M.: On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. Networks **16** (1986) 161–174
13. Solomon, M.: Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research **35** (1987) 254–265
14. Desrochers, M., Lenstra, J., Savelsbergh, M., Soumis, F.: Vehicle routing with time windows: optimization and approximation. In Golden, B., Assad, A., eds.: Vehicle Routing: Methods and Studies, Amsterdam, North Holland (1988) 65–84
15. Desrochers, M., Desrosiers, J., Solomon, M.: Using column generation to solve the vehicle routing problem with time windows. Operations Research **18** (1990) 411–419
16. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. Operations Research **40** (1992) 342–354
17. Kohl, N., Madsen, O.B.G.: An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. Operations Research **45** (1997) 395–406
18. Potvin, J.Y., Rousseau, J.M.: An exchange heuristic for routing problems with time windows. Journal of the Operational Research Society **46** (1995) 1433–1446
19. ILOG: ILOG CPLEX Callable Library 9.0 Reference Manual. ILOG (2003)

Freight Service Design for an Italian Railways Company

Marco Campetella[1], Guglielmo Lulli[2],
Ugo Pietropaoli[3] and Nicoletta Ricciardi[3]

[1] Trenitalia SpA, Direzione Pianificazione Network e Sistemi Informativi
00161 Roma, P.zza della Croce Rossa 1, Italy
m.campetella@trenitalia.it
[2] Università di Milano "Bicocca", Dipartimento di Informatica, Sistemistica e Comunicazione
20126 Milano, Via Bicocca degli Arcimboldi 8, Italy
lulli@disco.unimib.it
[3] Università di Roma "La Sapienza", Dipartimento di Statistica, Probabilità e Statistiche Applicate
00185 Roma, P.le Aldo Moro 5, Italy
{ugo.pietropaoli, nicoletta.ricciardi}@uniroma1.it

Abstract. In this paper, we present a mathematical model to design the service network, that is the set of origin-destination connections. The resulting model considers both full and empty freight car movements, and takes into account handling costs. More specifically, the model suggests the services to provide, as well as the number of trains and the number and type of cars traveling on each connection. Quality of service, which is measured as total travel time, is established by minimizing the waiting time of cars at intermediate stations.
Our approach yields a multi-commodity network design problem with concave arc cost functions. To solve this problem, we implement a tabu search procedure which adopts "perturbing" mechanisms to force the algorithm to explore a larger portion of the feasible region. Computational results on realistic instances show a significant improvement over current practice.

Keywords. railways transportation, service network design, tabu search

1 Introduction

Railways freight transportation is a relevant activity in many economies. It supports and makes possible most other economic activities and exchanges. In the last twenty years, the privatization and reorganization of most of the national railways companies, combined with the removal of many entrance barriers and national protectionism to markets and the consequent increase of competition have boosted the development of Operations Research methods applied to railways transportation.

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/685

For comprehensive reviews on planning models for freight transportation, the reader may refer to the surveys by Crainic and Laporte [1] and by Cordeau, Toth and Vigo [2]. The former describes the main issues in freight transportation planning and operations and presents OR models, methods and tools to be used in this field of application. The latter is specifically focused on railways transportation. The authors provide a complete list of routing and fleet management models for freight transportation, of scheduling models for train dispatching and of assignment models to assign locomotives and cars.

The two cited surveys also give a broad overview on several types of problems and issues arising within the freight transportation arena. Obviously, most of railways transportation models available in the literature belong to the class of network flow models [3], since this is the topological structure of the problem. However, the general network flow model is often customized to take into account specific aspects of the problem. We here consider the freight service design problem, i.e. the problem of deciding the set of origin-destination connections. The reader may refer to [1] for a survey of the topic. This problem is a typical tactical planning problem, since the effects of decisions span a mid-term planning horizon.

A first example of a tactical planning model applied to rail freight transportation can be found in [4]. In this paper the authors examined traffic routing, train scheduling and allocation of work between yards. They also presented an optimization model intended to compute efficient solutions, allowing a reduction of costs, and to provide good quality of services in terms of transportation delays and reliability, over a medium term planning horizon. A nonlinear mixed-integer multi-commodity formulation is given and a heuristic algorithm is tested on instances arising from the Canadian National Railroads. Kwon et al. [5] presented a network flow model on a time-space network to model the problem of dynamic freight car routing and scheduling, which is solved with the column generation technique. Holmberg and Hellstrand [6] proposed a Lagrangean heuristic within a Branch & Bound framework to solve the uncapacitated network design problem with single origins and destinations for each commodity, which can be used to model the rail freight transportation. Fukasawa et al. [7] presented a network flow model to maximize the total profits, while satisfying the demands within a certain period of time, given the schedules and the capacities of the trains.

In railways freight transportation the movement of empty cars represents an important source of costs. Dejax and Crainic [8] provided a complete review of models for empty cars movement in freight transportation. Moreover we can mention some recently published papers in this specific area of research. Holmberg et al. [9] faced the problem of identifying distribution plans for the movement of empty cars by solving an integer multi-commodity network flow model on a time-expanded network. Sherali and Suharco [10] proposed a tactical model for the distribution and repositioning of empty railcars for shipping automobiles. Jaborn et al. [11] analyzed the cost structure for the repositioning of empty cars, and showed that such costs often depend on the number of car groups handled at yards. In the cost structure they also compounded costs

due to car classification at intermediate yards. They observed the economy-of-scale behavior of total distribution costs, which can be reasonably decreased by building fewer but larger car groups. The authors modeled the empty freight car distribution problem as a capacitated network design on a time-dependent network and solved it with a tabu search meta-heuristic.

The application of railways transportation models to real operations generally requires the solution of large size instances, thus calling for a compromise between solution quality and computational time. On this subject, meta-heuristic algorithms have been widely implemented. For instance, we can mention Marín and Salmerón [12], [13] and Gorman [14] among others. Marín and Salmerón [12], [13] compared the application of three different local search meta-heuristics to the tactical planning of rail freight networks. Gorman [14] discussed both genetic and tabu search meta-heuristics to solve the weekly routing and scheduling problem of a major US freight railroad.

Herein, we present a case study on the Italian freight railways transportation, whose features make it rather different from other systems and in particular from the North American reality. In Italy, freight trains run according to a fixed schedule, and they cannot exceed 20 cars, due to length and weight limits. Moreover, cars are generally managed individually and *blocks* (groups) of cars are not composed. The problem of identifying a blocking plan in order to minimize handling costs, known as *Railroad Blocking Problem*, has been recently faced by several papers. For instance, Newton et al. [15] and Barnhart et al. [16] presented two network design formulations and solved real instances by a column-generation algorithm and a heuristic Lagrangean approach, respectively. Ahuja et al. [17] developed a Very Large-Scale Neighborhood Search for the blocking problem and tested it on data provided by three major US railroad companies. All these papers showed that relevant savings can be obtained by establishing "good" blocking plans. Moreover, composing blocks of cars permits large economies of scale in North American systems, as pointed out in [11]. This is true for the American reality. On the contrary, in Italy blocks are not formed. At any classification yard, cars are uncoupled from the incoming train, reclassified and coupled again to a new outgoing train, if they have not reached their final destination. Therefore, at any intermediate yard, car handling implies a coupling and an uncoupling manoeuvre, with relevant impacts on the total transportation costs and delivery times. Blocks are not formed mainly for organizational reasons: at the moment, it is not possible to organize blocks of cars directed to the same destination and the automatism to hook cars, enabling engine-drivers to easily handling groups of cars, is not available on the Italian cars. Moreover, American blocks may easily exceed 20 cars, which is the standard size of an Italian train. Therefore, no economies of scale can be achieved in the Italian reality.

In every rail transportation system, to satisfy the transportation demand, empty car repositioning has to be handled. The movement of empty cars is quite relevant in Italy and it reflects the economy of the country, characterized by production districts concentrated mainly in the northern part. The movement

of empty cars is about 36% of the total. Besides the movement of empty cars, there is another relevant source of cost: the car handling process at intermediate yards. More than one third of the cars are coupled to at least three trains before arriving at their final destinations. Handling costs at intermediate yards represent a large percentage of total transportation costs: on average they contribute for one third to the total transportation cost borne to move a car from its origin to its final destination. Furthermore, handling a car at intermediate yards has also consequences on delivery time of goods. Taking into account such costs also guarantees a good level of service to customers by reducing delivery time.

In the sequel, we present a service network design model (refer to [18] for more details on the problem) for the Italian rail transportation, which is devoted to define the set of direct train connections between yards. The mathematical model we propose is tailored to the Italian system. We focus our attention on movements of both full and empty cars, taking into account the specific cost structure of the Italian freight railways transportation. We formalize a mathematical model classified as a Concave-cost Multi-commodity Network Design Problem which takes into account both the specific cost structure and the movements of both full and empty cars. In the proposed model, we also indirectly guarantee the quality of service, by minimizing the waiting time of cars at intermediate yards.

This problem is $\mathcal{NP}$-hard (see [19]). To solve real instances of the model, we propose a tabu search algorithm. We furnish the algorithm with additional mechanisms adopted to perturb the current solution with the scope of exploring a larger portion of the feasible region. The tabu search procedure takes advantage of the particular structure of the network, thus resulting an efficient algorithm as opposed to previous experience with similar problems.

The paper is organized as follows. In Section 2, we propose a mathematical formulation for the tactical problem of designing the network of services to move cars in order to satisfy transportation demand at minimum cost. Section 3 is dedicated to the description of the meta-heuristic algorithm designed in order to solve real instances of the problem. Section 4 reports some results of computational tests, and Section 5 concludes.

2 A mathematical model for tactical planning

Railways freight transportation problems are generally large-size and high level of difficulty problems. Operations Research-based methods can be useful in providing valid decision support systems to decision makers. We here propose a model to design the set of services of the Italian rail network in order to satisfy the whole transportation demand at minimum cost.

To formulate our model, we make use of the network of possible services, which is represented by a directed graph $D = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ denotes the set of yards and $\mathcal{A}$ the set of all the possible direct services. Since it is always possible to have a direct train between any pair of yards, we can reasonably assume that the following hypothesis holds true.

Hypothesis 1 *The graph representing the network of* possible *services is complete:* $\mathcal{A} \equiv \mathcal{N} \times \mathcal{N}$.

The purpose of the model that we propose is to design the network of activated services, namely $D' = (\mathcal{N}, \mathcal{S})$, which is a subgraph of $D = (\mathcal{N}, \mathcal{N} \times \mathcal{N})$, the complete (directed) graph defined on $n = |\mathcal{N}|$ vertices. More precisely, if $(i, j) \in \mathcal{S}$, then yard $i \in \mathcal{N}$ is directly connected to yard $j \in \mathcal{N}$ by a *service*, that is, at least one direct train connects i to j.

Cars move from their origins to their destinations using either a direct service, if one exists, or a sequence of trains with intermediate stops. In the Italian railways transportation operations blocks of cars are not allowed or at least they are extremely rare, see Section 1. Therefore, we can assume, without incurring in poor approximations, the following hypothesis.

Hypothesis 2 *If a train stops at an intermediate classification yard, all its cars are reclassified.*

In view of Hypothesis 2, we can directly associate handling costs to arcs $(i, j) \in (\mathcal{N} \times \mathcal{N})$ representing direct connections between pairs of yards $i, j \in \mathcal{N}$.

Finally, according to the experience of the Italian railways operator, capacity constraints are not restrictive at a tactical decision level, since enough cars to satisfy all the demands can be routed on each railroad; therefore, we are allowed to assume what follows.

Hypothesis 3 *The network of services is uncapacitated.*

The scope of the model we here propose is twofold. First, define the number of direct trains connecting origin-destination pairs. Second, route cars on the network in order to satisfy all the transportation demands. We consider both the full and the empty cars, and all the decisions are taken in order to minimize the total costs.

The notation of the model is in the sequel:

- $\mathcal{N} = 1, 2, \ldots, n$: set of yards;
- $\mathcal{K} = 1, 2, \ldots, k$: set of types of cars;
- f_{ij}: unit cost associated with the planning of a train on the direct service (i, j);
- c_{ij}: unit cost associated to link (i, j). This cost compounds the movement of a car on the link and the handling costs at yard i (coupling manoeuvre) and at yard j (uncoupling manoeuvre);
- b_i^p: supply (if positive) or demand (if negative) of cars of type p at yard i;
- α: maximum number of cars that can be assigned to a train.

The decision variables of the model are:

- x_{ij}^p: number of cars of type $p \in \mathcal{K}$ assigned to a service (i, j);

– y_{ij}: frequency of the service (i, j), i.e., number of trains traveling on the direct service (i, j).

The non-linear integer programming formulation is the following:

$$Min \sum_{(i,j)\in\mathcal{A}} f_{ij} \cdot h(y_{ij}) \cdot y_{ij} + \sum_{(i,j)\in\mathcal{A}, p\in\mathcal{K}} c_{ij} \cdot x_{ij}^p$$

s.t.

$$\sum_{j\in\mathcal{N}} x_{ji}^p - \sum_{j\in\mathcal{N}} x_{ij}^p = b_i^p \qquad \forall i \in \mathcal{N},\ \ \forall p \in \mathcal{K}. \tag{1}$$

$$\sum_{p\in\mathcal{K}} x_{ij}^p \leq \alpha \cdot y_{ij} \qquad \forall i, j \in \mathcal{N} : i \neq j. \tag{2}$$

$$x_{ij}^p \geq 0 \text{ integer} \qquad \forall i, j \in \mathcal{N} : i \neq j,\ \ \forall p \in \mathcal{K}. \tag{3}$$

$$y_{ij} \geq 0 \text{ integer} \qquad \forall i, j \in \mathcal{N} : i \neq j. \tag{4}$$

$$\tag{5}$$

where $h(y_{ij}) = \left(a + \frac{b}{y_{ij}+1}\right)$, with $a, b \in I\!R^+$.

This is a multi-commodity network design problem with a concave cost function. The cost function is composed by two terms: a train cost, which takes into account the planning of direct trains between origin-destination pairs, and a car cost.

The train cost is computed by multiplying the number y_{ij} of trains planned on a link (i, j) by a locomotive cost f_{ij} (computed as $f_{ij} = f \cdot d_{ij}$, where d_{ij} is the length of the link (i, j) and f is a train cost per kilometer, which accounts depreciation, maintenance, power, toll and engine drivers) and by a factor $h(y_{ij})$, which depends on the number of trains planned on the direct service (i, j). This last factor, which makes the objective function *concave*, is introduced in order to take into account a "quality" cost, which depends on the frequency of a service, i.e. on the number of available trains. In fact, an increase in the frequency of a service implies a decrease in the waiting times of cars at intermediate yards and therefore an increase in the quality of the service provided to the customers.

The car cost is computed by multiplying the total number of trains traveling on link (i, j) by a car unit cost c_{ij} that is computed as $c_{ij} = c \cdot d_{ij} + m_i + m_j$, where c is a car cost per kilometer which accounts both car depreciation and maintenance, and m_i, m_j are car handling costs at yards i and j, respectively.

It is worth noticing that all cost parameters are chosen according to the cost structure of the accounting system of the Italian railways company, and that we deliberately do not consider in the model the possibility to form blocks of cars, since at the moment this opportunity cannot be implemented in the Italian transportation reality.

Minimum concave-cost network flow problems are known to be $\mathcal{NP}$-hard (refer to [20] for a survey on algorithms and applications). Several solution techniques have been developed, both exact and approximate. The first ones explicitly or implicitly enumerate the vertices of the polyhedron defined by the network constraints, and are based on branch & bound, extreme point ranking methods or dynamic programming [20]. The second ones can be subdivided into two classes: heuristic and approximate algorithms. Heuristics find local optima using standard convex programming techniques; approximate algorithms underestimate the concave objective functions with piecewise linear functions (for instance, see [21] for a recent result).

We here propose a heuristic algorithm based on a tabu search, which computes good solutions in reasonable CPU time.

3 A heuristic algorithm

The size of any realistic instance of the service network design model presented in Section 2 is extremely large. Even instances which consider only classification yards lead to formulations with a number of constraints and variables of order of several hundreds of thousands. Given the computational complexity of the problem ($\mathcal{NP}$-hard), to solve the formulation and compute a good solution in reasonable time, we propose a heuristic algorithm based on a tabu search. The algorithm is furnished with a perturbing mechanism which alters the current solution in order to explore a larger portion of the solution space.

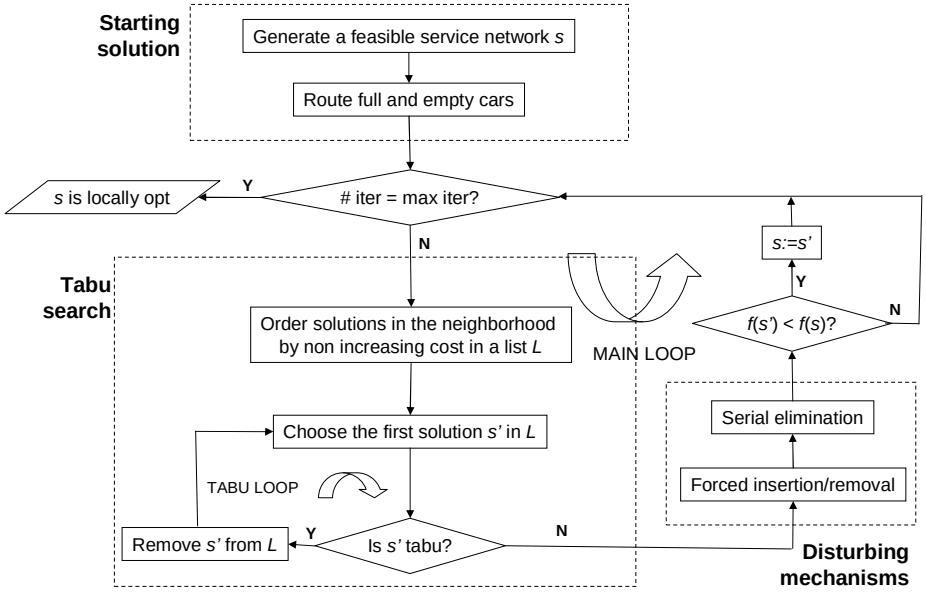


Fig. 1. Flow chart of the heuristic algorithm.

In our model, decisions can be classified into service decisions - deciding whether to activate or not a direct connection between a pair of yards of the network - and routing decisions. In the proposed heuristic we treat the two types of decisions in a hierarchical manner, i.e., we first fix the value of service decision variables and then we decide how to route cars using the activated services.

Finding a starting feasible solution. Our heuristic procedure is initialized with initial feasible solutions computed with the procedures reported in the sequel.

Minimum spanning tree: The initial solution is given by a minimum spanning tree (MST) found on a complete graph, whose arc costs r_{ij} are computed as follows:

$$r_{ij} = \frac{d_{ij}}{f_{ij}}, \quad \forall (i,j) \in \mathcal{N} \times \mathcal{N}$$

where f_{ij} represents the total demand for full cars from origin i to destination j, and d_{ij} is the length of the link (i, j). Observe that the MST minimizes the number of services, i.e. direct links between specified origins and destinations.

Currently used logistic network: We use the solution currently implemented, which is computed on practitioners' experience.

"Complete" graph: We compute a solution which satisfies all the demands with a direct service. This solution clearly minimizes the car handling costs.

Minimum distance sub-graph: We compute a feasible solution selecting arcs in a minimum length order until a connected sub-graph is obtained.

In the computational analysis we verified that the algorithm shows better performances when the minimum spanning tree or the currently used logistic network are chosen as initial feasible solutions.

Once the initial feasible service network is determined, we route the cars minimizing the overall cost. Observe that we deal with full and empty cars separately, since they have a different feature. Full cars are characterized by a fixed O/D pair, while empties have to satisfy only constraints on the type: any demand for empties can be satisfied with cars located at any origin. As observed in Section 2, the service network is uncapacitated and we can schedule as many trains as needed on any service. Therefore, the flow of full cars can be computed using a shortest path algorithm for each commodity. In particular, we implement a Floyd-Warshall algorithm. As regards the routing of empty cars, we solve an uncapacitated minimum cost flow problem. In particular, we solve a minimum cost flow for each commodity.

Tabu search procedure. As we have already mention, the core routine of our heuristic is a tabu search procedure, see [22] and [23] for details on tabu search. We implement as notion of neighborhood of the current solution within search for a better solution, the set of all the networks that can be obtained from the current

service network adding or deleting an arc. Adding an arc corresponds to opening a new service, while deleting an arc corresponds to closing the corresponding service.

For each possible "move" from the current solution to a new one in the neighborhood we have to evaluate the cost of such solution. This requires solving the routing sub-problem of cars, both empties and fulls. This procedure can be very time consuming and it may lead to high computational time. To overcome this drawback, we use estimates of such costs which are easily computed. Let us evaluate the cost of a solution obtained by opening a new direct service (arc) between nodes i and j. In this case all the flow going from i to j will be re-routed on the directed arc (i, j). Given the new flow vector, the new cost of the objective function is obtained simply adding up the new total transport and the new car handling costs. In closing a service, we first have to guarantee that the deletion of the arc leaves the graph connected. Therefore we only consider among all the arc deletions those which maintain graph connectivity. When the arc is removed, all the flow which previously used arc (i, j), is re-routed on the shortest path connecting i to j.

These procedures are fast and easy to implement; in fact, they do not compute the vector of flows (routing of cars) *from scratch*, but they focus their attention on the O/D pair $i-j$ whose connection status has changed. However, these estimates may clearly be non-exact. It is worth noticing that, once the most convenient move is determined based on the cost estimates, the new service network (new solution) is considered, and full and empty cars are routed optimally according to the procedures discussed above, thus computing the *optimal* vector of flows and the *exact* cost value.

Two different data structures are used in order to keep memory of the more recently visited solutions: a tabu list $TL1$ and a tabu list $TL2$. $TL1$ is a l-dimensional list which is filled with the last l solutions visited by the algorithm, while $TL2$ is an h-dimensional list which contains the last h moves performed by the procedure. The dimensions of the lists are read as input parameters at the beginning of the procedure. Once the number of solutions belonging to $TL1$ (respectively, $TL2$) is equal to its dimension l (respectively h) and it becomes necessary to add a new solution to the list, the maximum cost solution in the list is deleted. The algorithm ends when a maximum number of iterations is achieved.

Perturbing mechanisms. To visit a larger portion of the feasible region, we furnish our tabu search algorithm with a perturbing mechanism in the search procedure. Whenever the current optimal solution is not updated within a certain (fixed) number of iterations, the algorithm is forced to perform a move. The move consists in adding (*forced insertion*) or deleting (*forced removal*) an arc. The arc added or removed is the one that gives the highest improvement (or the lowest worsening) in the objective function, among a set of candidate arcs. The set of candidate arcs is composed by a certain (fixed) number of less recently selected arcs. Again, if a removal is performed, network connectivity has to be verified.

We also introduce another perturbing mechanism, referred to as *serial elimination* whose scope is to let the algorithm to "re-start from zero". Whenever the current optimal solution is not updated within a certain (fixed) number of iterations, the serial elimination forces the algorithm to re-start the local search from a service network which is composed by a "small" number of direct arcs. The procedure forces the algorithm to remove arcs until the solution becomes composed by a small fixed number of arcs. Again, removed arcs are those preserving network connectivity and providing the highest improvement (or the lowest worsening) in the objective function.

4 Computational experience on a real instance

The proposed heuristic algorithm has been tested on both a set of random generated instances and on a real instance relative to the Italian case. In particular, we considered a set of 25 random generated "small" instances, which we solved using a mixed-integer programming solver based on the Branch & Bound algorithm, called MINLP-B&B. All random generated instances have been solved to optimality by the Branch & Bound procedure quickly. We compared these optimal solution with the solutions proposed by our tabu search heuristic, and we verified that in most cases our algorithm finds the optimum and that the deviation from the optimal solution is rather low (1.78 %, on average).

The real-world instance has been obtained by an elaboration of the data contained into the historical databases, and into the accounting systems of our industrial partner. The maximum number of cars which can be assigned to a train has been set to 20. The parameters defining the multiplicative factor which gives the non-linearity of the objective function are fixed according to cost criteria which are currently used in the accounting systems. We restrict our analysis to a network composed by a set of 39 nodes, which correspond to the set of all the major classification yards and frontier passes. The number of commodities is 375. The main characteristics of the considered instance are reported in Table 1.

Table 1. Characteristics of a real instance of the problem.

	Real instance
Number of instances	1
Number of yards (n)	39
Number of commodities (p)	375
Distances (Km)	0 to 2000
Demands (number of cars)	-2000 to 2000
Car handling cost (euro)	0 to 50

In Table 2 we compare the solution proposed by our algorithm with the solution currently implemented by the Italian railways company. In particular, we consider the following technical and economical statistics.

1. C: it is the objective function value and represents a measure of the total cost.
2. S: number of directed services between yards.
3. T: number of trains necessary to deliver all the cars on the service network in order to satisfy all the transportation demands.
4. TK: total amount of kilometers covered by all the trains traveling on the network.
5. CK: total amount of kilometers covered by all the cars traveling on the network.
6. M: number of manoeuvres.

Table 2. Comparison between the currently used logistic network and the network proposed by the heuristic algorithm.

Statistic	Current network	Algorithm solution	% Comparison
C	11767851	11318144	-3.82%
S	144	158	+9.72%
T	4258	3851	-9.56%
TK	1060889	1027310	-3.17%
CK	21217665	20546329	-3.16%
M	170390	154014	-9.61%

It can be observed that total costs decrease of 3.82%, as well as the number of trains (-9.56%), car manoeuvres (-9.61%), and the total amount of kilometers covered by all the trains (cars, respectively) traveling on the service network (-3.17 and -3.16%, respectively). On the other hand, we have an increase in the number of directed services which are activated in the solution proposed by the algorithm ($+9.72\%$).

5 Conclusions

The purpose of this paper is two-fold. First, we focus on the current Italian rail transportation reality. Several different features identify the Italian system with respect to others, especially the Northern American ones. Relevant differences arise with respect to the planning process, the structure of railroads, the limits in terms of maximum weight and length of a train and finally in the policies adopted to compose trains. Furthermore, in Italy both empty cars distribution and cars handling at intermediate yards have a relevant impact on total transportation costs.

Second, we proposed a mathematical model intended to design the set of direct connections between yards, and to route cars on it. The model combines both the full and empty freight cars management and takes into account the handling costs, suggesting the services to provide and the number of trains and the number and type of cars traveling on each service. In order to ensure a certain level of quality of the service which is offered to the customers, a concave cost objective function is introduced. The resulting model, which can be classified within the class of minimum concave-cost network design problems, is solved with a tabu search meta-heuristic which adopts some perturbing mechanisms in order to force the search process to explore a larger portion of the feasible region.

The computational results showed a good behavior of the algorithm on realistic instances of the problem, thus proving the viability for an integration in a decision support system.

References

1. Crainic, T., Laporte, G.: Planning models for freight transportation. European Journal of Operational Research **97** (1997) 409–438
2. Cordeau, J., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation Science **32** (1998) 308–404
3. Ahuja, R.K., Magnanti, T., Orlin, J.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs, New Jersey (1993)
4. Crainic, T., Ferland, J., Rousseau, J.: A tactical planning model for rail freight transportation. Transportation Science **18** (1984) 165–184
5. Kwon, O., Martland, C., Sussman, J.: Routing and scheduling temporal and heterogeneous freight car traffic on rail networks. Transportations Research - E **34** (1998) 101–115
6. Holmberg, K., Hellstrand, J.: Solving the uncapacitated network design problem by a lagrangean heuristic and branch-and-bound. Operations Research **46** (1998) 247–259
7. Fukasawa, R., de Aragão, M.P., Porto, O., Uchoa, E.: Solving the freight car flow problem to optimality. Electronic Notes in Theoretical Computer Science **66** (2002)
8. Dejax, P., Crainic, T.: A review of empty flows and fleet management models in freight transportation. Transportation Science **21** (1987) 227–247
9. Holmberg, K., Joborn, M., Lundgren, J.: Improved empty freight car distribution. Transportation Science **32** (1998) 163–173
10. Sherali, H., Suharko, A.: A tactical decision support system for empty railcar management. Transportation Science **32** (1998) 306–329
11. Jaborn, M., Crainic, T., Gendreau, M., Holmberg, M., Lundgren, J.: Economies of scale in empty freight car distribution in scheduled railways. Transportation Science **38** (2004) 121–134
12. Marín, A., Salmerón, J.: Tactical design of rail freight networks. part i: Exact and heuristic methods. European Journal of Operational Research **90** (1996) 26–44
13. Marín, A., Salmerón, J.: Tactical design of rail freight networks. part ii: local search methods with statistical analysis. European Journal of Operational Research **94** (1996) 43–53

14. Gorman, M.: An application of genetic and tabu searches to the freight railroad operating plan problem. Annals of Operations Research **78** (1998) 51–69
15. Newton, H., Barnhart, C., Vance, P.: Constructing railroad blocking plans to minimize handling costs. Transportation Science **32** (1998) 330–345
16. Barnhart, C., Jin, H., Vance, P.: Railroad blocking: A network design application. Operations Research **48** (2000) 603–614
17. Ahuja, R.K., Jha, K., Liu, J.: Solving real-life railroad blocking problems. Technical report, Department of Industrial and Systems Engineering, University of Florida (2004) Submitted to *Transportation Research*.
18. Crainic, T.: Service network design in freight transportation. European Journal of Operational Research **122** (2000) 272–288
19. Magnanti, T., Wong, R.: Network design and transportation planning: Models and algorithms. Transportations Science **18** (1986) 1–55
20. Guisewite, G., Pardalos, P.: Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Annals of Operations Research **25** (1990) 75–100
21. Magnanti, T., Stratila, D.: Separable concave optimization approximately equals piecewise linear optimization. In Nemhauser, G., Bienstock, D., eds.: Integer Programming and Combinatorial Optimization, 10th International IPCO Conference, New York, NY, USA, June 7-11, 2004, Proceedings. Volume 3064 of Lecture Notes in Computer Science., Springer (2004)
22. Glover, F.: Tabu search part i. Journal on Computing **1** (1989) 190–206
23. Glover, F.: Tabu search part ii. Journal on Computing **2** (1990) 4–32

Locomotive and Wagon Scheduling in Freight Transport

Armin Fügenschuh[1], Henning Homfeld[1],
Andreas Huck[2] and Alexander Martin[1]

[1] Technische Universität Darmstadt, Arbeitsgruppe Optimierung
64289, Darmstadt, Schlossgartenstr. 7, Germany
{fuegenschuh,homfeld,martin}@mathematik.tu-darmstadt.de
[2] Deutsche Bahn AG, Konzernentwicklung, GSU 1
60326, Frankfurt, Stephensonstraße 1, Germany
andreas.huck@bahn.de

Abstract. We present a new model for a strategic locomotive scheduling problem arising at the Deutsche Bahn AG. The model is based on a multi-commodity min-cost flow formulation that is also used for public bus scheduling problems. However, several new aspects have to be additionally taken into account, such as cyclic departures of the trains, time windows on starting and arrival times, network-load dependend travel times, and a transfer of wagons between trains. The model is formulated as an integer programming problem, and solutions are obtained using commercial standard software. Computational results for several test instances are presented.

Keywords. Freight Transport, Vehicle Scheduling, Time Windows, Integer Programming.

1 Introduction

Deutsche Bahn AG (DB) is the largest German railway company with 216,000 employees and a turnover of 25 billion Euros in 2005. DB is active in both passenger and freight transportation. Per year, 1.8 billion passengers (72 billion passenger kilometers) and 253 million tons of goods (77 billion ton kilometers) are transported. Moreover, DB is the owner of the German railway system, where DB freight and passenger trains travel 887 million kilometers per year, and external railway companies around 110 million kilometers. The overall length of the railways is 34,000 kilometers, and about 4,400 freight trains and 30,000 passenger trains per day traverse this network [1,2]. All in all, DB's network is considered as one of the most dense and most frequently used railway network in the world.

For the long-term simulations and future predictions of the network load, DB developed a complex simulation tool. The entire simulation tool can be considered as a chain, which decomposes into several components. To this end it is possible to model the normal course of business operations, and to analyze

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/686

the influence of changing external parameters, such as future demands on goods, possible network expansions or the sensitivity to the price for oil. The components of the tool chain interact by generating output data, which is used as input for other parts of the chain. The entire simulation tool has evolved over the last 5 years, and is still under continuous improvement. In this article we describe the development of a new segment for the tool chain.

Currently, a chain's segment called *train scheduler* is responsible for the generation of trains from individual wagons. As soon as enough wagons are assembled, the train is started. That means, the starting times of the trains are not aligned to some timetable, they just follow the estimated customers' productions and demand peaks in the simulation. Hereby it is assumed that a locomotive for pulling this train is always and immediately available. Moreover, the locomotives are not scheduled.

From the area of public bus transport it is known that routing and scheduling of the vehicles is an important field in the optimization of the operator's business process. For instance, Löbel [3] and Gintner, Kliewer, and Suhl [4] developed models for the scheduling of public buses, which led to significant cost savings in public transport. Moreover, it was noted by Daduna and Völker [5] that in public bus transport an even fewer number of buses is necessary to serve all trips, if the starting times of the vehicles are altered within some small interval. Later, Fügenschuh [6] also included the customers' demands into the optimization such that the scheduling problem of the vehicles is solved together with the starting time problem of the trips, which leads to further reductions of vehicles and costs.

The scope of our research is to carry over these observations to the locomotive scheduling in freight transport of Deutsche Bahn. Several new aspects have to be taken into account, such as cyclic departures of the trains, time windows on starting and arrival times, network-load dependend travel times, and a transfer of wagons between trains. The model presented in this article aims at a support of strategic simulations of the future, for example, simulating the network load in freight transport in the year 2015. The model is formulated as a linear integer programming problem (IP, for short). We give a computational evaluation of the resulting IPs and show whether standard commercial IP solvers (such as ILOG Cplex [7]) are able to handle problem sizes of instances that occur in the context of DB.

The remainder of this article is organized as follows. In Section 2 we describe the problem in greater detail. In Section 3 we provide a stepwise refined model, formulated as an integer programming problem. In Section 4 we present computational results for the different varients of our model using standard software. An outlook to further work is finally given in Section 5. For a survey on combinatorial optimization problems in connection with rail transport we refer to the literature, for instance, the survey articles of Bussieck, Winter, and Zimmermann [8] and Caprara, Fischetti, Toth, and Vigo [9].

2 The Problem Settings

In this section we give details of the integrated scheduling problem and introduce the terminology used at DB.

Wagons. A *wagon* is a rolling stock for freight transport. The wagons have to be delivered between a source and a destination point (goods station) within the network. Large customers produce and/or consume so much goods that they order whole trains. In these cases, the route of the wagons equals the route of the train. Smaller customers order individual wagons. Then the wagons of different customers are assembled to trains and pulled as a whole to an intermediate destination (a shunting yard), where the trains are disaggregated and reassembled to new trains. The trains and the yards where the wagon transfer between trains are known in advance. When changing the starting time of the trains, one has to take care that these transfers still remain feasible.

Trains. A *freight train* (also called *production trip*) consists of several *wagons*. Each train has a *start* and a *destination*, which are goods stations or railroad shunting yards. Also given are *starting times* and *arrival times*. These can be either fixed times or intervals, in which the start or the arrival has to take place. We assume that the trains start cyclical every 24 hours. The *trip duration* is the time difference between start and arrival. The average travel speed of freight trains is not as high as in passenger transport, especially at daytime, when passenger trains always have priority, such that some trips can last up to 3 days. The trains have different length and weight and thus require locomotives with sufficient driving power. In contrast to the problems described by Ahuja et al. [10] or Ziarati [11], a train is always pulled by a single locomotive. At the start a locomotive is attached to the train, and at the destination station it is detached (uncoupled). For both *coupling* processes, a certain train-dependent amount of time has to be taken into account (15 to 30 minutes). At this stage, technical checks and refueling of diesel locomotives are carried out.

Locomotives. DB uses up to 30 different locomotives of several manufacturers. However, the differences between them are often minor, so they are grouped into 3 to 6 *classes* of similar locomotives. The main differences among the classes are the driving power of the engines, and the *traction* (i.e., the motor type, diesel or electrical). Electrical locomotives can only be used on electrical tracks, whereas diesel locomotives in principle can drive everywhere. However, diesel soots the electrical wires, so one wants to avoid their deployment on such tracks. Hence, it is only possible to assign such locomotives to trains that have a sufficient power and the right traction for the track.

Deadheads. A locomotive is either active, i.e., pulling a train, or deadheading, i.e., driving under its own power without pulling a train from the destination station of one train to the start of another train. The duration for a deadhead trip depends on the distance between these two points, and on the class of the locomotive (diesel and electrical might have to use different routes), but

not so much on the network load (i.e., independent of daytime or nighttime), because it is assumed that a single locomotive can always be pushed through.

Goals and Objectives. The main goal is to compute feasible starting and arrival times of the trains such that the wagons are transported as fast as possible from their start to the destination within the trains. At intermediate shunting stations the stopover of wagons should not exceed certain limits. The main objective is to reduce operating expenses, that is, to use as few locomotives as possible to pull all trains and, on a subordinate level, to schedule the locomotives in such a way that the deadhead trips are as short as possible.

3 Models

The models we describe in this section can be classified by one main characteristic, the starting time intervals. In the first type of the models, the starting time is given by the pre-scheduler of the tool chain. By this tool, a train is started as soon as enough wagons are assembled. What "enough" in this context means is guided by a local criterion, which is mainly based on the number, total weight, and total length of the wagons. However, this local criterion does not take the availability of locomotives into account. It is simply assumed that a locomotive is always available, if required by some train. A model for scheduling the locomotives under this assumptions is presented in Section 3.1.

On the other hand, there is always a little flexibility in the departure and arrival of the trains, which has to be negotiated with the customers. In Section 3.2 we describe a model where the starting time can vary within a given interval. This model is much more complex, since it has to take care of the syncronization of the train departures and the wagon schedules. A further refinement of this model is given in Section 3.3. Here the trip durations are not constant but dynamically depending on the actual network load.

For all models we use the following notations. Let $\mathcal{V}$ be the set of freight trains (production trips), and let $\mathcal{B}$ be the set of locomotive classes. We introduce a parameter $a_{b,i} \in \{0,1\}$ with $a_{b,i} = 0$ if a class b locomotive cannot pull train i. Let $\mathcal{A} := \mathcal{V} \times \mathcal{V}$ denote the set of all deadhead trips. Denote $a_{b,(i,j)} := a_{b,i} \cdot a_{b,j} \in \{0,1\}$, then we have $a_{b,(i,j)} = 0$ if and only if the deadhead trip from i to j is not feasible for a class b locomotive. Moreover, those $a_{b,(i,j)}$ can be set to zero where the corresponding deadhead trip exceeds a certain length.

3.1 Fixed Starting Times

To begin with, we take the starting and arrival times for the trains as they were computed by the train scheduler of the tool chain. This computation also ensures that individual wagons can transfer between trains. We introduce a decision variable $x_{b,(i,j)} \in \{0, a_{b,(i,j)}\}$ with $x_{b,(i,j)} = 1$ if trains i and j are connected and both served with a locomotive of class b, and $x_{b,(i,j)} = 0$ otherwise. Each train j

must be served with one class of locomotive, that is,

$$\sum_{b\in\mathcal{B}}\sum_{i:(i,j)\in\mathcal{A}} x_{b,(i,j)} = 1. \tag{1}$$

There is a flow conservation in the sense that the cycles of each class b and each trip j must be closed, that is,

$$\sum_{i:(i,j)\in\mathcal{A}} x_{b,(i,j)} = \sum_{k:(j,k)\in\mathcal{A}} x_{b,(j,k)}. \tag{2}$$

The connection of production and deadhead trips is called *cycle*. This notion is justified since each trip has a unique predecessor and a unique successor, and therefore at some point each locomotive will serve the first trip again. We denote by $\lambda_{b,(i,j)}$ the number of locomotives of class b that are additionally necessary due to the connection of i with j. Similar to Liebchen and Möhring [12] this number is computed as

$$\lambda_{b,(i,j)} := \left\lceil \frac{\hat{t}_i - \hat{t}_j + \delta_{b,(i,j)}}{1440} \right\rceil \geq 0. \tag{3}$$

Here $\hat{t}_i, \hat{t}_j$ are the pre-scheduled starting times of trains i and j, respectively. The constant 1440 refers to the number of minutes per day, which is the basis of the cycles (i.e., all trips are repeated on a daily base), and $\delta_{b,(i,j)}$ denotes the total trip and deadhead trip duration, that is,

$$\delta_{b,(i,j)} := \delta_i^{\text{trp}} + \delta_i^{\text{uncpl}} + \delta_{b,(i,j)}^{\text{dhd}} + \delta_j^{\text{cpl}}, \tag{4}$$

where

- δ_i^{trp} denotes the trip duration, i.e., the time the locomotive is active while pulling train i,
- δ_i^{uncpl} denotes the time for uncoupling the locomotive from the train at the arrival,
- $\delta_{b,(i,j)}^{\text{dhd}}$ denotes the time for deadheading from the end of i to the start of train j, and
- δ_j^{cpl} denotes the time for coupling the locomotive to the train at the start of j.

Remark that the driving time δ_i^{trp} is assumed to be independent of the actual class, whereas the deadhead time $\delta_{b,(i,j)}^{\text{dhd}}$ is class dependent (since diesel and electrical might use different routes).

A *capacity* in form of an upper bound B_b on the number of available locomotives of class b can be specified:

$$\sum_{(i,j)\in\mathcal{A}} \lambda_{b,(i,j)} x_{b,(i,j)} \leq B_b. \tag{5}$$

As objective function we want to minimize the total costs defined as

$$\sum_{b\in\mathcal{B}}\sum_{(i,j)\in\mathcal{A}}(\gamma_b^{\mathrm{cls}}\lambda_{b,(i,j)}+\gamma_{b,(i,j)}^{\mathrm{dhd}})x_{b,(i,j)} \tag{6}$$

where

- γ_b^{cls} denotes the costs for a locomotive of class b,
- $\gamma_{b,(i,j)}^{\mathrm{dhd}}$ denotes the costs in connection with deadheading from i to j with locomotive b.

The most important objective is the reduction of the deployed locomotives, and second is the reduction of deadhead costs. Hence $\gamma_b^{\mathrm{cls}} \gg \gamma_{b,(i,j)}^{\mathrm{dhd}}$.

The optimization problem is the minimization of (6) subject to the constraints (1), (2), (5), and the integrality of all $x_{b,(i,j)}$. Due to the cyclic character of the schedules of the locomotives we call this problem the *capacitated cyclic vehicle scheduling problem* (CVSP). In the case of no upper bounds (or $B_b = \infty$) we also speak of the *uncapacitated* CVSP.

For $|\mathcal{B}| = 1$, this problem reduces to a single-commodity minimum-cost flow problem, which can be solved efficiently by polynomial or pseudopolynomial algorithms (for instance by the Hungarian Method, see Ahuja, Magnanti, and Orlin [13] for details). For $|\mathcal{B}| > 1$, the uncapacitated CVSP is a multi-commodity min-cost flow problem, which is known to be NP-hard. Moreover, it is NP-complete to decide whether a feasible solution exists for the capacitated CVSP (see Löbel [3]).

3.2 Variable Starting Times

We change the above model for locomotive scheduling for the case where the starting times of the trains are allowed to be changed within a given interval. The corresponding model is called (capacitated or uncapacitated) *cyclic vehicle scheduling problem with time windows* (CVSPTW). In comparison with the CVSP, the CVSPTW model is more complicated, because we have to take care of the transfer of wagons between trains.

We first introduce bounds on the starting time of the trains. The starting time for $i \in \mathcal{V}$ is denoted by $t_i \in \mathbb{Z}_+$. The set $\mathcal{V}$ is devided into two subsets, $\mathcal{V} = \mathcal{C} \dot\cup \mathcal{S}$. In $\mathcal{C}$ there are all trips i with a connected starting time interval $[\underline{t}_i, \bar{t}_i] \subseteq [0, 1439] \cap \mathbb{Z}$ in which the starting time must lie:

$$\underline{t}_i \le t_i \le \bar{t}_i. \tag{7}$$

In particular, each trip has to start during the first day. If the starting time exceeds this limit, the interval is split into two. This is the case for all trips $i \in \mathcal{S}$. Their starting time must be in $([\underline{t}_i, 1439] \cup [0, \bar{t}_i]) \cap \mathbb{Z}$. We introduce a binary variable $y_i \in \{0, 1\}$ with $y_i = 1$ if the starting time is in $[0, \bar{t}_i]$ (after midnight), and $y_i = 0$ if it is in $[\underline{t}_i, 1439]$ (before midnight). We then obtain the following constraints for the starting times:

$$\underline{t}_i(1 - y_i) \le t_i \le 1439 + (\bar{t}_i - 1439)y_i. \tag{8}$$

As in the CVSP model we introduce decision variables $x_{b,(i,j)} \in \{0, a_{b,(i,j)}\}$ for the deadhead trips, and use the same multi-commodity flow formulation:

$$\sum_{b \in \mathcal{B}} \sum_{i:(i,j) \in \mathcal{A}} x_{b,(i,j)} = 1, \tag{9}$$

and

$$\sum_{i:(i,j) \in \mathcal{A}} x_{b,(i,j)} = \sum_{k:(j,k) \in \mathcal{A}} x_{b,(j,k)}. \tag{10}$$

If $i, j \in \mathcal{V}$ are connected, then the corresponding starting times must be synchronized,

$$t_i + \delta_{b,(i,j)} - 1440 l_{b,(i,j)} \leq t_j + 5760(1 - x_{b,(i,j)}). \tag{11}$$

The constant $5760 = 4 \cdot 1440$ reflects the assumption that the train starting time is fixed to the first day, and a trip duration per train is at most 3 days. Since the starting time selection is now integrated in the model, we cannot compute the number of locomotives $\lambda_{b,(i,j)}$ beforehand, as in the case of the CVSP. Instead we introduce a variable $l_{b,(i,j)} \in \mathbb{Z}_+$ which represents the number of additional locomotives due to the connection of i with j.

Finally we have to syncronize those trains $i, j \in \mathcal{V}$ where wagons transfer from one to the other. Since all trains are cyclic, there are in principle no missed transfers. That is, if j departs before the arrival of i, then the wagons have to wait at most 24 hours until the arrival of the next train j. However, long idle waiting times of the wagons are undesired. This is modeled by the following inequality:

$$0 \leq (t_j + 1440 q_{i,j}) - (t_i + \delta_i^{\text{trp}} + \delta_{i,j}^{\text{shnt}}) \leq 719 + 720 p_{i,j}, \tag{12}$$

where

- $\delta_{i,j}^{\text{shnt}}$ denotes the time for shunting the wagon from i to j,
- $q_{i,j} \in \{0, \ldots, 4\}$ is a variable to shift the starting time of j within the same day of the arrival of i, and
- $p_{i,j} \in \{0, 1\}$ is a decision variable with $p_{i,j} = 1$ if and only if the wagons from i to j are idle for more than 12 hours.

In this formulation, the variables $p_{i,j}$ are put into the objective function with a suitable scaling coefficient. In this way it is possible to analyze how many locomotive could potentially be saved if some transfers are missed. It is also possible to set $p_{i,j} := 0$ for all transfers i, j, which gives a hard constraint, i.e., the transfers are much more important than saved locomotives.

The objective is to minimize the total costs defined as

$$\sum_{(i,j) \in \mathcal{A}} \gamma_{i,j}^{\text{idle}} p_{i,j} + \sum_{b \in \mathcal{B}} \sum_{(i,j) \in \mathcal{A}'} (\gamma_b^{\text{cls}} l_{b,(i,j)} + \gamma_{b,(i,j)}^{\text{dhd}} x_{b,(i,j)}), \tag{13}$$

where γ_b^{cls} and $\gamma_{b,(i,j)}^{\text{dhd}}$ are defined as above, and $\gamma_{i,j}^{\text{idle}}$ denotes the costs for idle wagons, i.e., a wagon missing its subsequent train which then has to wait for

more than 12 hours. In general, these coefficients reflect the following ordering: Reducing idle wagons is most important, since high contract penalties for late arrivals have to be paid. Second is now the reduction of the deployed locomotives, and the reduction of deadhead costs is moved to the third place.

3.3 Netload-dependent Travel Times

We further refine the above CVSPTW model to the case that the driving time of the trains is not constant, but a function depending on the total network load. At daytime the freight transport has to wait for the passenger transport such that the traveling speed is much lower than at nighttime.

To this end, the whole day is partitioned into a discrete number of time slices $\mathcal{H} = \{1, \dots, H\}$, that is, $[0, 1439] = \dot{\bigcup}_{h \in \mathcal{H}} [\underline{\tau}_h, \overline{\tau}_h]$. For each train i we introduce decision variables $z_{i,h} \in \{0, 1\}$ with $z_{i,h} = 1$ if and only if the train starts within time slice h. Exactly one slice must be selected:

$$\sum_{h \in \mathcal{H}} z_{i,h} = 1. \tag{14}$$

The slice selection is coupled to the starting time of train i such that the "right" slice h is chosen:

$$\underline{\tau}_h - t_i \leq 1439(1 - z_{i,h}), \tag{15}$$
$$t_i - \overline{\tau}_h \leq 1439(1 - z_{i,h}). \tag{16}$$

Then the trip duration of train i is given by

$$d_i^{\text{trp}} = \sum_{h \in \mathcal{H}} \delta_{i,h}^{\text{trp}} z_{i,h}, \tag{17}$$

where $d_i^{\text{trp}} \in \mathbb{Z}_+$ is a new variable, and $\delta_{i,h}^{\text{trp}}$ is a parameter giving the trip duration of train i when being started in slice h. The actual values of $\delta_{i,h}^{\text{trp}}$ are statistically estimated along historical data. Then in the CVSPTW model, δ_i^{trp} is replaced by d_i^{trp}.

Morover, in case of dynamic trip durations it is desired to specify bounds on the arrival time of some of the trains in addition to the starting time bounds, i.e.,

$$\underline{T}_i \leq t_i + d_i^{\text{trp}} \leq \overline{T}_i, \tag{18}$$

where $[\underline{T}_i, \overline{T}_i] \subseteq [0, 5759] \cap \mathbb{Z}$ is the arrival time interval of train i.

4 Computational Results

The CVSP and the CVSPTW problems are formulated as integer programming problems. Thus one can use standard IP solvers to compute feasible or optimal solutions. For an introduction to integer programming we refer to the literature (Nemhauser and Wolsey [14] for instance). For our computational studies we used

ILOG Cplex 10 [7], one of the currently fastest IP solvers. We made exhaustive tests with the number of parameters that guide the solution process. We made overall best experiences when setting cut generation and probing to the highest level. These settings yield the shortest computation times that are reported in the sequel. The software was running on a Linux server with 16 GByte main memory and 4 dual core AMD Opteron 880 processors running at 2.4 GHz each. Cplex is able to make use out of such environment by parallelizing the branch-and-bound tree.

From the data base of the tool chain we extracted 7 test instances, which we refer to as `A`, `B`, `C`, `D`, `E`, `F`, `G` in the sequel. Here `A` is the smallest instance with 42 trains and 3 classes of locomotives, whereas `G` is the largest, having 1,537 trains and 4 classes of locomotives (for details see the first three columns of Table 1 or Table 2). The instances are related to certain regions within the DB railway network. For example, instance `G` consists of trains mainly from the south of Germany, trains from `A` serve north-south connections, whereas `E` comprises trains from all over Germany.

4.1 Results for the CVSP

At first we take the fixed starting times for the trains in the form they were generated by the train scheduling part of the tool chain. Here each train is assumed to depart as soon as enough wagons are assembled. For most instances the solver was able to find optimal solutions within short amount of time, with exception of `G`, where more than 3 hours of computation time was needed. Note that an instance with around n trains and m classes leads to integer programs with around n^2m many variables, which is more than 3 Mio. for instances `F` and `G`. For the instances from `A` to `G` the solution time grows with the size of the instance (see column 4 of Table 1 or Table 2).

name	trips	classes	time	locomotives	$\sum$	km
A	42	3	1	2/19/2	23	3,808
B	82	3	2	3/34/8	45	52
C	120	4	18	6/35/1/1	43	2,560
D	394	5	27	12/11/21/15/26	85	27,132
E	945	3	480	137/51/45	233	42,057
F	1,507	4	652	28/19/287/52	386	58,446
G	1,537	4	9,502	53/22/26/72	173	2,942

Table 1. Solutions for the uncapacitated CVSP.

The actual amount of locomotives per class that is needed to serve all trips is shown in column 5, the sum of these is given in column 6. Table 1 shows the solutions for the uncapacitated case, that is, it is possible to deploy arbitrarily

name	trips	classes	time	locomotives	$\sum$	km
A	42	3	1	6/15/2	23	958
B	82	3	2	3/26/16	45	52
C	120	4	7	15/26/1/1	43	1,919
D	394	5	27	12/14/22/19/20	87	26,776
E	945	3	2,837	150/38/45	233	35,375
F	1,507	4	4,971	28/19/216/123	386	56,130
G	1,537	4	13,864	53/22/46/52	173	2,838

Table 2. Solutions for the capacitated CVSP.

many locomotives of each class. In contrast, Table 2 shows the solutions of the capacitated case, that is, the number of locomotives per class is limited to the actual stock of DB. In general, the solution times for the capacitated case were higher, in particular for the larger instances `E`, `F`, `G`. However, the total number of locomotives is unchanged (with the exception of instance `D`), only the locomotives per class are different between the capacitated and the uncapacitated case. The last column of Table 1 and Table 2 shows the sum of deadhead trips lengths for all locomotives. Interestingly, the total length (in kilometer) of deadhead trips decrease when switching from the uncapacitated to the capacitated version of the CVSP. This could be ascribed to an increase in diesel locomotives for example, whose deadhead trips are in general shorter.

4.2 Results for the CVSPTW

In the CVSP instances the starting times of all trains were fixed to the times that were computed by the tool chain's scheduler. We now allow the starting times to be altered within a small time window centered around the pre-scheduled starting time, that is, we consider the (uncapacitated) vehicle scheduling problem with time windows (CVSPTW). It turns out that the computation time heavily depends on the actual size of the time windows. Generally speaking, the larger the time window, the more time the solver needs. For all computations we now impose a time limit of 3,600 seconds. We consider the three smallest instances `A`, `B`, and `C`, and take intervals of $\pm 10, \pm 30, \pm 60$, and ± 120 minutes around the current (pre-scheduled) starting time. Moreover, the solver Cplex allows to specify so-called starting solutions, which are integral feasible solutions to the problems that can be generated by other methods (primal heuristics, for instance). For our computations, we take the respective optimal solutions to the (uncapacitated) CVSP (presented in Table 1) as input for the ± 10 instances. Then, we take the optimal (or best feasible) solutions of ± 10 as input for ± 30, and so on. The computational results are shown in Table 3. For the other instances `D` to `G` the solver did not find feasible solutions, or did not even solve the root LP relaxation within the time limit. In column 3 of Table 3 we show the computation time in seconds. An asterisk (*) marks whether the given limit was reached. In those

cases the integrality gap (i.e., the distance between the best known solution and the corresponding lower bound) shown in column 4 is non-zero. The last three columns of Table 3 depict the quality of the optimal or best feasible solution. It turns out that there is a significant potential to save locomotives by changing the departure times of the trains. The possible savings are higher the wider the time windows are open. The price one has to pay for this additional flexibility is the increasing solution time.

instance	time w.	time	gap	locomotives	$\sum$	km
A-42-3	± 10	4	0.00 %	2/18/2	22	3,931
A-42-3	± 30	30	0.00 %	2/14/2	18	3,731
A-42-3	± 60	3,600*	11.69 %	2/11/2	15	2,516
A-42-3	±120	3,600*	27.78 %	2/10/2	14	1,640
B-82-3	± 10	79	0.00 %	3/27/14	44	52
B-82-3	± 30	183	0.00 %	3/23/16	42	52
B-82-3	± 60	3,600*	5.32 %	3/15/19	37	6
B-82-3	±120	3,600*	9.59 %	2/14/16	32	1,591
C-120-4	± 10	753	0.00 %	5/35/1/1	42	2,301
C-120-4	± 30	3,600*	21.84 %	6/32/1/1	40	3,608
C-120-4	± 60	3,600*	38.14 %	6/30/1/1	38	3,120
C-120-4	±120	3,600*	79.06 %	6/29/1/1	37	2,882

Table 3. CVSPTW with different time window sizes.

Note that in these computations the number of missed transfers of wagons is not shown. This is due to the fact that the corresponding variables $p_{i,j}$ are fixed to their lower bounds beforehand. This reflects the fact that a fast transfer of wagons is always more important than saved locomotives.

4.3 Results for the refined CVSPTW

Finally we consider the (uncapacitated) CVSPTW with netload-dependend travel times for the trains. This is the most evolved model in our hierarchy and it comes with no surprise that the solution times here are even higher than for the CVSPTW with constant traveling times. As before, we take the best feasible solution of an instance with a smaller time window as integer starting solution for the instance with the next bigger time window. Our results are summarized in Table 4. Depending on the starting time, the total travel time varies between 70 % and 130 % of the constant travel time. The bias of the travel time is estimated using historical data.

The results in Table 4 give an impression on the current state of problem sizes that can be solved using Cplex out of the box, with some altered parameter settings. Since we are far from solving even medium-size instances to optimality,

instance	time w.	time	gap	locomotives	$\sum$	km
A-42-3	± 10	5	0.00 %	2/18/2	22	3,808
A-42-3	± 30	34	0.00 %	2/13/2	17	2,510
A-42-3	± 60	3,600*	19.38 %	2/11/2	15	2,516
A-42-3	±120	3,600*	36.99 %	3/9/2	14	2,125
B-82-3	± 10	102	0.00 %	3/24/17	44	98
B-82-3	± 30	213	0.00 %	3/19/18	40	52
B-82-3	± 60	3,600*	9.41 %	3/20/13	36	6
B-82-3	±120	3,600*	16.56 %	2/18/11	31	1,649
C-120-4	± 10	1,033	0.00 %	5/35/1/1	42	2,301
C-120-4	± 30	3,600*	22.60 %	8/30/1/1	40	3,083
C-120-4	± 60	3,600*	41.10 %	7/29/1/1	38	3,168
C-120-4	±120	3,600*	79.06 %	6/29/1/1	37	4,191

Table 4. CVSPTW with netload-dependent travel times.

one can try to use heuristic reductions of the problem's complexity. A first idea in this respect is to remove those deadhead trips that are above a certain limit. The hope is that sufficiently many long deadhead trips are removed by this, such that the remaining instance is smaller and computationally easier to solve, and on the other hand the solution is not too far away from the optimal solution. Similar as above, the best feasible solution of one instance with a tighter bound on the maximal deadhead length can be used as input for another instance with a larger bound. This we use here for the `C` instances, which cannot be solved to optimality within the given time limit. Our results are shown in Table 5. Column 3 of this table contains the upper bound on the deadhead trip length $\delta^{\text{dhd}}_{b,(i,j)}$. If $\delta^{\text{dhd}}_{b,(i,j)}$ exceeds the respective bound then $a_{b,(i,j)}$ is set to zero. The last three columns of Table 5 show the deviation from the optimality (for those instances `A` and `B` where the optimal solution is known, cf. Table 4). For convenience we repeat in the first row of each block `A`, `B`, `C` the results from Table 5 (with an upper bound of ∞, which is equivalent to no upper bound). As one can see, the loss is only a small one, whereas the solution time (given in column 4) is much lower now.

5 Conclusions and Further Work

In this article we presented new models for the strategic locomotive scheduling. These models were in part inspired by similar optimization problems in public bus transport. However, several new rail-specific requirements emerged such that the models could not be directly carried over. The models were formulated as integer programs. Thus commercial standard software for their solution could be applied. The evaluation of the capability of this software was part of our project. It turned out that for the cyclic vehicle scheduling without time

instance	time w.	bound	time	gap	locomotives	$\sum$	km
A-42-3	±30	∞	34	0.00 %	2/13/2	17	2,510
A-42-3	±30	600	27	0.00 %	3/12/2	17	958
A-42-3	±30	300	9	0.00 %	3/12/2	17	958
A-42-3	±30	100	2	0.00 %	3/16/2	21	6
A-42-3	±30	0	2	0.00 %	3/16/3	22	0
B-82-3	±30	∞	213	0.00 %	3/19/18	40	52
B-82-3	±30	100	39	0.00 %	3/25/12	40	52
B-82-3	±30	50	39	0.00 %	3/23/14	40	52
B-82-3	±30	10	24	0.00 %	3/24/14	41	6
B-82-3	±30	0	22	0.00 %	4/23/15	41	0
C-120-4	±30	∞	$3,600^*$	21.08 %	4/34/1/1	40	2,459
C-120-4	±30	600	$3,600^*$	18.63 %	4/34/1/1	40	2,459
C-120-4	±30	300	$3,600^*$	18.54 %	5/33/1/1	40	2,113
C-120-4	±30	100	$3,600^*$	13.85 %	6/37/1/1	45	486
C-120-4	±30	0	$3,600^*$	12.28 %	7/44/2/1	54	0

Table 5. CVSPTW with upper bounds on the deadhead trips.

windows, the software was able to solve even larger instances. As soon as time windows enter the scene, the sizes where global optimal solutions were computed was reduced by some orders of magnitude. Our further work thus aims at an improvement of the model formulation, where we want to develop and implement primal heuristics, problem-specific cutting planes, and branching rules such that larger models can be routinely solved to optimality. Also a further refinement of the presented model is on our agenda. An example in this direction is the inclusion of deadheading locomotives which are included in trains and thus do not need own power and staff for operating.

Acknowledgements. We thank Dr. Gerald Pfau (Deutsche Bahn AG), Jörg Wolfner (Universität Dortmund, Lehrstuhl Verkehrssysteme und -logistik), and Andreas Ginkel (Universität Göttingen) for fruitful and productive discussions. Last but not least we would like to thank the anonymous reviewers for their various helpful comments.

References

1. Deutsche Bahn: Geschäftsbericht des Konzerns. Deutsche Bahn AG, Potsdamer Platz 2, 10785 Berlin (2005)
2. Railion Deutschland: Geschäftsbericht. Railion Deutschland AG, Rheinstraße 2, 55116 Mainz (2005)
3. Löbel, A.: Optimal Vehicle Scheduling in Public Transit. Shaker Verlag, Aachen (1997)
4. Gintner, V., Kliewer, N., Suhl, L.: Solving large multi-depot multi-vehicle-type bus scheduling problems in practice. OR Spectrum **27** (2005) 507 – 523
5. Daduna, J., Völker, M.: Fahrzeugumlaufbildung im ÖPNV mit unscharfen Abfahrtszeiten. Der Nahverkehr **11** (1997) 39 – 43
6. Fügenschuh, A.: The Integrated Optimization of School Starting Times and Public Transport. Logos Verlag, Berlin (2005)
7. ILOG Ltd.: ILOG Cplex 10 Solver Suite. Technical report, ILOG Cplex Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA (2006)
8. Bussieck, M., Winter, T., Zimmermann, U.: Discrete optimization in public rail transport. Mathematical Programming **79** (1997) 415 – 444
9. Caprara, A., Fischetti, M., Toth, P., Vigo, D.: Algorithms for railway crew planning. Mathematical Programming **79** (1997) 125 – 141
10. Ahuja, R., Liu, J., Orlin, J., Sharma, D., Shughart, L.: Solving real-life locomotive scheduling problems. Transportation Science **39** (2002) 503 – 517
11. Ziarati, K.: A heuristic to find cyclical planning solution for locomotive assignment problems. In: Proc. 1st Nat. Ind. Eng. Conf., Iran, Sharif University of Technology (2001)
12. Liebchen, C., Möhring, R.: The modeling power of the periodic event scheduling problem: Railway timetables – and beyond. Technical Report 2004/20, Technische Universität Berlin (2004)
13. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, New Jersey (1993)
14. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. Wiley Interscience, New York (1999)

Periodic Metro Scheduling*

Evangelos Bampas, Georgia Kaouri, Michael Lampis, Aris Pagourtzis

School of Elec. and Comp. Engineering, National Technical University of Athens
15780, Athens, Polytechnioupoli, Greece
{ebamp,gkaouri,mlampis,pagour}@cs.ntua.gr

Abstract. We introduce the Periodic Metro Scheduling (PMS) problem, which aims in generating a periodic timetable for a given set of routes and a given time period, in such a way that the minimum time distance between any two successive trains that pass from the same point of the network is maximized. This can be particularly useful in cases where trains use the same rail segment quite often, as happens in metropolitan rail networks.
We present exact algorithms for PMS in chain and spider networks, and constant ratio approximation algorithms for ring networks and for a special class of tree networks. Some of our algorithms are based on a reduction to the Path Coloring problem, while others rely on techniques specially designed for the new problem.

Keywords. train scheduling, path coloring, delay-tolerant scheduling, periodic timetabling

1 Introduction

In railway networks where trains use the same railway segment quite often (e.g. metro) it would be desirable to schedule trains so as to guarantee an ample time distance between successive trains that pass from the same point of the network (in the same direction). Such a scheduling would result in a more delay-tolerant system. This is a particularly essential requirement in cases where there are several overlapping routes that have to be carried out periodically and the time limits are such that some route must start before the termination of another route with which it shares a part of the network.

Here, we formulate this situation by introducing the problem Periodic Metro Scheduling (PMS): given a rail network, a set of routes (described as paths over the network graph), and a time period, we seek to arrange the departure times of routes so that the minimum time distance between any two trains that pass from the same point of the network is maximized. Although

* Research supported in part (a) by the European Social Fund (75%) and the Greek Ministry of Education (25%) through "Pythagoras" grant of the Operational Programme on Education and Initial Vocational Training and (b) by the National Technical University of Athens through "Protagoras" grant.

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/684

our motivation comes from railway optimization, PMS may also describe other transportation media timetabling problems.

We show the *NP*-hardness of PMS, by reduction from Path Coloring (PC), which is the problem of coloring paths in a graph with minimum number of colors so that intersecting paths receive different colors. We further investigate the relation between the two problems and present exact algorithms for chain and spider networks that rely on a reduction from PMS to PC. Moreover, we show that this technique also applies to rings for which the time needed to traverse the ring is a multiple of the given period. This results to a $\left(\frac{1}{\rho}\frac{L}{L+1}\right)$-approximation algorithm for such instances, where ρ is the approximation ratio we can achieve for PC and L is the maximum number of routes passing through any edge of the network. For ring instances that do not satisfy this condition we present a more involved algorithm that achieves an approximation guarantee of $\frac{1}{6}$. Finally, we show how to apply the path coloring technique to tree networks where the time distance between stations is a multiple of the half of the period, resulting in a $\left(\frac{1}{\rho}\frac{L}{L+1}\right)$-approximation algorithm for this topology as well. Our algorithms employ known algorithms for PC [4, 22, 9, 13] as subroutines.

Related work. To the best of our knowledge Periodic Metro Scheduling has not been studied before in the form of an optimization problem. The decision version of PMS, namely the problem of requiring a minimum safety distance not smaller than a given threshold between all pairs of overlapping routes, can be described in terms of a generic problem known as Periodic Event Scheduling Problem (PESP) [23]. PESP has been studied by several researchers, see e.g. [25, 19, 18, 20] and references therein. However, we are not aware of any concrete results for PESP that could apply to PMS, as PESP is usually studied in conjuction with several other constraints that render the problem quite hard and the proposed methods for solving it are mainly heuristics based on "branch-and-bound", "branch-and-cut", and "branch-and-price" methods. A similar problem as PMS has been defined in [11], and it has been proven to be *NP*-complete. However, the setting is broader and the completeness results apply to general graphs[1].

There is a huge bibliography on railway optimization topics; the interested reader is referred to [3] for a nice collection of concepts and earlier results on railway optimization. More recent work on periodic train scheduling includes a rolling stock minimization problem where routes are given and it is sought to determine departure times either arbitrarily (as in our case) or within an allowed time window [7]; however, the objective there is quite different, namely to serve all routes with a minimum number of trains while it is allowed for routes to simultaneously depart from the same station even if they follow the same direction. The rolling stock minimization problem with fixed departure times has been extensively studied: the simplest version, also known as Minimum Fleet Size [2], or Rolling Stock Rostering [8], can be solved exactly in

[1] Unfortunately, we were not able to thoroughly check the similarity of those results to ours because that thesis is available only in German.

polynomial time; Dantzig and Fulkerson [6] give the first known algorithm and Erlebach et al. [8] present one of improved complexity. In [8] some variations are also studied and shown *APX*-hard: allowing empty rides and requiring that the trains pass through a *maintenance station*; constant ratio approximation algorithms have been proposed.

A problem that has recently drawn attention is that of delay management, that is, how to reduce or increase delays of trains in order to better serve railway customers [24, 12, 14]. Several other railway optimization problems have been considered in the literature [26, 5, 21, 1], most of which are hard to solve; a number of heuristics have been presented for them in the corresponding papers.

2 Definitions – Preliminaries

We assume that all trains move at the same speed, therefore the duration of traveling between any two connected stations is the same for all trains. In the sequel we denote the travel time between two stations connected by edge e as $t(e)$. We also assume that all edges represent directed railway lines and any two connected stations are linked by a pair of opposite directed edges. For simplicity we consider that the waiting time at stations is negligible.

We are interested in maximizing the time distance between any two overlapping routes, that is, routes that share at least one edge. Note that, due to the uniformity of speed, it suffices to measure the time distance between overlapping routes only at the starting node (station) of the first edge of each common section. More precisely, let e be a common edge between routes r and r' and t (resp. t') be the time at which r (resp. r') enters edge e. Then the time distance of r and r' at edge e is defined as $\min(t - t' \bmod T, t' - t \bmod T)$. When the time distance between two routes in an edge is 0 we say that the routes collide.

We will denote the source of a route r by $s(r)$, and its target by $e(r)$. We define $\tau(i, j)$ as the time distance between nodes i and j in the input graph.

Let us now formally define our problem.

Periodic Metro Scheduling (PMS)
Input: A directed graph $G = (V, E)$, an inter-station time function $t : E \to \mathbb{N}$, an integer time period T and a collection $R = \{r_1, \ldots, r_k\}$ of simple paths on G (routes).
Feasible solution: A schedule for R, that is, a function $stime : R \to [0, T)$ which assigns a departure time to each route such that no two routes enter the same edge at the same time.
Goal: Maximize the minimum time distance between any two overlapping routes.

We define $L(e)$ to be the number of paths that pass through an edge e of the graph. Let $L = \max_e L(e)$. It is not hard to see that $\frac{T}{L}$ is an upper bound to the objective value of an optimal solution (OPT), because routes cannot be spaced further apart than $\frac{T}{L}$ on the edge with the maximum load.

In our study we will show a close relation between PMS and PC. The definition of PC is as follows:

Path Coloring (PC)
Input: A directed graph $G = (V, E)$ and a collection $\mathcal{P} = \{r_1, \ldots, r_k\}$ of paths on G.
Feasible solution: An assignment of colors to all paths of $\mathcal{P}$ such that no two paths which share an edge are assigned the same color.
Goal: Minimize the number of colors used.

PC (note that here we consider the directed version) can be solved optimally in polynomial time in chains (folklore, see e.g. [15]), stars and spiders using L colors, but is known to be *NP*-hard in rings [13] and trees [22].

We will widely use the notation $a \equiv_T b$ to denote that $a \bmod T = b \bmod T$.

3 PMS in chain, star and spider networks

A chain is a graph that consists of a single path. A star is a tree with at most one internal node. A spider is a tree in which at most one internal node has degree ≥ 3, called the central node; that is, a spider is a graph resulting from a star whose edges have been replaced by chains (also called legs of the spider).

In chains we label the nodes of the graph from 0 to $n-1$ successively. In stars we label the central node 0 and the peripheral nodes $1, \ldots, n$. In a spider with k legs each node is labeled (i, j) where $i = 1, \ldots, k$ represents the leg the node belongs to and j represents the node's position in the leg relative to the central node.

3.1 An algorithm for chains

Since all connections are bidirectional we can divide any problem instance into two subproblems, one containing paths moving to the right and one containing paths moving to the left and solve them separately.

Let t_i be the time distance from node i to $i+1$ for $i = 1, \ldots, n-1$. In the case of chain networks the time distance between two nodes i and j is

$$\tau(i, j) = \sum_{k=i}^{j-1} t_k$$

We will make use of the fact that PC can be solved optimally for chain networks by using a simple greedy technique.

We propose the following algorithm:

Algorithm 1 An algorithm for PMS in chain networks

1: Compute a path coloring of routes with exactly L colors from $\{0, \ldots, L-1\}$. Let $color(r)$ denote the color assigned to route r.
2: Set $t = \frac{T}{L}$ and define L time slots as follows: $0, t, 2t, \ldots, (L-1)t$.
3: Assign time slots to routes according to the coloring obtained in step 1, namely $timeslot(r) := color(r) \cdot t$.
4: For each r set starting time

$$stime(r) = (timeslot(r) + \tau(0, s(r))) \bmod T$$

Theorem 1. *Algorithm 1 computes an optimal solution for* PMS *in chains.*

Proof. Let r and r' be two overlapping paths and without loss of generality assume that $s(r') \leq s(r)$. The first point of their common section is $s(r)$ and their time distance at $s(r)$ is:

$$\begin{aligned} d(r, r', s(r)) = \min\ & ((stime(r') + \tau(s(r'), s(r)) - stime(r)) \bmod T, \\ & (stime(r) - (stime(r') + \tau(s(r'), s(r)))) \bmod T) \end{aligned}$$

Note that

$$\begin{aligned} stime(r') + \tau(s(r'), s(r)) - stime(r) &= timeslot(r') + \tau(0, s(r')) + \tau(s(r'), s(r)) \\ &\quad -(timeslot(r) + \tau(0, s(r))) \\ &= timeslot(r') - timeslot(r) \end{aligned}$$

Therefore

$$\begin{aligned} d(r, r', s(r)) &= \min\left((timeslot(r') - timeslot(r)) \bmod T, (timeslot(r) - timeslot(r')) \bmod T\right) \\ &\geq \frac{T}{L} \end{aligned}$$

Hence, the solution returned by the algorithm is optimal, since $\frac{T}{L}$ is an upper bound for the value of any feasible solution.

□

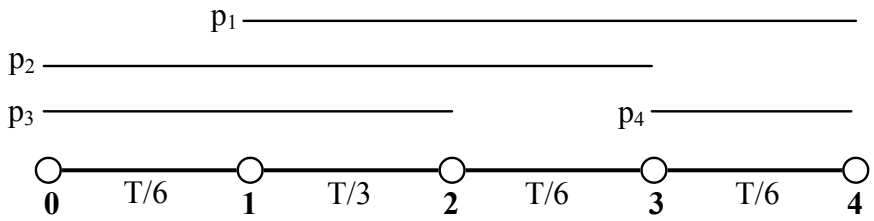


Fig. 1. An instance of PMS in chain networks.

Example 1. Consider the instance of Figure 1. The maximum load is $L = 3$ and as a result the path coloring algorithm will yield a solution with 3 colors. The time slots corresponding to these colors are: 0, $\frac{T}{3}$ and $\frac{2T}{3}$. Assume that paths p_3 and p_4 are assigned time slot 0, p_2 is assigned time slot $\frac{T}{3}$ and p_1 is assigned time slot $\frac{2T}{3}$. According to Algorithm 1, $stime(p_1) = \frac{5T}{6}$, $stime(p_2) = \frac{T}{3}$, $stime(p_3) = 0$ and $stime(p_4) = \frac{2T}{3}$. Observe that on edge $(1, 2)$ the three overlapping paths p_1, p_2, p_3 have time distance at least $\frac{T}{3}$, which is optimal. Furthermore, path p_1 reaches node 3 at time $\frac{T}{3}$ ("wrapping around" the end of the time period), thus also having distance $\frac{T}{3}$ from p_4.

3.2 An algorithm for stars and spiders

Given an instance of PMS in a star or a spider, we will utilize an optimal path coloring of the given instance in order to produce an optimal time schedule; note that such an exact algorithm for spiders can be obtained by appropriate combination of an exact algorithm for stars and the algorithm for chains. We should note that some routes may be confined in one of the spider's legs while others may be directed from one leg to another.

Algorithm 2 An algorithm for PMS in spider networks

1: Compute a path coloring of routes with exactly L colors from $\{0, \ldots, L-1\}$. Let $color(r)$ denote the color assigned to route r.
2: Set $t = \frac{T}{L}$ and define L time slots as follows: $0, t, 2t, \ldots, (L-1)t$.
3: Assign time slots to routes according to the coloring obtained in step 1, namely $timeslot(r) := color(r) \cdot t$.
4: For each r passing through the central node, set starting time

$$stime(r) = (timeslot(r) - \tau(0, s(r))) \bmod T$$

5: For each r confined in a single leg and directed towards the central node, set starting time

$$stime(r) = (timeslot(r) - \tau(0, s(r))) \bmod T$$

6: For each r confined in a single leg and directed away from the central node, set starting time

$$stime(r) = (timeslot(r) + \tau(0, s(r))) \bmod T$$

Theorem 2. *Algorithm 2 computes an optimal solution for* PMS *in spiders.*

Proof. We will first prove the claim for the case where the spider is a star. Let r and r' be two overlapping routes. Therefore they receive different colors, hence also different time slots. There are two cases: either $s(r) = s(r')$ or $e(r) = e(r')$. In both cases, it suffices to examine their time distance at the central node. Each route arrives at or departs from the central node at time equal to its time slot.

Therefore their time distance is a nonzero multiple of $t = \frac{T}{L}$, which is an upper bound for OPT.

In a general spider network, we consider two cases. For two overlapping routes that pass through the central node, we can use the same argumentation as above for star networks. For two overlapping routes that lie in the same leg, the proof is similar to the proof of Theorem 1 for chains since it can be shown that the same properties hold considering either the central node or the tip of a leg as the first node of the chain (possibly with an appropriate time shift).

□

4 PMS in ring networks

In the case of ring networks, that is, networks which consist of a single cycle, we can assume that all trains travel in the same direction (clockwise, without loss of generality), for the same reasons as for chains. Nodes are labeled by picking one arbitrarily and labeling it 0, then labeling every other node $1, \ldots, n-1$ starting from 0's neighbor in the direction trains travel. We define $\tau(i, j)$ as the time distance from node i to node j in the clockwise direction. We also define the ring perimeter C as the total time needed to travel around the ring.

For ring networks we can distinguish between two cases: the case where the ring perimeter C is a multiple of the period T and the case where it is not. In the following two sections we will analyze these cases.

4.1 The case $C \equiv_T 0$

Theorem 3. *An instance of* PMS *in a ring with* $C \equiv_T 0$ *admits a solution of value at least* $\frac{T}{k}$ *if and only if the corresponding* PC *instance can be colored with* k *colors.*

Proof. For the "if" direction, we can produce the desired schedule by using Algorithm 1 for PMS in chains, starting from step 2 and using k instead of L. Let r and r' be two overlapping paths; without loss of generality assume that $s(r')$ is closer to 0 than $s(r)$ in the clockwise direction. Because $C \equiv_T 0$, it can be shown that it suffices to check their time distance on only one of their common segments, even if there are two such segments.

Following similar arguments as those in the proof of Theorem 1, it can be shown that the time distance is:

$$\min((\mathit{timeslot}(r) - \mathit{timeslot}(r')) \bmod T, (\mathit{timeslot}(r') - \mathit{timeslot}(r)) \bmod T) \geq \frac{T}{k}$$

For the "only if" direction, suppose we have a schedule for the PMS instance with value $\frac{T}{k}$. We will show how to obtain a coloring with k colors for the corresponding PC instance. For each route r, let $\mathit{timeslot}(r) = (\mathit{stime}(r) - \tau(0, s(r))) \bmod T$. Assign to r the color $w - 1$, where w is the smallest integer such that $\mathit{timeslot}(r) < w \cdot \frac{T}{k}$. Since w ranges from 1 to k and for any two overlapping paths r and r' we have $|\mathit{timeslot}(r) - \mathit{timeslot}(r')| \geq \frac{T}{k}$, this is a valid coloring.

□

Corollary 1. PMS *in rings is NP-hard.*

Proof. We present a reduction from the decision version of PC in rings to the decision version of PMS in rings. PC is known to be *NP*-hard in rings [13]. Suppose we are given an instance of PC in a ring with n nodes and a path set $\mathcal{P}$, asking if $\mathcal{P}$ is colorable with k colors. We construct an instance of PMS in a ring with n nodes, routes identical to the paths in $\mathcal{P}$, inter-station distances of one time unit and $T = n$, asking if it is possible to achieve an objective function value of $\frac{T}{k}$. Clearly, the corresponding PC instance for the PMS instance we produced is the original PC instance. Therefore Theorem 3 applies, implying that the original PC instance can be colored with k colors if and only if a solution of value $\frac{T}{k}$ can be achieved for the PMS instance.

□

At a first glance Theorem 3 seems to imply that a ρ-approximation algorithm for PC would give a $\frac{1}{\rho}$-approximation algorithm for PMS. However, this is true only in the case that the optimal solution for the PMS instance divides exactly T. In the general case we can show something slightly weaker.

Theorem 4. *A ρ-approximation algorithm for* PC *in rings implies an $\left(\frac{1}{\rho}\frac{L}{L+1}\right)$-approximation algorithm for* PMS *in rings with $C \equiv_T 0$.*

Proof. We will use the algorithm of Theorem 3. Let $\mathrm{OPT}_{\mathrm{PMS}}$ be the value of an optimal solution of an instance of PMS and $\mathrm{OPT}_{\mathrm{PC}}$ the cost of an optimal solution of the corresponding PC instance. We observe that $\mathrm{OPT}_{\mathrm{PMS}} < \frac{T}{\mathrm{OPT}_{\mathrm{PC}}-1}$ because a solution of PMS of value $\frac{T}{\mathrm{OPT}_{\mathrm{PC}}-1}$ would lead to a coloring with only $\mathrm{OPT}_{\mathrm{PC}} - 1$ colors by Theorem 3. Recall also that $\mathrm{OPT}_{\mathrm{PMS}} \leq \frac{T}{L}$.

A ρ-approximation algorithm for PC returns a solution $\mathrm{SOL}_{\mathrm{PC}} \leq \rho \cdot \mathrm{OPT}_{\mathrm{PC}}$. By Theorem 3 we can compute a solution for PMS of value $\mathrm{SOL}_{\mathrm{PMS}} = \frac{T}{\mathrm{SOL}_{\mathrm{PC}}} \geq \frac{1}{\rho} \cdot \frac{T}{\mathrm{OPT}_{\mathrm{PC}}}$. By the observations above it turns out that:

$$\mathrm{SOL}_{\mathrm{PMS}} \geq \frac{1}{\rho} \cdot \frac{T}{\frac{T}{\mathrm{OPT}_{\mathrm{PMS}}} + 1} = \frac{1}{\rho} \cdot \frac{T \cdot \mathrm{OPT}_{\mathrm{PMS}}}{T + \mathrm{OPT}_{\mathrm{PMS}}} \geq \frac{1}{\rho} \cdot \frac{L}{L+1} \cdot \mathrm{OPT}_{\mathrm{PMS}}$$

□

Corollary 2. *There is a $\left(\frac{2}{3} \cdot \frac{L}{L+1}\right)$-approximation algorithm and a $\left(0.73 \cdot \frac{L}{L+1}\right)$-approximation randomized algorithm for* PMS *in rings with $C \equiv_T 0$.*

Proof. By using Theorem 4 and the deterministic approximation algorithm of Karapetian [16] and the randomized approximation algorithm of Kumar [17] that achieve ratios $\frac{3}{2}$ and 1.368 respectively.

□

Remark 1. So far, we have assumed that the departure times of trains could be any rational number in $[0, T)$. However there is a possibility that trains need to be assigned integer departure times. In this case following by appropriate modification of the above analysis, it can be shown that our results carry over with a further reduction of the approximation ratio to:

$$\frac{1}{\rho} \cdot \frac{L}{L+1} - \frac{1}{\mathrm{OPT}_{\mathrm{PMS}}}$$

It should be noted though, that this approximation ratio is asymptotically equal to the approximation ratio obtained for rational departure times.

4.2 The case $C \not\equiv_T 0$

Consider a ring network with n nodes and two paths p_1, p_2 with $0 = s(p_1) < e(p_2) < s(p_2) < e(p_1)$ and $\tau(s(p_1), s(p_2)) = x$. Let t_1, t_2 be the moments in time where the trains traveling along p_1 and p_2 arrive at node 0. These trains reach node $s(p_2)$ at times $(t_1 + x) \bmod T$ and $(t_2 - D + x) \bmod T$ respectively, where $D = C \bmod T$. As a result in order to maximize the minimum distance of the two trains, we have to take into account the following time differences: $(t_1 - t_2) \bmod T$, $(t_2 - t_1) \bmod T$, $(t_1 - t_2 + D) \bmod T$ and $(t_2 - t_1 - D) \bmod T$. It is now clear that the algorithm of Theorem 3 may produce an infeasible solution if $D = (t_2 - t_1) \bmod T$ (see Figure 2). Therefore we need a new algorithm for this case.

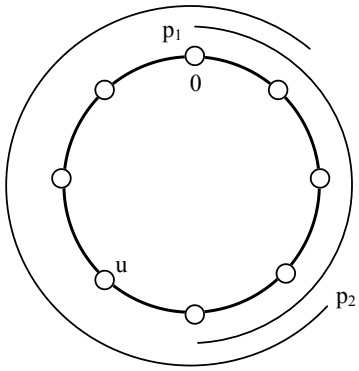


Fig. 2. An example showing that the "path coloring" technique does not work for rings with $C \not\equiv_T 0$. Assuming $\tau(0, u) = T$ and $\tau(u, 0) = \frac{T}{2}$, the path coloring technique would assign time slots 0 and $\frac{T}{2}$ to paths p_1 and p_2 respectively and the two paths would collide at node 0 at any time which is an integer multiple of T.

Theorem 5. *Let S be a set of paths passing through node 0 in a ring network. There is an enumeration of S such that for any two paths p and p', if $e(p) > s(p')$ then p appears before p' in the enumeration.*

Proof. We define the binary relation $\prec$ over S: $p \prec p'$ if and only if $e(p) > s(p')$. This relation is antisymmetric: suppose $p \prec p'$ and $p' \prec p$. Then $e(p') > s(p) > e(p) > s(p')$ which is a contradiction, because for each path $r \in S$ it holds that $s(r) > e(r)$. It is also transitive: if $p \prec r$ and $r \prec q$ then we have $e(p) > s(r) > e(r) > s(q)$, thus $p \prec q$. As a result, the relation $\prec$ is a strict partial order. The theorem follows.

□

Algorithm 3 An algorithm for PMS in ring networks with $C \not\equiv_T 0$

1: Split R into two sets $\mathcal{P}_0$ and $\mathcal{P}_c$. $\mathcal{P}_0$ contains the paths passing through node 0 (i.e. having node 0 as an intermediate node) and $\mathcal{P}_c = R \setminus \mathcal{P}_0$. Let $L_0 = |\mathcal{P}_0|$ and L_c be the maximum load with respect to $\mathcal{P}_c$.
2: Define $t = \frac{T}{6L'}$ and the set of available time slots as follows: $S = \{0, t, 2t, \ldots, (6L' - 1)t\}$, where $L' = \max\{L_0, L_c\}$.
3: Assign colors to routes of $\mathcal{P}_c$ by using an algorithm for PC in chains.
4: **for** each color k, $1 \leq k \leq L_c$ **do**
5: define $timeslot(k) = kt$
6: **for** each path p colored with k **do**
7: assign departure time $stime(p) = timeslot(k) + \tau(0, s(p))$.
8: **end for**
9: Remove kt from S
10: Remove from S all time slots whose distance from $kt + D$ is smaller than t.
11: **end for**
12: Enumerate paths in $\mathcal{P}_0$ as implied by Theorem 5.
13: **for** each $p \in \mathcal{P}_0$ in the order of the enumeration **do**
14: Set $timeslot(p) = wt$, where wt is the smallest available time slot.
15: Set $stime(p) = (wt - \tau(s(p), 0)) \bmod T$.
16: Remove wt from S
17: Remove from S all time slots whose distance from $wt + D$ is smaller than t.
18: **end for**

Theorem 6. *Algorithm 3 is a $\frac{1}{6}$-approximation algorithm for* PMS *in rings.*

Proof. First let us observe that $6L'$ time slots suffice to arrange the departure times of all routes. As far as paths in $\mathcal{P}_c$ are concerned, each one uses one time slot and excludes at most two others, in total using at most $3L_c$ time slots. Similarly, paths in $\mathcal{P}_0$ use at most $3L_0$ time slots.

The distance between time slots in Algorithm 3 is $\frac{T}{6L'}$. We will show that the time distance between any two overlapping routes at any point is not smaller than the time distance between the corresponding time slots. Since the algorithm

assigns different time slots to overlapping routes, the minimum time distance is at least $\frac{T}{6L'}$.

Now, observe that no two paths in $\mathcal{P}_c$ can have a time difference smaller than $\frac{T}{6L'}$ in a scheduling produced by the algorithm, because their arrangement is essentially the same as in the case of a chain network.

Suppose we have two overlapping paths $r \in \mathcal{P}_c$ and $r' \in \mathcal{P}_0$. We need to show that their time difference is not less than $\frac{T}{6L'}$ at nodes $s(r)$ and $s(r')$ if these nodes are shared between the two paths, since all trains travel at the same speed.

Let $t_0 = \mathit{timeslot}(r')$ be the time when r' passes through node 0. r' reaches $s(r)$ (if $s(r)$ is contained in r') at time $(\tau(0, s(r))+t_0) \bmod T$, and since $\mathit{stime}(r) = (\tau(0, s(r)) + \mathit{timeslot}(r)) \bmod T$ if r and r' had a time distance of less than $\frac{T}{6L'}$ then they would have been assigned the same time slot, which is a contradiction.

Route r arrives at $s(r')$ at time $\mathit{stime}(r) + \tau(s(r), s(r')) = \mathit{timeslot}(r) + \tau(0, s(r'))$ while r' departs from $s(r')$ at time $\mathit{stime}(r') = \mathit{timeslot}(r') - \tau(s(r'), 0) \equiv_T \mathit{timeslot}(r') - D + \tau(0, s(r'))$. If r and r' had time distance of less than $\frac{T}{6L'}$, then $\mathit{timeslot}(r')$ would have a distance of less than $\frac{T}{6L'}$ from $\mathit{timeslot}(r) + D$ which is also a contradiction.

Finally, let us consider two paths r, r' in $\mathcal{P}_0$ to which the algorithm has assigned time slots $\mathit{timeslot}(r)$ and $\mathit{timeslot}(r')$ respectively. Suppose, without loss of generality, that $s(r') > s(r)$. It is clear that since they are assigned different time slots these two paths cannot have a time distance of less than $\frac{T}{6L'}$ at node 0 and therefore neither at node $s(r')$. We now need to show that their time distance is not less than $\frac{T}{6L'}$ at node $s(r)$. We should examine two cases depending on whether $r \prec r'$ or not.

Suppose $e(r') > s(r)$. In that case $r' \prec r$ and r' will be assigned a time slot before r. Route r' will reach $s(r)$ at time $\mathit{stime}(r') + \tau(s(r'), 0) + \tau(0, s(r)) \equiv_T \mathit{timeslot}(r') + \tau(0, s(r))$. Route r departs from $s(r)$ at time $\mathit{stime}(r) = \mathit{timeslot}(r) - \tau(s(r), 0) \equiv_T \mathit{timeslot}(r) - D + \tau(0, s(r))$. However, the time distance between $\mathit{timeslot}(r') + D$ and $\mathit{timeslot}(r)$ is at least $\frac{T}{6L'}$, because r' was assigned a time slot before r and $\mathit{timeslot}(r') + D$ was excluded from the list of available time slots.

Let us now assume that $e(r') < s(r)$. In that case the the two paths have only one common segment that starts at $s(r')$ and contains 0. Therefore, the fact that they have been assigned different time slots suffices to guarantee that their time difference is at least $\frac{T}{6L'}$.

□

5 PMS in tree networks

In the case of tree networks one might attempt to use Algorithm 2 for spiders, after picking an arbitrary node 0. However, this idea may lead to the production of an infeasible solution. Figure 3 illustrates this situation.

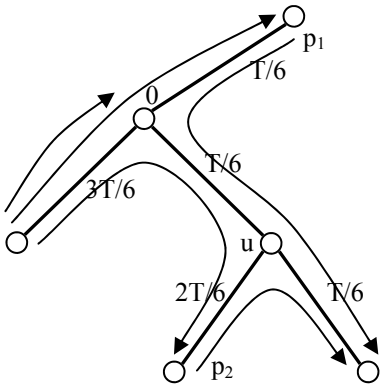


Fig. 3. An example showing that Algorithm 2 for spiders does not work for trees. Assuming that path p_1 is assigned time slot 0 and path p_2 is assigned time slot $\frac{T}{3}$, path p_1 collides with path p_2 at node u at time $\frac{T}{6}$.

However, if we consider tree networks in which the time needed to travel along each edge is a multiple of $\frac{T}{2}$ it turns out that we can use Algorithm 4. In these networks the following useful property holds.

Remark 2. For any three nodes a, b, c: $\tau(a, b) + \tau(b, c) \equiv_T \tau(a, c)$.

Theorem 7. *An instance of* PMS *in a tree where the time needed to travel along each edge is a multiple of $\frac{T}{2}$ admits a solution of value at least $\frac{T}{k}$ if and only if the corresponding* PC *instance can be colored with k colors.*

Proof. For the "if" direction, the following algorithm yields a solution with the desired value.

Algorithm 4 An algorithm for PMS in tree networks where each edge is a multiple of $\frac{T}{2}$

1: Assume that a coloring with k colors is given.
2: Pick a root r_0 arbitrarily.
3: Set $t = \frac{T}{k}$ and the available time slots as follows: $0, t, 2t, \ldots, (k-1)t$.
4: **for** each path p **do**
5: Let $timeslot(p)$ be the time slot corresponding to the color of p.
6: Set the starting time of p as follows:

$$stime(p) = (\tau(r_0, s(p)) + timeslot(p)) \bmod T$$

7: **end for**

Assume there are two paths p and p' overlapping on a single edge $e = (u, v)$. Path p reaches node u at time $(\mathit{timeslot}(p) + \tau(r, s(p)) + \tau(s(p), u)) \bmod T = (\mathit{timeslot}(p) + \tau(r, u)) \bmod T$. By the same reasoning path p' reaches node u at time $(\mathit{timeslot}(p') + \tau(r, u)) \bmod T$. Hence, the time distance of the two paths is equal to $(\mathit{timeslot}(p') - \mathit{timeslot}(p)) \bmod T$ which is clearly at least $\frac{T}{k}$.

For the "only if" direction, we pick an arbitrary node r_0 and, for each path p, we consider the value $\mathit{timeslot}(p) = \mathit{stime}(p) - \tau(r_0, s(p))$. Following the proof of Theorem 3 and using remark 2, we obtain a valid coloring with k colors for the original PC instance.

□

Corollary 3. PMS *in trees is NP-hard.*

Proof. We will reduce the decision version of PC in trees to the decision version of PMS in trees. Given a PC instance and an integer k we will construct a PMS instance with time distances between nodes equal to one time unit and period $T = 2$. Theorem 7 implies that it is possible to achieve a solution of value at least $\frac{T}{k}$ if and only if the original PC instance can be colored with at most k colors.

□

Theorem 8. *Given a ρ-approximation algorithm for* PC *in bidirectional trees, Algorithm 4 achieves an approximation ratio of $\frac{1}{\rho}\frac{L}{L+1}$.*

Proof. The key observation is that if $\mathrm{OPT}_{\mathrm{PMS}} \geq \frac{T}{\mathrm{OPT}_{\mathrm{PC}}-1}$ then by using algorithm 4 we could achieve a coloring with $\mathrm{OPT}_{\mathrm{PC}} - 1$ colors, which is a contradiction. Therefore $\mathrm{OPT}_{\mathrm{PMS}} < \frac{T}{\mathrm{OPT}_{\mathrm{PC}}-1}$ and the rest of the proof follows along the lines of the proof of Theorem 4.

□

Corollary 4. *There is a $\left(\frac{3}{5}\frac{L}{L+1}\right)$-approximation algorithm for* PMS *in trees where the time distances between nodes are multiples of $\frac{T}{2}$.*

Proof. By using Theorem 8 and the $\frac{5}{3}$-approximation algorithm of Erlebach et al. [9].

□

6 Conclusions

We have introduced the Periodic Metro Scheduling problem, which aims at generating a periodic timetable for a given set of routes and a given time period, in such a way that the minimum time distance between successive trains is maximized.

We have presented exact algorithms for chain and spider networks, and constant ratio approximation algorithms for ring networks, as well as for a special class of tree networks. Some of our algorithms make use of a reduction to Path Coloring. We have left open the question of the approximability of PMS in general tree networks. Another interesting open question is the variation where only the end stations of a route are given and one should determine both a path for each route and a departure time; such a variation applies in topologies that contain cycles, such as rings, grids and trees of rings.

References

1. L. Anderegg, S. Eidenbenz, M. Gantenbein, C. Stamm, D.S. Taylor, B. Weber, and P. Widmayer. Train Routing Algorithms: Concepts, Design Choices, and Practical Considerations. Proceedings of The Fifth Workshop on Algorithm Engineering and Experiments (ALENEX'03), pp. 106–118, 2003.
2. A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17:271–281, 1987.
3. M.R. Bussieck, T. Winter, and U. Zimmermann. Discrete optimization in public rail transport. *Math. Program.* 79: 415-444 (1997).
4. M. C. Carlisle and E. L. Lloyd, "On the k-coloring of intervals," *Discrete Applied Mathematics*, vol. 59, pp. 225–235, 1995.
5. G. Carpaneto, M. Dell'Amico, M. Fischetti, and P. Toth. A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19:531–548, 1989.
6. G. Dantzig and D. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Nav. Res. Logistics Q.*, 1:217–222, 1954.
7. S. Eidenbenz, A. Pagourtzis, and P. Widmayer. Flexible Train Rostering. Proc. ISAAC 2003, LNCS 2906, pp. 615–624, Springer-Verlag 2003.
8. T. Erlebach, M. Gantenbein, D. Hürlimann, G. Neyer, A. Pagourtzis, P. Penna, K. Schlude, K. Steinhöfel, D.S. Taylor, and P. Widmayer. On the Complexity of Train Assignment Problems. Proc. ISAAC 2001, LNCS 2223, pp. 390–402, Springer-Verlag 2001.
9. T. Erlebach, K. Jansen, C. Kaklamanis, M. Mihail, and P. Persiano. Optimal wavelength routing in directed fiber trees. *Theoretical Computer Science*, 221:119–137, 1999.
10. P.J. Fioole, L.G. Kroon, G. Maróti, A. Schrijver. A rolling stock circulation model for combining and splitting of passenger trains. CWI Report PNA-E0420, ISSN 1386-3711, 2004. To appear in *European Journal of Operational Research*.
11. Z. Genç. Ein neuer Ansatz zur Fahrplanoptimierung im ÖPNV: Maximierung von zeitlichen Sicherheitabständen. PhD thesis, Universität zu Köln, Mathematisch-Naturwissenschaftliche Fakultät, Institut für Informatik, 2003. Available at http://kups.ub.uni-koeln.de/volltexte/2003/950/
12. M. Gatto, B. Glaus, R. Jacob, L. Peeters, and P. Widmayer. Railway delay management: Exploring its algorithmic complexity. In Algorithm Theory - Proceedings SWAT 2004, pages 199–211. Springer-Verlag LNCS 3111, 2004.
13. M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arc graphs and chords. *SIAM Journal of Discrete Math*, vol. 1, no. 2, pp. 216-227, 1980.

14. M. Gatto, R. Jacob, L. Peeters, A. Schöbel. The Computational Complexity of Delay Management. In Proc. WG 2005, LNCS 3787, pp. 227–238, Springer 2005.
15. F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph, SIAM J. on Computing, 1(1972)180-187.
16. I.A. Karapetian. On the coloring of circular arc graphs. Docladi (Reports) of the Academy of Science of the Armenian Soviet Socialist Republic, vol. 70(5), pp. 306–311, 1980, in Russian. English translation by D. Gamarnik, D. Williamson and N. Edwards in http://www.orie.cornell.edu/~nedwards/wdm-routing/
17. V. Kumar. Approximating circular arc coloring and bandwidth allocation in alloptical ring networks. Lecture Notes in Computer Science 1444 (1998), 147-158, Springer-Verlag 1998.
18. L.G. Kroon and L.W.P. Peeters. A Variable Trip Time Model for Cyclic Railway Timetabling. *Transportation Science* vol. 37 no. 2 pp. 198-212, INFORMS, 2003.
19. C. Liebchen and R.H. Möhring. A Case Study in Periodic Timetabling. *Electr. Notes Theor. Comput. Sci.* 66(6), 2002.
20. C. Liebchen. A Cut-Based Heuristic to Produce Almost Feasible Periodic Railway Timetables. In Proc. WEA 2005, LNCS 3503, pp. 354-366, Springer-Verlag 2005.
21. A. Löbel. Optimal vehicle scheduling in public transit *PhD thesis, TU Berlin*, 1998.
22. Milena Mihail, Christos Kaklamanis, Satish Rao: Efficient Access to Optical Bandwidth - Wavelength Routing on Directed Fiber Trees, Rings, and Trees of Rings. FOCS 1995: 548-557.
23. P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4), 550–581, (1989).
24. A. Schöbel. A model for the delay management problem based on mixedinteger-programming. In Christos Zaroliagis, editor, *Electronic Notes in Theoretical Computer Science*, volume 50. Elsevier, 2001.
25. A. Schrijver. Routing and Timetabling by Topological Search. *Documenta Mathematica*, Extra Volume ICM III (1998), 687-695.
26. A. Schrijver. Minimum circulation in railway stock. *CWI Quarterly*, 6(3):205–217, 1993.

QoS-aware Multicommodity Flows and Transportation Planning *

George Tsaggouris[1,2], Christos Zaroliagis[1,2]

[1] Computer Technology Institute, N. Kazantzaki Str,
Patras University Campus, 26500 Patras, Greece
[2] Department of Computer Engineering and Informatics,
University of Patras, 26500 Patras, Greece
{tsaggour,zaro}@ceid.upatras.gr

Abstract. We consider the *QoS-aware Multicommodity Flow* problem, a natural generalization of the weighted multicommodity flow problem where the demands and commodity values are elastic to the Quality-of-Service characteristics of the underlying network. The problem is fundamental in transportation planning and also has important applications beyond the transportation domain. We provide a FPTAS for the QoS-aware Multicommodity Flow problem by building upon a Lagrangian relaxation method and a recent FPTAS for the non-additive shortest path problem.

1 Introduction

Consider a capacitated directed network $G = (V, E)$ in which we wish to route k commodities to meet certain initial demands. Each commodity i is associated with a specific origin-destination pair (s_i, t_i), a demand d_i, and a value v_i representing the *profit* of routing one unit of flow from that commodity. Also, for each commodity i, a weight $wt_i : E \to I\!R_0^+$ is defined that quantifies the provided *quality of service (QoS)* when this commodity is routed along an edge e or a path p, where $wt_i(p) = \sum_{e \in p} wt_i(e)$. Smaller weight means better QoS. When a commodity is not routed along its shortest w.r.t. wt_i (optimal w.r.t. the QoS) path due to capacity restrictions, then (i) a portion of its demand d_i drops (the worse the QoS of the path, the larger the portion of d_i that is lost), and (ii) its value v_i is reduced (the worse the QoS, the larger the reduction). In other words, demands and values are *elastic* to the provided QoS. The objective is to compute the maximum weighted multicommodity flow (sum over all commodities and over all paths of the flow routed from every commodity on each path multiplied by the commodity's value) subject to the QoS-elastic demands and values. We call the above the *QoS-aware Multicommodity Flow* (MCF) problem.

The QoS-aware MCF problem is a natural generalization of the weighted MCF problem that (is motivated by and) plays a key role in transportation

* This work was partially supported by the FET Unit of EC (IST priority – 6th FP), under contracts no. FP6-021235-2 (ARRIVAL) and no. IST-2002-001907 (DELIS).

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/682

planning: one of the prime issues that planners of transport operators in public transportation networks have to deal with concerns the routing of various commodities (customers with common origin-destination pairs) to meet certain demands [9,13,14]. A customer, when provided with a non-optimal path (route) due to unavailable capacity, s/he will most likely switch to another operator or even other means of transport and the probability in doing so increases as the QoS drops (actually, as a result of statistical measurements over several years, major European railway companies know quite accurately the percentage of customers they lose in such cases as a function of the path's QoS [9,14]). To minimize the loss of customers, the value charged for the requested service is usually reduced to make the alternative (worse in QoS) path, offered for that service, attractive.

Consequently, transportation planners are confronted with the following network and line planning issues:

- Which is the maximum profit obtained with the current capacity policy that incurs certain QoS-elastic demands and values?
- How much will this profit improve if the capacity is increased?
- Which is the necessary capacity to achieve a profit above a certain threshold?

A fast algorithm for the QoS-aware MCF problem would allow transportation planners to address effectively such network and line planning issues by identifying capacity bottlenecks and proceed accordingly.

It is worth mentioning that the QoS-aware MCF problem is also fundamental in applications beyond the transportation domain. For instance, in networking (e.g., multimedia) applications over the Internet [8], or in information dissemination over various communication networks [3]. In such a setting, a "server" (owned by some service provider) sends information to "clients", who retrieve answers to queries they have posed regarding various types of information. Common queries are typically grouped together. Answering a query incurs a cost and a data acquisition time that depends on the communication capacity. When a "client" is provided with an non-optimal service (e.g., long data acquisition time due to capacity restrictions), s/he will most likely switch to another provider. On the other hand, the provider may reduce the cost of such a service in order to minimize the loss.

A related problem, called *max-flow with QoS guarantee* (QoS max-flow), has been considered in [2]. The problem asks for computing the maximum (unweighted) MCF routed along a set of paths whose cost does not exceed a specific bound, and has been shown to be NP-hard in [2]. In the same paper [2], a pseudopolynomial time approximation scheme for QoS max-flow is given. It can be easily seen that QoS max-flow is a special case of the QoS-aware MCF problem (Section 5).

In this paper, we show that the QoS-aware MCF problem can be formulated (in a non-straightforward manner) as a fractional packing LP, and provide a FPTAS for its approximate solution. Our algorithm builds upon the Garg & Könemann Lagrangian relaxation method for fractional packing LPs [5], combined with the phases technique by Fleischer [4]. A crucial step of the method is

to construct an oracle that identifies the most violated constraint of the dual LP. While in the classical weighted MCF problem the construction of the oracle is harmless (reduces to the standard, single objective shortest path problem), this is *not* the case with the QoS-aware MCF problem. The construction turns out to be non-trivial, since it reduces to a multiobjective (actually non-additive) shortest path problem due to the QoS-elastic demands and values. Building upon a recent FPTAS for non-additive shortest paths [12], we are able to construct the required oracle and hence provide a FPTAS for the QoS-aware MCF problem. Our approach gives also a FPTAS for the QoS max-flow problem, thus improving upon the result in [2].

The rest of the paper is organized as follows. We start (Section 2) with some necessary preliminaries, a formal definition of the problem and its LP formulation. We then proceed (Section 3) with a review of the GK method [5] upon which our algorithm builds. Subsequently, we give the details of our FPTAS (Section 4), and present extensions of our results to constrained versions of the QoS-aware problem (Section 5). We conclude in Section 6.

2 Preliminaries and Problem Formulation

2.1 Problem Definition and LP Formulation

We are given a digraph $G = (V, E)$, along with a capacity function $u : E \to I\!R_0^+$ on its edges. We are also given a set of k commodities. A commodity i, $1 \leq i \leq k$, is a tuple $(s_i, t_i, d_i, wt_i(\cdot), \mathrm{f_i}(\cdot), \mathrm{v_i}(\cdot))$, whose attributes are defined as follows. Attributes $s_i \in V$ and $t_i \in V$ are the source and the target nodes, respectively, while $d_i \in I\!R_0^+$ is the demand of the commodity. The weight function $wt_i : E \to I\!R_0^+$ quantifies the *quality of service (QoS)* for commodity i (smaller weight means better QoS). For any s_i-t_i path p, $wt_i(p) := \sum_{e \in p} wt_i(e)$ and let $\delta_i(s_i, t_i)$ be the length of the shortest path from s_i to t_i w.r.t. the weight function $wt_i(\cdot)$. The non-decreasing function $\mathrm{f_i} : [1, \infty) \to [0, 1]$ is the *elasticity function* of i that determines the portion $\mathrm{f_i}(x)$ of the commodity's demand d_i that is lost if the provided path is x times worse than the shortest path w.r.t. $wt_i(\cdot)$; that is, if a units of d_i were supposed to be sent in case the provided path was shortest (optimal), then only $(1 - \mathrm{f_i}(x))a$ units will be shipped through the actually provided (non-optimal) path, while $\mathrm{f_i}(x)a$ units will be lost. Commodity i is also associated with a non-increasing *profit function* $\mathrm{v_i} : [1, \infty) \to I\!R_0^+$ that gives the profit $\mathrm{v_i}(x)$ from shipping one unit of flow of commodity i through a path that is x times worse than the shortest path w.r.t. $wt_i(\cdot)$. The objective is to maximize the total profit, i.e., the sum over all commodities and over all paths of the flow routed from every commodity on each path multiplied by the commodity's profit, subject to the capacity and demand constraints, and w.r.t. the QoS-elasticity of demands and profits. We call the above the *QoS-aware Multicommodity Flow* (MCF) problem.

Let $P_i = \{p : p \text{ is an } s_i\text{-}t_i \text{ path}\}$ be the set of *candidate paths* along which flow from commodity i can be sent. Consider such a particular path $p \in P_i$ and let $X_i(p) \in I\!R_0^+$ denote the flow of commodity i routed along p. The definition

of the elasticity function implies that for each unit of flow of commodity i routed along p, there are $\frac{1}{1-\mathrm{f_i}(x)}$ units *consumed* from the demand of the commodity. Thus, we define a *consumption function* $\mathrm{h_i} : [1, \infty) \to [1, \infty)$ with $\mathrm{h_i}(x) = \frac{1}{1-\mathrm{f_i}(x)}$. Since $\mathrm{f_i}$ is non-decreasing, $\mathrm{h_i}$ is also non-decreasing. Accordingly, we define the *consumption* $h_i(p) \geq 1$ of a path p as the amount of demand consumed for each unit of flow routed along p:

$$h_i(p) = \mathrm{h_i}\left(\frac{wt_i(p)}{\delta_i(s_i, t_i)}\right).$$

Similarly, we define the *value* $v_i(p)$ of a path p as the profit from routing one unit of flow of commodity i through p:

$$v_i(p) = \mathrm{v_i}\left(\frac{wt_i(p)}{\delta_i(s_i, t_i)}\right).$$

Using the above definitions, the QoS-aware MCF problem can be described by the following linear program (LP).

$$\max \sum_{i=1}^{k} \sum_{p \in P_i} v_i(p) X_i(p) \tag{1}$$

$$s.t. \sum_{i=1}^{k} \sum_{e \in p, p \in P_i} X_i(p) \leq u(e), \forall e \in E \tag{2}$$

$$\sum_{p \in P_i} X_i(p) h_i(p) \leq d_i, \forall i = 1 \ldots k \tag{3}$$

$$X_i(p) \geq 0, \forall i = 1 \ldots k, \forall p \in P_i$$

2.2 Non-Additive Shortest Paths

In this section, we introduce the non-additive shortest path (NASP) problem that will be used as a subroutine in our FPTAS for the QoS-aware MCF problem.

In NASP, we are given a digraph $G = (V, E)$ and a d-dimensional function vector $\mathbf{c} : E \to [I\!R^+]^d$ associating each edge e with a vector of attributes $\mathbf{c}(e)$ and a path p with a vector of attributes $\mathbf{c}(p) = \sum_{e \in p} \mathbf{c}(e)$. We are also given a d-attribute non-decreasing and *non-linear* utility function $\mathcal{U} : [I\!R^+]^d \to I\!R$. The objective is to find a path p^*, from a specific source node s to a destination t, that minimizes the objective function, i.e., $p^* = \mathrm{argmin}_{p \in P(s,t)} \mathcal{U}(\mathbf{c}(p))$. (It is easy to see that in the case where $\mathcal{U}$ is linear, NASP reduces to the classical single-objective shortest path problem.) For the general case of non-linear $\mathcal{U}$, it is not difficult to see that NASP is NP-hard.

The first FPTAS for NASP were independently presented in [11] (for any $d > 1$ and polynomially bounded utility function) and in [1] (for $d = 2$ and quasi-polynomially bounded utility function), with the former having a better time complexity. Recently, an improved FPTAS for NASP was given [12] that holds

for *any* $d > 1$ and a larger than quasi-polynomially bounded family of utility functions. The new result improves considerably upon those in [1,11] w.r.t. time (dependence on $1/\varepsilon$), number of objectives, and class of utility functions.

The following lemma is an immediate consequence of [12, Theorem 4].

Lemma 1. *Let the utility function of NASP be of the form* $\mathcal{U}([x_1, x_2]^T) = x_1\mathcal{U}_1(x_2) + \mathcal{U}_2(x_2)$*, where* $\mathcal{U}_1, \mathcal{U}_2$ *are any non-negative and non-decreasing functions. Then, for any* $\varepsilon > 0$*, there is an algorithm that computes an* $(1+\varepsilon)$*-approximation to the optimum of NASP in time* $O(n^2 m \frac{\log(nC_1)}{\varepsilon})$*, where* $C_1 = \frac{\max_{e \in E} c_1(e)}{\min_{e \in E} c_1(e)}$.

3 Review of the GK Method

The linear program for the QoS-aware MCF problem (given in Section 2.1) is a (pure) fractional packing LP, i.e., a linear program of the form $\max\{c^T x | Ax \leq b, x \geq 0\}$, where $A_{M\times N}$, $b_{M\times 1}$ and $c_{N\times 1}$ have positive entries. By scaling we also assume that $A(i,j) \leq b(i)$, $\forall i, j$. The dual of that problem is $\min\{b^T y | A^T y \geq c, y \geq 0\}$. In [5], Garg and Könemann present a remarkably elegant and simple FPTAS for solving fractional packing LPs. Their algorithm maintains a primal and a dual solution. At each step they identify the most violated constraint in the dual and increase the corresponding primal variable, and the dual variables. The most violated constraint is identified by using an exact oracle.

The algorithm works as follows. Let the *length* of a column j with respect to the dual variables y be $\text{length}_y(j) = \sum_i \frac{A(i,j)}{c(j)} y(i)$. Let $a(y)$ denote the length of the minimum-length column, i.e., $a(y) = \min_j \text{length}_y(j)$. Let also $D(y) = b^T y$ be the dual objective value with respect to y. Then, the dual problem is equivalent to finding an assignment y that minimizes $\frac{D(y)}{a(y)}$. The procedure is iterative. Let y_{k-1} be the dual variables and f_{k-1} be the value of the primal solution at the beginning of the k-th iteration. The initial values of the dual variables are $y_0(i) = \delta/b(i)$, where δ is a constant to be chosen later, and the primal variables are initially zero. In the k-th iteration, a call to an oracle is made that returns the minimum length column q of A, i.e., $\text{length}_{y_{k-1}}(q) = \alpha(y_{k-1})$. Let now $p = \text{argmin}_i \frac{b(i)}{A(i,q)}$ be the "minimum capacity" row. In this iteration, we increase the primal variable $x(q)$ by $\frac{b(p)}{A(p,q)}$, thus the primal objective becomes $f_k = f_{k-1} + c(q)\frac{b(p)}{A(p,q)}$. The dual variables are updated as

$$y_k(i) = y_{k-1}(i)\left(1 + \varepsilon \frac{b(p)/A(p,q)}{b(i)/A(i,q)}\right),$$

where $\varepsilon > 0$ is a constant depending on the desired approximation ratio. For brevity we denote $a(y_k)$ and $D(y_k)$ by $a(k)$ and $D(k)$, respectively. The procedure stops at the first iteration t such that $D(t) \geq 1$. The final primal solution constructed may not be feasible since some of the packing constraints may be

violated. However, scaling the final value of the primal variables by $\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$ gives a feasible solution (see Lemma 6 in the Appendix).

The above algorithm can be straightforwardly extended to work with an approximate oracle[1]. Simply, in the k-th iteration we call an oracle that returns an $(1+w)$-approximation of the minimum length column of A. If q is the column returned by the oracle, then we have that $\text{length}_{y_{k-1}}(q) \leq (1+w)a(y_{k-1})$. By working similarly to [5] and choosing $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$, we can show the following theorem (whose proof is in the Appendix for the sake of completeness).

Theorem 1. *There is an algorithm that computes an $(1-\varepsilon)^{-2}(1+w)$-approximation to the packing LP after at most $M\lceil \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta} \rceil = M\lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon} M) \rceil$ iterations, where M is the number of rows.*

4 The FPTAS for QoS-aware Multicommodity Flows

In this section, we describe the (non-straightforward) details of solving the QoS-aware MCF problem by building upon the GK method. We start by obtaining the dual of the LP formulation of the QoS-aware MCF problem. We introduce for each edge e a dual variable $l(e)$ that corresponds to the capacity constraint (2) on e, and for each commodity i we introduce a dual variable ϕ_i that corresponds to the demand constraint (3) on i. The dual LP becomes

$$\min \quad D = \sum_{e \in E} l(e)u(e) + \sum_{i=1}^{k} \phi_i d_i \tag{4}$$

$$s.t. \quad l(p) + \phi_i h_i(p) \geq v_i(p), \forall i = 1 \ldots k, \forall p \in P_i \tag{5}$$

$$l(p) \geq 0, \forall i = 1 \ldots k, \forall p \in P_i \quad \phi_i \geq 0, \forall i = 1 \ldots k$$

where $l(p) := \sum_{e \in p} l(e)$.

To apply the GK method, it must hold $u(e) \geq 1$, $\forall e \in E$, and $d_i \geq h_i(p)$, $\forall 1 \leq i \leq k$, $p \in P_i$. To ensure this, we scale the capacities and demands by $\min\{\min_{e \in E} u(e), \min_{1 \leq i \leq k} \frac{d_i}{h_i^{max}}\}$, where $h_i^{max} = h_i(\frac{(n-1)\max_{e \in E} wt_i(e)}{\delta_i(s_i,t_i)})$ is an upper bound on the maximum possible value of $h_i(\cdot)$.

Given an assignment (l, ϕ) for the dual variables, the length of a dual constraint is defined as $\text{length}_{(l,\phi)}(i,p) = \frac{l(p)+\phi_i h_i(p)}{v_i(p)}$ and the length of the most violated constraint is denoted by $a(l,\phi) = \min_{1 \leq i \leq k} \min_{p \in P_i} \text{length}_{(l,\phi)}(i,p)$. The algorithm maintains a dual variable $l(e)$ for each edge e, initially equal to $\frac{\delta}{u(e)}$, and a dual variable ϕ_i for each commodity i, initially equal to $\frac{\delta}{d_i}$, where $\delta = (1+\varepsilon)((1+\varepsilon)(m+k))^{-\frac{1}{\varepsilon}}$.

The algorithm proceeds in iterations. Initially all flows are zero. In each iteration, it makes a call to an oracle that returns a commodity i' and a path

[1] Such an extension of the GK approach to work with approximate oracles was known before [6], and its combination with the phases technique of Fleischer [4] for solving packing problems has been first observed by Young [15] for solving the more general case of mixed packing LPs.

$p \in P_{i'}$ that approximately minimizes $\text{length}_{(l,\phi)}(i,q)$ over all $1 \le i \le k$ and $q \in P_i$; i.e., we have $\text{length}_{(l,\phi)}(i',p) \le (1+\varepsilon)a(l,\phi)$. It then augments $\Delta = \min\{\frac{d'_i}{h'_i(p)}, \min_{e\in p} u(e)\}$ units of flow from commodity i' through p and updates the corresponding dual variables by setting $l(e) = l(e)(1+\varepsilon\frac{\Delta}{u(e)})$, $\forall e \in p$, and $\phi_{i'} = \phi_{i'}(1+\varepsilon\frac{\Delta h_{i'}(p)}{d_{i'}})$. The algorithm terminates at the first iteration for which $D = \sum_{e\in E} l(e)u(e) + \sum_{i=1}^{k} \phi_i d_i > 1$, and scales the final flow by $\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}$.

We now turn to the most crucial step of the algorithm: to build a suitable approximate oracle to identify the most violated constraint (5) of the dual. Our task is to approximately minimize, overall $1 \le i \le k$ and $q \in P_i$, the function

$$\frac{l(q)+\phi_i h_i(q)}{v_i(q)} = \frac{l(q)+\phi_i \cdot \mathrm{h}_i\left(\frac{wt_i(q)}{\delta_i(s_i,t_i)}\right)}{\mathrm{v}_i\left(\frac{wt_i(q)}{\delta_i(s_i,t_i)}\right)}.$$

Note that for a fixed i, this requires the solution of a NASP instance with objective function $\mathcal{U}([x_1,x_2]^T) = \frac{x_1+\phi_i \mathrm{h}_i\left(\frac{x_2}{\delta_i(s_i,t_i)}\right)}{\mathrm{v}_i\left(\frac{x_2}{\delta_i(s_i,t_i)}\right)}$ and cost vector $\mathbf{c} = [l, wt_i]^T$. Note also that the above function is of the form required by Lemma 1 with $\mathcal{U}_1(x) = \frac{1}{\mathrm{v}_i\left(\frac{x}{\delta_i(s_i,t_i)}\right)}$ and $\mathcal{U}_2(x) = \phi_i \cdot \frac{\mathrm{h}_i\left(\frac{x}{\delta_i(s_i,t_i)}\right)}{\mathrm{v}_i\left(\frac{x}{\delta_i(s_i,t_i)}\right)}$. Consequently, we can apply Lemma 1 for any fixed i and make use of a non-additive shortest path routine $\bar{p} = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$ that returns an s_i-t_i path $\bar{p}$ that approximately (within $(1+\varepsilon)$) minimizes the above function, overall $q \in P_i$, in time $O(n^2 m \frac{\log(nL)}{\varepsilon})$, where $L = \frac{\max_{e\in E} l(e)}{\min_{e\in E} l(e)}$.

To efficiently implement the oracle, we do not call the NASP routine for every value of i. Instead, the oracle proceeds in phases (like in [4]), maintaining a lower bound estimation $\overline{a}$ of $a(l,\phi)$, initially equal to $\overline{a} = \frac{1}{1+\varepsilon}\min_{1\le i\le k}\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} | p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$. In each phase, the oracle examines the commodities one by one by performing NASP computations. For each commodity i the oracle returns a path $p = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$ for which $\frac{l(p)+\phi_i h_i(p)}{v_i(p)} \le \overline{a}(1+\varepsilon)^2$. As long as such a path can be found, the oracle sticks to commodity i. Otherwise, it continues with commodity $i+1$. After all k commodities are considered in a phase, we know that $a(l,\phi) \ge (1+\varepsilon)\overline{a}$ and proceed to the next phase by setting $\overline{a} = (1+\varepsilon)\overline{a}$. The pseudocodes of our algorithm and the oracle are given in Fig. 1.

To discuss correctness and time bounds, we start with the following lemma that establishes an upper bound on the ratio of the lengths of the minimum length column at the start and the end of the GK algorithm.

Lemma 2. *Let $a(0)$ and $a(t)$ be the lengths of the minimum length column at the start and the end of the algorithm, respectively. Then, $\frac{a(t)}{a(0)} \le \frac{1+\varepsilon}{\delta}$.*

Proof. By the initial values of the dual variables, we have $a(0) = \min_j \sum_i \frac{A(i,j)}{c(j)} y_0(i) = \delta \cdot \min_j \sum_i \frac{A(i,j)}{c(j)b(i)}$. Since now the algorithm stops at the first iteration t such

QoS-MCF($G, u, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d}, \boldsymbol{wt}, \boldsymbol{v}, \varepsilon$) {
  forall $e \in E$ { $l(e) = \frac{\delta}{u(e)}$ }
  for $i = 1$ to k { $\phi_i = \frac{\delta}{d_i}$ }
  for $i = 1$ to k { **forall** $e \in E$ { $X_i(e) = 0$ }}
  $D = (m + k)\delta$;
  for $i = 1$ to k { $p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$ }
  $\overline{a} = \frac{1}{1+\varepsilon} \min_{1 \le i \le k} \left\{ \frac{l(p_i) + \phi_i h_i(p_i)}{v_i(p_i)} \right\}$;
  $i = 1$;
  while $D \le 1$ {
    $(p, i, \overline{a}) = \text{QoS-MCF-oracle}(G, \boldsymbol{s}, \boldsymbol{t}, l, \boldsymbol{wt}, \boldsymbol{v}, \boldsymbol{\phi}, \varepsilon, i, \overline{a})$;
    $\Delta = \min\{ \frac{d_i}{h_i(p)}, \min_{e \in p} u(e) \}$;
    $X_i(p) = X_i(p) + \Delta$;
    forall $e \in p$ **do** $l(e) = l(e)(1 + \varepsilon \frac{\Delta}{u(e)})$;
    $\phi_i = \phi_i(1 + \varepsilon \frac{\Delta h_i(p)}{d_i})$;
    $D = D + \varepsilon \Delta \frac{l(p) + \phi_i h_i(p)}{v_i(p)}$;
  }
  for $i = 1$ to k { **forall** $e \in E$ { $X_i(e) = X_i(e) / \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$ }}
}

QoS-MCF-oracle($G, \boldsymbol{s}, \boldsymbol{t}, l, \boldsymbol{wt}, \boldsymbol{v}, \boldsymbol{\phi}, \varepsilon, j, \overline{a}$) {
  while true {
    for $i = j$ to k {
      $p = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$;
      if $\frac{l(p) + \phi_i h_i(p)}{v_i(p)} \le \overline{a}(1 + \varepsilon)^2$
        return $(p, i, \overline{a})$;
    }
    $\overline{a} = \overline{a}(1 + \varepsilon)$; /* update rule for
                    next phase */
  }
}

Fig. 1. The approximation algorithm for the QoS-MCF problem.

that $D(t) > 1$ and the dual variables increase by at most $1+\varepsilon$ in each iteration, it holds that $D(t) \leq 1+\varepsilon$. Consequently, $\sum_i b(i)y_t(i) \leq 1+\varepsilon$, which implies that $y_t(i) \leq (1+\varepsilon)\frac{1}{b(i)}$, $\forall i$. Hence, $a(t) = \min_j \sum_i \frac{A(i,j)}{c(j)} y_t(i) \leq (1+\varepsilon) \cdot \min_j \sum_i \frac{A(i,j)}{c(j)b(i)} = \frac{1+\varepsilon}{\delta} a(0)$. □

The following lemma establishes the approximation guarantee for the oracle.

Lemma 3. *A call to the oracle returns an $(1+\varepsilon)^2$-approximation of the most violated constraint in the dual.*

Proof. Let $\overline{a}_j$ be the value of $\overline{a}$ during the j-th phase of the algorithm. It suffices to show that for all phases $j \geq 1$, $\overline{a}_j \leq a(l,\phi)$.

Initially ($j=1$), we set $\overline{a}_1 = \frac{1}{1+\varepsilon} \min_{1\leq i\leq k}\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} | p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$. By the definition of the NASP routine, we get $\overline{a}_1 \leq \frac{1}{1+\varepsilon} \min_{1\leq i\leq k}\{(1+\varepsilon)\min_{p\in P_i} \frac{l(p)+\phi_i h_i(p)}{v_i(p)}\} = a(l,\phi)$.

For any subsequent phase $j > 1$, consider phase $j-1$. The oracle finishes the examination of a commodity i and proceeds with $i+1$ only when a call to $\text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$ in phase $j-1$ returns a path p_i for which $\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} > \overline{a}_{j-1}(1+\varepsilon)^2$. This inequality and the definition of the NASP routine imply that at the end phase $j-1$, we have for each commodity i

$$\overline{a}_{j-1}(1+\varepsilon)^2 < (1+\varepsilon) \min_{p\in P_i} \frac{l(p)+\phi_i h_i(p)}{v_i(p)}.$$

Hence, by the definition of $a(l,\phi)$, and since $l(e)$ can only increase during the algorithm, at the end of the phase we have $\overline{a}_{j-1}(1+\varepsilon)^2 < (1+\varepsilon)a(l,\phi)$. Since $\overline{a}_j = \overline{a}_{j-1}(1+\varepsilon)$, we get $\overline{a}_j < a(l,\phi)$. □

To establish a bound on the time complexity of the algorithm, we need to count the number of NASP computations. Clearly, at most one NASP computation is needed per augmentation of flow. The rest of NASP computations (not leading to an augmentation) are bounded by k times the number of phases. The following lemma establishes a bound on the total number of phases.

Lemma 4. *The number of phases of algorithm QoS-MCF is bounded by $\lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k))\rceil + 2$.*

Proof. Let $a(0)$ and $a(t)$ be the lengths of the most violated constraint at the start and the end of the algorithm, respectively. Let now $\overline{a}_j$ be the value of $\overline{a}$ during the j-th phase of the algorithm, and T be the last phase of the algorithm.

Initially, we set $\overline{a}_1 = \frac{1}{1+\varepsilon} \min_{1\leq i\leq k} \left\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} | p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\right\}$ and by the definition of the NASP routine we get that $a(0) \leq (1+\varepsilon)\overline{a}_1$. From the proof of Lemma 3, we have that $\overline{a}_T \leq a(t)$, and from Lemma 2 we get that $a(t) \leq \frac{1+\varepsilon}{\delta} a(0)$. Combining the last three inequalities we get $\overline{a}_T \leq \frac{(1+\varepsilon)^2}{\delta}\overline{a}_1$. By the update rule for $\overline{a}$ on each phase, we have that $\overline{a}_T = \overline{a}_1(1+\varepsilon)^{T-1}$, and therefore $\overline{a}_1(1+\varepsilon)^{T-1} \leq \frac{(1+\varepsilon)^2}{\delta}\overline{a}_1$, which implies that $T \leq \log_{1+\varepsilon} \frac{(1+\varepsilon)^3}{\delta}$. Hence, the

number of phases is bounded by $\lceil \log_{1+\varepsilon} \frac{(1+\varepsilon)^3}{\delta} \rceil = \lceil \frac{1}{\varepsilon}(1 + \log_{1+\varepsilon}(m+k)) \rceil + 2$, since $\delta = (1+\varepsilon)((1+\varepsilon)(m+k))^{-\frac{1}{\varepsilon}}$. □

We are now ready for the main result of this section.

Theorem 2. *There is an algorithm that computes an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the QoS-aware MCF problem in time $O((\frac{1}{\varepsilon})^3(m+k)\log(m+k)mn^2(\frac{1}{\varepsilon}\log(m+k)+\log(nU))$, where n is the number of nodes, m is the number of edges, k is the number of commodities, and $U = \frac{\max_{e\in E} u(e)}{\min_{e\in E} u(e)}$.*

Proof. From Theorem 1 (with $M = m + k$) and Lemma 3 we have that the algorithm computes an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the optimal and terminates after at most $(m+k)\lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k)) \rceil$ augmentations. Since for each phase at most k NASP computations do not lead to an augmentation, we get from Lemma 4 that the oracle performs at most $k\lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k)) \rceil + 2k$ NASP computations not leading to an augmentation. Therefore, the total number of NASP computations during an execution of the algorithm is $O(\frac{1}{\varepsilon}(m+k)\log_{1+\varepsilon}(m+k)) = O((\frac{1}{\varepsilon})^2(m+k)\log(m+k))$.

A NASP computation is carried out in time $O(\frac{1}{\varepsilon}n^2 m \log(nL))$, where $L = \frac{\max_{e\in E} l(e)}{\min_{e\in E} l(e)}$. From the initialization of $l(e)$, and since they can only increase during the algorithm, it is clear that $\min_{e\in E} l(e) \geq \frac{\delta}{\max_{e\in E} u(e)}$. Since now the algorithm stops at the first iteration such that $\sum_{e\in E} l(e)u(e) + \sum_{i=1}^{k} \phi_i d_i > 1$ and the dual variables increase by at most $1+\varepsilon$ in each iteration, it holds that $\sum_{e\in E} l(e)u(e) + \sum_{i=1}^{k} \phi_i d_i \leq 1+\varepsilon$. Consequently at the end of the algorithm we have $l(e) \leq \frac{(1+\varepsilon)}{u(e)}$, $\forall e \in E$, and thus $\max_{e\in E} l(e) \leq \frac{1+\varepsilon}{\min_{e\in E} u(e)}$. Hence, $L \leq \frac{1+\varepsilon}{\delta}U$. By our choice of $\delta = (1+\varepsilon)((1+\varepsilon)(m+k))^{-\frac{1}{\varepsilon}}$, we have that $L \leq ((1+\varepsilon)(m+k))^{\frac{1}{\varepsilon}}U$, and hence the time required for a NASP computation is $O(\frac{1}{\varepsilon}mn^2(\frac{1}{\varepsilon}\log(m+k)+\log(nU)))$. Thus, we get an algorithm that computes an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the QoS-aware MCF problem in time $O([(\frac{1}{\varepsilon})^2(m+k)\log(m+k)][\frac{1}{\varepsilon}mn^2(\frac{1}{\varepsilon}\log(m+k)+\log(nU)]) = O((\frac{1}{\varepsilon})^3(m+k)\log(m+k)mn^2(\frac{1}{\varepsilon}\log(m+k)+\log(nU))$, which is polynomial to the input and $\frac{1}{\varepsilon}$. □

5 Extensions

Better bounds can be obtained for the *constrained* version of the QoS-aware MCF problem. In that version all profits are constant (non QoS-elastic) — i.e., $v_i(p) = v_i$, $\forall i, p \in P_i$ — and there is an upper bound (constraint) on the QoS per path provided — i.e., the consumption functions are now defined as:

$$h_i(p) = \begin{cases} 1 & \text{if } wt_i(p) \leq b_i \\ +\infty & \text{otherwise} \end{cases}, \quad \forall i, p \in P_i.$$

The objective is to maximize the total profit. In this case, we can achieve a FPTAS by implementing the oracle using a FPTAS for the Restricted Shortest Path

(RSP) problem instead of NASP. The currently best FPTAS for general digraphs is due to Lorenz and Raz [7], and runs in $O(mn(\log\log n+1/\varepsilon))$ time. Arguing as in Theorem 2 and taking into account the time complexity of the FPTAS for RSP [7], we can achieve a running time of $O((\frac{1}{\varepsilon})^2(m+k)\log(m+k)mn(\log\log n+1/\varepsilon))$ for the constrained version of the QoS-aware MCF problem.

For the version of the problem with unbounded demands, which constitutes the QoS max flow problem defined in [2], we can achieve a better time bound of $O((\frac{1}{\varepsilon})^2 nm^2 \log m(\log\log n + 1/\varepsilon))$, since the number of constraints in its corresponding LP is m.

6 Conclusions

We considered the QoS-aware MCF problem, a natural and important generalization of the weighted multicommodity flow problem with elastic demands and values that is fundamental in transportation planning (and beyond). We formulated the problem as a fractional packing LP, and provided a FPTAS for its solution by building upon a Lagrangian relaxation method combined with a recent FPTAS for non-additive shortest paths. Finally, we presented better FPTAS for constrained versions of the QoS-aware MCF problem.

Acknowledgments. We are indebted to Naveen Garg, Jochen Könemann, Spyros Kontogiannis, Frank Geraets (aka Wagner) for various useful discussions, and to Christos Papadimitriou for bringing [2] to our attention.

References

1. H. Ackermann, A. Newman, H. Röglin, and B. Vöcking, "Decision Making Based on Approximate and Smoothed Pareto Curves", in *Algorithms and Computation – ISAAC 2005*, LNCS Vol. 3827 (Springer 2006), pp. 675-684; full version as Tech. Report AIB-2005-23, RWTH Aachen, December 2005.
2. K. Chaudhuri, C. Papadimitriou, and S. Rao, "Optimum Routing with Quality of Service Constraints", manuscript, 2004.
3. A. Datta, D. Vandermeer, A. Celik, and V. Kumar, "Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users", *ACM Transactions on Database Systems*, 24:1 (1999), pp. 1-79.
4. L.K. Fleischer, "Approximating fractional multicommodity flows independent of the number of commodities", *SIAM Journal on Discrete Mathematics*, 13:4 (2000), pp. 505-520.
5. N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems", in *Proc. 39th IEEE Symposium on Foundations of Computer Science* – FOCS'98, (IEEE CS Press, 1998), pp.300-309.
6. N. Garg and J. Könemann, personal communication, 2005.
7. D.H. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest path problem", *Operations Research Letters*, 28 (2001) pp.213-219.

8. P. Van Mieghem, F.A. Kuipers, T. Korkmaz, M. Krunz, M. Curado, E. Monteiro, X. Masip-Bruin, J. Sole-Pareta, and S. Sanchez-Lopez, "Quality of Service Routing", Chapter 3 in *Quality of Future Internet Services*, LNCS Vol. 2856 (Springer-Verlag, 2003), pp. 80-117.
9. PIN project (Projekt Integrierte Netzoptimierung), Deutsche Bahn AG, 2000.
10. S. Plotkin, D. Shmoys, and E. Tardos, "Fast Approximation Algorithms for Fractional Packing and Covering Problems", *Mathematics of Operations Research* 20 (1995), pp. 257-301.
11. G. Tsaggouris and C. Zaroliagis, "Improved FPTAS for Multiobjective Shortest Paths with Applications", CTI Techn. Report TR-2005/07/03, July 2005.
12. G. Tsaggouris and C. Zaroliagis, "Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-linear Objectives with Applications", CTI Techn. Report TR-2006/03/01, March 2006. Preliminary version in Proc. ISAAC 2006, LNCS (Springer, 2006).
13. F. Wagner, "Challenging Optimization Problems at Deutsche Bahn", AMORE Workshop (invited talk), 1999.
14. F. Wagner (Deutsche Bahn AG), personal communication, 2004.
15. N. Young, "Sequential and Parallel Algorithms for Mixed Packing and Covering", in *Proc. 42nd IEEE Symp. on Foundations of Computer Science* – FOCS 2001, pp. 538-546.

A APPENDIX

A.1 Proof of Theorem 1

The analysis is straightforward from [5]. We only consider an approximate oracle. In order to prove Theorem 1 we need the next two lemmata. In the first lemma, we establish a bound on the ratio of the optimal dual value to the primal objective value at the end of the algorithm.

Lemma 5. *Let* $\beta = \min_y \frac{D(y)}{a(y)}$ *be the optimal dual value and let* t *be the last iteration of the algorithm. The ratio of the optimal dual value to the primal objective value at the end of the algorithm is bounded by* $\frac{\varepsilon(1+w)}{\ln(1/M\delta)}$.

Proof. For each iteration $k \geq 1$ it is

$$\begin{aligned} D(k) &= \sum_i b(i) y_k(i) \\ &= \sum_i b(i) y_{k-1}(i) + \varepsilon \frac{b(p)}{A(p,q)} \sum_i A(i,q) y_{k-1}(i) \\ &\leq D(k-1) + (1+w)\varepsilon (f_k - f_{k-1}) a(k-1) \end{aligned}$$

which implies that

$$D(k) \leq D(0) + (1+w)\varepsilon \sum_{l=1}^{k} (f_l - f_{l-1}) a(l-1).$$

Since $\beta = \min_y D(y)/a(y)$ it is $\beta \leq D(l-1)/a(l-1), \forall l = 1 \ldots k$, and thus

$$D(k) \leq M\delta + \frac{(1+w)\varepsilon}{\beta} \sum_{l=1}^{k} (f_l - f_{l-1}) D(l-1).$$

Observe now that for fixed k, this right hand side is maximized by setting $D(l-1)$ to its maximum possible value for all $1 \leq l-1 < k$, and let us denote this maximum value by $D'(k)$, i.e.,

$$D'(k) = M\delta + \frac{(1+w)\varepsilon}{\beta} \sum_{l=1}^{k} (f_l - f_{l-1}) D'(l-1).$$

Consequently,

$$\begin{aligned}
D(k) &\leq D'(k) \\
&= D'(k-1) + \frac{(1+w)\varepsilon}{\beta}(f_k - f_{k-1}) D'(k-1) \\
&= D'(k-1)\left(1 + \frac{(1+w)\varepsilon}{\beta}(f_k - f_{k-1})\right) \\
&\leq D'(k-1) e^{\frac{(1+w)\varepsilon}{\beta}(f_k - f_{k-1})} \\
&\leq D'(0) e^{\frac{(1+w)\varepsilon}{\beta} \sum_{l=1}^{k} (f_l - f_{l-1})} \\
&\leq D'(0) e^{\frac{(1+w)\varepsilon}{\beta}(f_k - f_0)}
\end{aligned}$$

Since now $D'(0) = M\delta$ and $f_0 = 0$ it follows that

$$D(k) \leq M\delta e^{(1+w)\varepsilon f_k/\beta}.$$

From our stopping condition it is $1 \leq D(t) \leq M\delta e^{(1+w)\varepsilon f_t/\beta}$ and hence

$$\frac{\beta}{f_t} \leq \frac{\varepsilon(1+w)}{\ln(1/M\delta)}.$$

□

The final primal solution constructed may not be feasible since some of the packing constraints may be violated. The second lemma shows that the final primal assignment can be appropriately scaled as to obtain a feasible solution.

Lemma 6. *Scaling the final primal assignment by* $\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$*, we obtain a feasible solution to the fractional packing LP.*

Proof. When we pick a column q and increase the left-hand-side of the i-th constraint by $\frac{A(i,q)b(p)}{A(p,q)b(i)}$. Simultaneously we increase the dual variable $y(i)$ by a multiplicative factor of $1 + \varepsilon\frac{A(i,q)b(p)}{A(p,q)b(i)}$. By the definition of p it follows that $\frac{A(i,q)b(p)}{A(p,q)b(i)} \leq 1$ and thus increasing the left-hand-side of the i-th constraint by one

causes an increase in $y(i)$ by a multiplicative factor of $1+\varepsilon$. Since t is the first iteration for which $D(t) > 1$, it is $y_{t-1}(i) < 1/b(i)$ and thus $y_t(i) < (1+\varepsilon)/b(i)$. Since now $y_0(i) < \delta/b(i)$ it follows that the left-hand-side of the i-th constraint is no more than $\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ for any i. Thus scaling the primal solution by $\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ gives a feasible solution. □

We now proceed with the proof of Theorem 1.

Proof. In the k-th iteration we increase the dual variable of the "minimum capacity" row by a factor of $1+\varepsilon$. Since we stop the algorithm at the first iteration t such that $D(t) > 1$ it follows that $D(t) < 1+\varepsilon$ and thus $y_t(i) < \frac{1+\varepsilon}{b(i)}$ for any row. Since now $y_0(i) = \frac{\delta}{b(i)}$ and $y_t(i) < \frac{1+\varepsilon}{b(i)}$ and there are M rows the total number of iterations is at most $M\lceil\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}\rceil = M\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon}M\rceil$, by choosing $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$.

The ratio of the optimal dual value to objective value of the scaled final primal assignment is $\gamma = \frac{\beta}{f_t}\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ by substituting the bound on $\frac{\beta}{f_t}$ from Lemma 5 we get

$$\gamma \leq \frac{\varepsilon(1+w)}{\ln(1/M\delta)}\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right) = \frac{\varepsilon(1+w)}{\ln(1+\varepsilon)}\frac{\ln\frac{1+\varepsilon}{\delta}}{\ln(1/M\delta)}$$

For $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$, the ratio $\frac{\ln\frac{1+\varepsilon}{\delta}}{\ln(1/M\delta)}$ equals $(1-\varepsilon)^{-1}$, hence we have

$$\gamma \leq \frac{\varepsilon(1+w)}{(1-\varepsilon)\ln(1+\varepsilon)} \leq \frac{\varepsilon(1+w)}{(1-\varepsilon)(\varepsilon-\varepsilon^2/2)} \leq (1-\varepsilon)^{-2}(1+w)$$

□

Robustness and Recovery in Train Scheduling - a simulation study from DSB S-tog a/s

M. Hofman, L. Madsen, J. J. Groth, J. Clausen, J. Larsen

Department of Informatics and Mathematical Modelling, The Technical University of Denmark,jjg@imm.dtu.dk

Abstract. This paper presents a simulation model to study the robustness of timetables of DSB S-tog a/s, the city rail of Copenhagen. Dealing with rush hour scenarios only, the simulation model investigates the effects of disturbances on the S-tog network. Several timetables are analyzed with respect to robustness. Some of these are used in operation and some are generated for the purpose of investigating timetables with specific alternative characteristics.

1 Background

DSB S-tog (S-tog) is the sole supplier of rail traffic on the infrastructure of the city-rail network in Copenhagen. S-tog has the responsibility of buying and maintaining trains, ensuring the availability of qualified crew, and setting up plans for departures and arrivals, rolling stock, crew etc. The infrastructural responsibility and the responsibility of safety lie with Banedanmark, which is the company owning the major part of the rail infrastructures in Denmark.

The S-tog network consists of 170 km double tracks and 80 stations. At the most busy time of day the network presently requires 103 trains to cover all lines and departures, including 4 standby units. There are at daily level 1100 departures from end stations and additionally appr. 15.000 departures from intermediate stations. Figure 1 illustrates the current line structure covering the stations of the network.

All lines of the network have a frequency of 20 minutes and are run according to a cyclic timetable with a cycle of 1 hour. The frequency on stations in specific time periods as e.g. daytime is increased by adding extra lines to the part of the network covering these specific stations. This way of increasing frequency makes it easy for to customers to remember the line routing both in the regular daytime and in the early and late hours.

Each line must be covered by a certain number of trains according to the length of its route. The trains covering one line forms a circuit. The time of a circuit is the time it takes to go from one terminal to the other and back.

The network consists of two main segments, the small circular rail segment, running from Hellerup in the north to Ny Ellebjerg in the south, and the remaining major network. This consists of seven segments - six "fingers" and a central segment combining the fingers. A consequence of this structure is that a high

ATMOS 2006
6th Workshop on Algorithmic Methods and Models for Optimization of Railways
http://drops.dagstuhl.de/opus/volltexte/2006/687

Fig. 1. The DSB S-tog network according to the 2006 timetable

number of lines pass the central segment resulting in substantial interdependency between these lines. This interdependency makes the network very sensitive to delays and it is thus imperative to S-tog to reduce the line interdependency as much as possible in the early planning stages. The plans of timetable, rolling stock and crew should if possible be robust against disturbances of operations. It is, however, in general non-trivial to achieve such robustness.

1.1 Simulation

One way to identify characteristics regarding robustness is by simulating the operation of the network. Simulation helps identifying critical parts of the network, the timetable and the rolling stock and crew plans. One example is poor crew planning in relation to the rolling stock plan. It is unfortunate to have too little slack between two tasks of a driver, if the tasks involve two different sets of rolling stock.

Simulation also provides a convenient way to compare different types of timetables on their ability to maintain reliability in the operation. This allows better decisions to be made on a strategic level regarding which timetable to implement. Specifically, for the network structure of S-tog the number of lines intersecting the central segment has proven important to the stability in operation in the past. It has been a common understanding that an increasing number of lines passing the central segment will lead to a decreasing regularity.

Time slack is often used as a remedy for minor irregularities at the time of operation. Time slack can for example be added to running times along the route, dwell times on intermediate stations and turn around times at terminals. Common for these types of slack are that they are introduced at the time of timetabling in the planning phase.

It is common knownledge that time slack increases the ability of a timetable and a rolling stock plan to cope with the facts of reality, i.e. the unavoidable disturbances arising in operation. Slack in a plan is, however, costly since resources are idle in the slack time if no disturbance occurs. It is therefore not evident which type of slack to use, exactly where to use it, and how much to use.

The stability of a network is not only related to the "inner robustness" introduced through time slack. As noted earlier, slacks in the plans are intended to compensate for minor disturbances. When larger disturbances occur action must be taken to bring the plan back to normal. This process is called recovery. There are various types of recovering plans. For example, cancelling departures decreases the frequency of trains on stations, which in turn increases freedom in handling the disturbance.

The simulation model to be presented is used for testing various timetables with different characteristics. Also we use the model for testing some of the strategies of recovery used by rolling stock dispatchers at S-tog. Firstly, in Section 2, related literature on the subject is presented. Recovery strategies employed at S-tog are described in Section 3. In Section 4 we present the background for the simulation model, and Section 5 discusses assumptions and concepts of the model. The model itself is presented in Section 6, and the test setups and results are presented in sections 7 and 8. Finally, Section 9 gives our conclusions and suggestions for further work.

More details on the topic can be found in the M.Sc. thesis [5] by Hofman and Madsen.

2 Related work

Related work involves studies on robustness and reliability, simulation and recovery. The first subject area, robustness and reliability, focuses on identifying and quantifying robustness and reliability of plans. Simulation is used for various purposes within the rail industry, and the models of the various subjects often have similar characteristics. The area of recovery presents various strategies and systems for recovery. Systems are often based on optimization models.

2.1 Robustness and reliability studies

Analytical and simulation methods for evaluating stability are often too complex or computationally extremely demanding. The most common method is therefore using heuristic measures. In [1] Carey describes various heuristic measures of stability that can be employed at early planning stages. Carey and Carville [2] present a simulation model used for testing schedule performance regarding the

probability distribution of so-called secondary delays (knock-on effects) caused by the primary delays, given the occurrence of these and a schedule. The model is used for evaluating schedules with respect to the ability to absorb delays. In [13] Vroman, Dekker and Kroon present concepts of reliability in public railway systems. Using simulation they test the effect of homogenizing lines and number of stops in timetables. Mattsson [9] presents a literature study on how secondary delays are related to the amount of primary delay and the capacity utilization of the rail network. An analytic tool for evaluating timetable performance in a deterministic setting, PETER, is presented by Goverde and Odijk [4]. The evaluation of timetables is done without simulation, which (in contrast to simulation based methods) makes PETER suitable for quick evaluations.

2.2 Simulation studies

Hoogheimstra and Teunisse [6] presents a prototype of a simulator used for robustness study of timetables for the Dutch railway network. The simulation prototype is called the DONS-simulator and is used for generating timetables. Similarly, in [10] Middelkoop and Bouwman present a simulation model, Simone, for analysing timetable robustness. The model simulates a complete network and is used to identify bottlenecks. Sandblad et al. [12] offer a general introduction to simulation of train traffic. A simulation system is discussed with the multiple purposes of improving methods for train traffic planning, experimenting with developing new systems, and training of operators.

2.3 Recovery studies

In [3] Goodman and Takagi discuss computerized systems for recovery and various criteria for evaluating recovery. In particular, they present two main methods of implementing recovery strategies: Either recovering from a known set of recovery rules or optimizing the individual situation, i.e. determining the optimal recovery strategy for the specific instance at hand. A train holding model is presented in [11] by Puong and Wilson. The objective of the model is to minimize the effect of minor disturbances by levelling the distance between trains by holding them at certain times and places of the network. In [7] Kawakami describes the future framework of a traffic control system for a network of magnetically levitated high speed trains in Japan. Different recovery strategies are presented, one of which is increasing the speed of delayed trains.

3 Recovery strategies

When a timetable is exposed to disturbances and disruption occurs, it is crucial how the operation returns to normal, and how fast the strategy can be implemented. At present, the procedure of returning to a normal state of operation is manual with support from operation surveillance systems and a system showing the plan of operation constructed in advance. The different manual actions available are mainly the following:

Platform changes on-the-day It is planned in advance which platforms to use for the different train arrivals and departures at the time of operation. If a planned platform is occupied at the time of arrival of the next train, the train is rescheduled to another vacant platform if possible. For example, at Copenhagen Central (KH) there are two platforms in each direction. When one platform is occupied with a delayed train the trains can be lead to the other vacant platform for that direction.

Trains skipping stations i.e. making fast-trains out of stop-trains If a train is delayed it is possible to skip some of its stops at stations with minor passenger loads and few connecting lines. However, two consequtive departures on the same line cannot be skipped.

Shortening the routes of trains A train can be "turned around" before reaching its terminal i.e. the remainder of the stations on its route can be skipped, cf. Figure 2. Again, two consequtive trains cannot be turned.

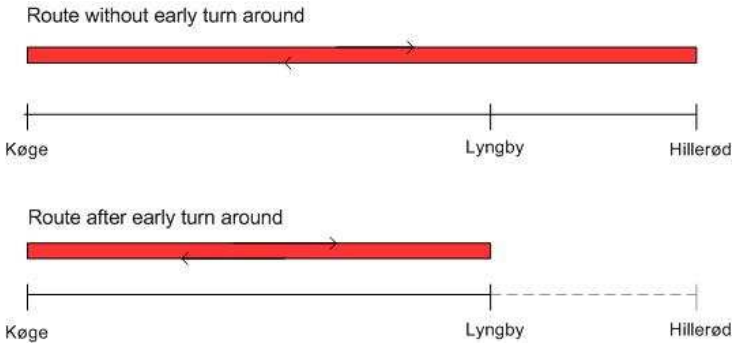


Fig. 2. The train movement at early turn around

Swapping the tasks/routes of fast-trains catching up with stop-trains On some of the segments of the network both slow trains stopping at all stations and faster trains that skip certain stations are running. Delays some times occur so that fast lines catch up with slow lines leading to a delay of the fast trains. Here, it is possible do a "virtual overtaking", i.e. to swap the identity of the two trains so that the slow train is changed to a fast train and vice versa.

Inserting replacement trains from KH for trains that are delayed Trains covering lines that intersect the central section run from one end of the network to the other passing Copenhagen Central. Here, a major rolling stock depot as well as a crew depot is located. If a train is delayed in the first part of its route, it is often replaced by another train departing on-time from KH. Thus, a new train is set in operation at KH, which proceeds on the route of the delayed train. This is on arrival at KH taken out of operation.

Inserting replacement trains for trains that have broken down In case of rolling stock failure the train is replaced by new unit of rolling stock from a nearby depot.

Reducing dwell times to a minimum At stations there are pre-decided dwell times. These vary with the different passenger flows of the stations and with different special characteristics such as a driver depot. The latter demands extra time for the releasing of drivers. In the case of a disruption the dwell times on all stations are reduced to minimum.

Reducing headways to a minimum In the outer ends of the network there are some slack on the headways. In the case of delays headways are reduced making the trains drive closer to each other. As the frequency of trains in the central section is high there is less slack here for decreasing headways.

Reducing running times to a minimum Timetables are constructed given predefined running times between all sets of adjacent stations. The running time is always the minimum running time plus some slack. In case of a disruption, running times between all stations are reduced to a minimum given the particular context.

Allowing overtaking on stations with available tracks Handling operations is less complex if there is a predetermined order of train lines. In the case of a disruption the predetermined order of lines can be broken on stations with several available platforms in the same direction i.e. where overtaking between trains is possible. This is for example used when a fast train reaches a delayed stop train at KH.

Cancelling of entire train lines In the case of severe disruption entire lines are taken out, i.e. all trains currently servicing the departures on the relevant lines are taken out of operation. In the case of severe weather conditions such as heavy snow, the decision is taken prior to the start of the operation.

The main components in recovery strategies are increasing headways or exploiting slack in the network, called respectively re-establishing and re-scheduling. The first handles disturbances by employing prescheduled buffers in the plans. The latter refers to the handling of disturbances by making some changes in the plan to bring the situation back to normal. The ways of changing the plan are in most cases predefined.

4 Background of the problem

4.1 Planning and designing timetables

In S-tog the first phase of timetabling consists of deciding the overall line-structure of the train network. The basis for the decision includes various criteria such as number of passengers on the different fingers, passenger travel-patterns and rotation time of lines. Regarding the latter criteria, it is from a crewing perspective an advantage to keep the rotation time at a level matching a reasonable duration for driver-tasks. In the next phase the stopping patterns are decided automatically from input such as driving time, minimum headways and

turn-around times. In the third phase, we then verify whether the plan is feasible with respect to rolling stock. These first three first phases are all carried out internally in S-tog. The following phases involve various other parties, each of which evaluates the proposed timetable, including BaneDanmark and the National Rail Authority. When all involved parties have accepted the timetable, the phase of rolling stock planning begins.

The process of designing and constructing a timetable is exceedingly long. It is made up by the long process of constructing possible timetables that might be rejected in other phases of the process, thereby forcing the process of timetabling to be highly iterative. Many stakeholders are involved in the decision of which timetable to implement in operation, and these may very well have conflicting interests. In all phases of the timetabling process there is an urgent need for being be able to discuss specific plans both qualitatively and quantitatively. Quantitative information can be obtained by simulation. Often it is an advantage not to have too many details in the input of a simulation. To compare different timetables it may e.g. not be necessary to know all details about tracks and signals. Therefore, a decision regarding the timetable to be developed for operation may be taken early in the planning process.

4.2 Disturbances at S-tog

The disturbances at S-tog can be classified into categories at several levels leading to various actions when experienced during operations. First of all, disturbances are categorized as being the consequence of some specific primary incident as e.g. rolling stock defects (causing speed reductions), passenger's questions to the train driver, illness of a driver, or signal problems (forcing the trains to stop). We distinguish between primary incidents caused by the rail system (trains, rails, passengers etc.) and driver related incidents.

Incidents with a very long duration and complete breakdowns of the system are considered as a separate type of incidents. An example of a complete breakdown is the fall-down of overhead wires.

Secondary incidents occur as a consequence of primary incidents. These incidents occur because primary incidents have influenced the operation, forcing trains to stop or to slow down. The slack present in the timetable and the number of secondary incidents that usually occur during operation are directly related. That is, when slack is decreased the number of secondary delays increases and vice versa.

The general measures of disturbances in the S-tog network are termed regularity and reliability. These refer respectively to lateness and cancellations in the network. Regularity is calculated as

$$\left(1 - \frac{LateDepartures}{DeparturesinTotal}\right) * 100\%$$

Traffic is considered stable when regularity exceeds a limit of 95%. A departure is late when it is delayed more than 2.5 minutes. Reliability is calculated as

$$\left(\frac{ActualDepartures}{ScheduledDepartures}\right) * 100\%$$

Contractually, reliability must be higher than 97% over the day.

4.3 Recovery strategies

Implementing different recovery strategies in a simulation model makes it possible to evaluate, which actions lead to the quickest recovery and least sizeable disruption with respect to affected trains. We have chosen to investigate three specific S-tog strategies for recovery. These have been implemented in the simulation model and are evaluated individually i.e. two different recovery strategies are not employed at the same time in any of the presented test-cases. The three recovery strategies chosen were "Early turn around", "Insertion of on-time trains on KH" and "Cancelling of entire train lines". All of these recovery strategies are frequently used in operation. They each contribute to increased headways in some segment of the network. Furthermore, these three methods of recovery are employed both in case of smaller and of medium size delays. Also they have varying effects on customer service level.

Early turn around increases headways in the part of the network not serviced because of the early turn around, and the train catches up on schedule in the following departures. As a result, the number of secondary delays is decreased as the train is often turned to an on-time departure. The negative consequences of the recovery strategy are that some departures are cancelled when the train is turned around before the end station of its route. This decreases the reliability. Also, it becomes difficult to locate the rolling stock according to the circular schedule, which must continue the following morning. In reality the trains are turned without any respect of the line of the train. The train simply turns and departs according to the first scheduled departure.

In the simulation model the strategy has been implemented with the costraint that two successive trains can not be turned, i.e. one of them must continue to the end station to meet passenger demands. Also, a train can not be turned in both ends of its route. The shortening of routes are, apart from these two constraints, invoked for each individual train by judging whether it is either more late than a certain threshold or more late than can be gained by using the buffer at the end station. In priciple, it is physically possible to turn around trains on all stations in the S-tog network. However, as only a subset of the larger stations are used for turn around in practice, these are also the only stations in the simulation model where turn around is feasible. In the model, a turned around train must match the departures that was originally planned for that particular train.

Cancelling of entire train lines is invoked by the condition of the regularity of the line in question. If the regularity of the line is below a certain threshold, the line or a predefined extra line on the same route is taken out. The line may be reinserted when the regularity again exceeds a certain lower limit and has been above this limit for a predefined amount of time. When put into action this

recovery strategy increases the headways on the segment of the network where the line in question runs. A positive effect of the recovery strategy is that the number of secondary delays decreases. As entire lines are cancelled, employing this strategy has a considerable negative impact on the reliability.

Specific characteristics of the recovery strategy are that trains on the line in question can only be taken out at rolling stock depots and that at the time of insertion it must be ensured that drivers are available at these depots. As drivers are not simulated in the model, the latter restriction is not included.

Insertion of on-time trains on KH is the strategy of replacing a late train with train being on-time from KH. This means that the time the network is serviced by the delayed train is decreased. Like the recovery strategy of shortening routes, this strategy is also employed when the relevant train is more than a predefined threshold late. The threshold limit is set by the duration of the buffer at end station. The strategy has no impact on the reliability as no trains are being cancelled. It does, though, have a limited positive effect on the regularity. As no headways are increased the headways are merely levelled out in the part of the route from KH to the end station. It is assumed in the model that only one train in each direction on the same line can be replaced at the same time. Hence, at least every second train services the entire line.

5 Assumptions

One of the difficulties in simulation modelling is to decide on the level of detail to use, i.e. to decide whether it is necessary to implement a very detailed model or whether trustworthy conclusions can be made on the basis of more coarse grained information. In the rail universe we have to determine whether signals and tracks must be modelled with high precision or whether it is sufficient to model a network with stations as the nodes and tracks between them as the edges.

Additional considerations regarding specific details must also be made. Below we describe the assumptions we have made in modelling the S-tog network.

All experiments are based on the worst case scenario of operating peak hour capacity throughout the simulation. This will not affect the validity of the results as stability and robustness are lowest when production and demand are highest.

We assume that the stopping pattern of each lines is constant over the day. In most cases, each line has a fixed individual stopping pattern over the day. Deviations do occur, especially in the early morning hours and in the evening. As we have chosen only to simulate peak hours not intersecting these time intervals, we assume that the stopping pattern for each line is fixed.

The stopping times of trains in the timetable are given with the accuracy of half a minute. Therefore, the train in reality arrives at a station approximately at the time defined by the timetable. Arrivals "before schedule" may thus occur. Since we do not allow a train to depart earlier than scheduled, these early arrivals have not been implemented in our simulation-model.

The circular rail segment has been omitted from the test scenarios. In general, it has a very high regularity and its interaction with the remainder of the network is very limited.

In the model, all minimum headways have been set to 1.5 minutes. This makes the model less exact than if minimum headways are kept at their real levels, which vary depending on the area of the network. In reality, network parts where trains drive with high speed have larger minimum headways than low speed parts. However, due to the heavy traffic the low speed parts constitute the bottleneck network parts.

In our model delays are added at stations. The alternative is to add delays between stations describing the track segment between two stations to some predefined detail. This, however, complicates the model without giving any additional benefits regarding the possible comparisons between time tables and recovery strategies.

Delays are genereated from delay-distributions of historical data. We hence assume that the delays in the system will occur mainly caused by the same events as they have done up till now. However, there may be a variation in delay patterns stemming from the structure of the timetable. Even if no timetable similar to the timetable in a test scenario have been in operation, the delays observed at stations in the past still seem to offer the best basis for generating delays for the test scenario in question.

The probability of delay on a station is set to 50%. This is estimated from the historical data as a worst case situation. Almost no time registrations are zero (i.e. the departure is exactly on time).

In our model, regaining time is only possible at stations and terminals and not while running between stations. Even though time can be gained between the stations in the outer part of the network, this is insignificant compared to what can be gained in the terminals. Again, it is clear that the regularity of a test case in real-life will be at least as good as the one observed in the simulation model, since extra possibilities for regaining lost time are present.

The single track of 500 m on a part between Værløse and Farum is not modelled. This is the only part of the network with a single track. As the single track part only accounts for 0.3% of the network this has no measurable effect on the results.

In the central section there are four junctions in the form of stations where lines merge and split up. To enable the use of a simple common station model, these junctions are not explicitly modelled in the simulation model. To compensate for this, virtual stations are introduced in the model. On the hub stations, where different sections of the network intersect, a station is added for merging or parting of the lines meeting at the hub. As a result of the extra station, the model merges and divides at slightly other times than in reality. An example of this is Svanemøllen (SAM). At SAM the northbound track divides into two. Hence, the lines that have passed the central section divide into two subsets. In the 2003 timetable, the subsets are two lines running towards Ryparken (RYT) and the remainder running towards Hellerup (HL). SAM is modelled as four sta-

tions; two stations where trains run towards respectively come from RYT and two that run towards respectively come from HL. Going south this means that when departing from SAM the trains must merge so no "crash" appears. When a station has several platforms in each direction, this is also handled in the model by adding in an extra station for each platform. For example, KH is modelled as four stations, two in each direction. This means that KH has two platforms available for each direction and can have up to four trains in the station at the same time.

The changes in the infrastructure since 2003 mostly concern the expansion of the circular rail of the network. Therefore, results obtained using the 2003 structure are still valid.

The simulation model is in general coarse grained and contains several minor modifications in relation to the facts of reality. Nevertheless, the model is adequate for comparing timetables and for evaluating the immediate impact of one recovery method compared to either one of the two other implemented recovery methods or no recovery cf. the text sections above.

6 The simulation model

The simulation model has been implemented in Arena [8], which is a general programming tool for implementing simulation models. The model is based on the circulations of rolling stock for each of the lines. Therefore, the main model of the simulation is built based on the lines. It has an entrance for each line where entities are created corresponding to the trains necessary to run the line. The trains circulate in a general station submodel common for all stations. A recovery method is given before the entities enter the station submodel and start iterating over it.

The input to the model is the line sequences, the departures, and various station information such as for example whether a particular station is a terminal, an intermediate stopping station or an intermediate non-stopping station, and the dwelling time at each station.

6.1 Station submodel

In the station submodel attributes are first updated for the next step and the next station respectively as these are used in the model relative to the current step and station. The model iterates over the stations in each line of the network. Therefore, the model reiterates from the beginning when the final station in the route is reached. Secondly, the attribute of direction is updated depending on the arriving train entity. Thirdly, the entity is put on hold if the station of the current step is occupied by another train. If the station is not occupied, the entity in question is allowed to enter the station. This is emphasized in the model by setting an "occupied" flag on the station. Thereafter, it is decided which type of station is entered, given the three possibilities.

The next action of the station submodel is handling the train dwelling time depending on the type of the station. If the train entity is set to stop at the station, the train is delayed by the predefined dwelling time. The dwelling time assigned depends on whether the train entity is already delayed from a previous station. If the train is delayed it should use the minimum dwelling time allowed. If not, it should use the standard dwelling time. No train can leave earlier than scheduled.

Next a possible delay is added. Delay is added at 50% of the stations. There are no delays added in the model before all trains have been introduced. Delays are added to the trains according to a distribution based on historical data.

The station is now marked unoccupied, as the train leaves the station after have performed its stop including dwelling time and possible delay. The regularity and the reliability are updated immediately after the station has been registered as unoccupied. These are calculated for each train on each of its stations. The overall regularity and reliability are the final averages of the individual values.

Now the entity enters some recovery method depending on which method was chosen initially. The method may be that no recovery action should be taken at all.

After recovery, the specific case of merging the lines B and B+ is handled in the submodel merge. If the line of the train entity is either the B or the B+ line and the current station is Høje Taastrup (HTAA), the trains merge and drive alternately B and B+ unless recovery has cancelled line B+. The merge is handled simply by alternating an attribute on the entity characterizing which line the train entity runs. If B+ has been cancelled, merging is not possible and the trains are instead delayed 10 minutes, which is the frequency between B and B+.

Routing is also handled in the station submodel. In the routing part, the train entity is routed from the current station to the next. First the train is held back to ensure sufficient headway. Next the train is held back in a queue until there is an open platform at the following station. There is a maximum number on the queue length identical to the space on tracks between stations in the S-tog network. If the current station is a terminal, the train can gain time and is routed to the same station in opposite direction otherwise it is routed to the next station in its line sequence without the possibility of gaining lost time. Finally, time is updated for the train entity with the driving from one station to the next.

6.2 Recovery submodels

Early turn-around The basic idea of this recovery method is that if a train is delayed more than a certain threshold, it will change direction at an intermediate station before it reaches the planned next terminal. This is checked in the beginning of the model together with a check of whether the line has been turned on its previous trip in the opposite direction.

If the current station is a possible turn-around station, the turn-around is performed and the next step and the starting time are decided. By creating a duplicate of the train entity turned around, it is possible to ensure that the following train is not also turned early.

Take Out This recovery method cancels specific lines in the network in case of disruption. The cancellation of lines are initiated by regularity falling below a certain threshold. When regularity has reattained another certain threshold, the method reinserts the trains on the cancelled line.

The candidates to be cancelled are predefined. For example, if delays are on line A, line A+ is cancelled.

Trains can only be taken out on depot stations. We assume the availability of drivers at the time of reinsertion. The method sets the train entities on hold. The cancellation of some entity is simply done by setting the train entities to be cancelled on hold and reinsertion is initiated by signalling. Time and station are then updated according to the time on hold and the line of the entity, and the train entity continues to run from that specific station along its planned line sequence.

Replace This recovery method inserts an on-time train from KH to replace a train delayed along its route, which is then taken out. It is activated when a train is more late than a certain threshold and the previous train was allowed to continue along its entire route.

The model of the method is divided in two. One handling the take out of trains at KH and one handling observation of delay at all other stations and scheduled insertion on KH. In the latter of these, a duplicate of the train entity is created to ensure that the train is taken out when it reaches KH.

It is at all times assumed that rolling stock is available at KH for inserting trains.

7 Test Cases

For the purpose of testing the simulation model 7 timetables has been used, some of which are run in several versions to make results more comparable. Two of the timetables are actual timetables of respectively 2003 with 10 lines intersecting the central section and 2006 with 9 lines intersecting the central section. They are both of the structure seen in Figure 1 Three timetables are potential timetables for years to come. They have respectively 10, 11 and 12 lines intersecting the central section. See Figure 3 and Figure 4. Finally, two artificial timetables have been constructed especially for the test session. The first of these has 19 lines on the fingers and 1 central metro line in the central section. The other has in total 17 lines, with a combination of circular and drive through lines in the central section. See Figure 5.

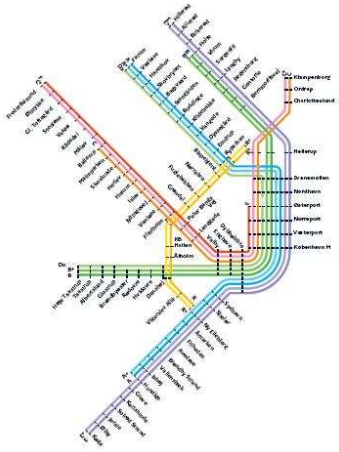


Fig. 3. Network with 10 lines through the central section

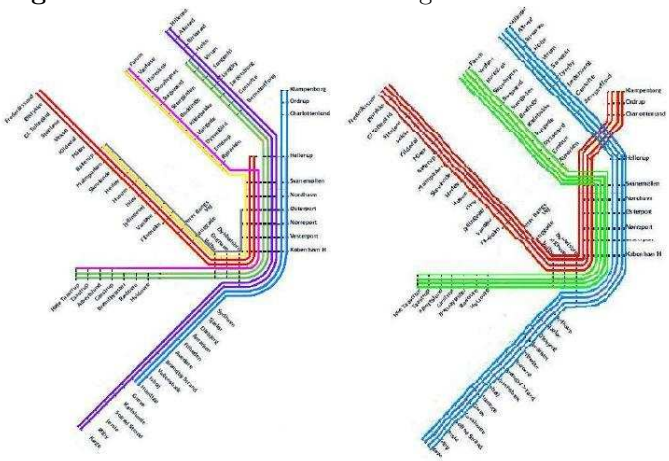


Fig. 4. Networks with respectively 11 and 12 lines through the central section

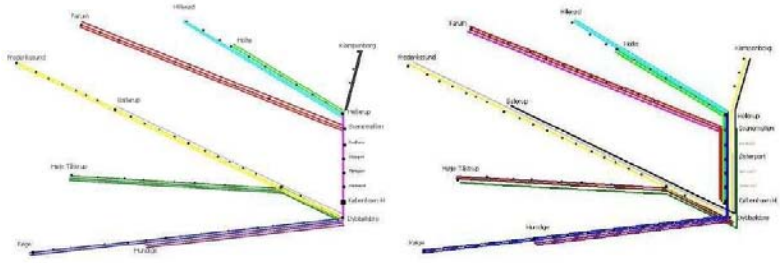


Fig. 5. Network on the left has one central metro line. Network on the right is a kombination of metro and through-going lines

The purpose of the test session with so different timetables is to test the effect of different characteristics such as a varied number of lines, different stopping patterns, line structures, cycle times, homogeneous use of double tracks, homogeneous scheduled headways and buffer times at terminals.

To make results comparable, changes have been made to some of the timetables. For example, lines have been extended and headways have been evened out.

The recovery methods have been tested with varying thresholds for activation of the methods. The *Early Turn around* and *Replace* methods have been tested for activation when the train in question is more late than respectively 2.5 minutes, 5 minutes, and "the amount of buffer time" at the terminal. For the *Cancellation* method, activation has been set at regularity falling below 80% without reinsertion, or 90% both with or without reinsertion. Reinsertion takes place when regularity increases above 95%. The recovery methods are not tested on the artificial timetables as these are so different from the timetables of today that recovery results are incomparable.

A series of tests were run with varying buffer time at terminals.

Tests with small and large delays are performed. In these test cases we have added respectively small delays, large delays and both large and small delays. The definition of small and large delays are derived from the historical data. The delays divide the stations into two subset of respectively 80 stations with small delays and 81 stations with large delays. For the first two of the three test scenarios, delay can hence only occur occur at 50 % of the stations. The tests are run with no recovery and 100% probability of delay on the relevant stations.

8 Computational Results

A variety of tests have been carried out with the simulation model. We have chosen to present specifically test results regarding *the comparison of timetables*, *the effect of large versus small delays on operation* and *varying sizes of terminal buffer times*. The complete set of tests is described in [5].

The main measures used for evaluating results are *regularity* and *reliability*. The registration in the simulation model starts when the start-up period is completed, i.e. when all trains has been inserted in the current model run.

When evaluating the results, it is also interesting to evaluate the cost of a timetable with respect to the number of trains necessary to maintain circulation. An optimal solution is a robust timetable operated by as few trains as possible. This is an obvious trade-off since fewer trains in a solution implies that the times of circuits for lines are decreased. The result is less "room" for slack in the timetable and therefore generally less robustness.

8.1 Comparing Timetables without recovery

A total of 12 different timetables has been tested with and without recovery. Figure 6 shows a plot of the regularity of different timetables run without recovery.

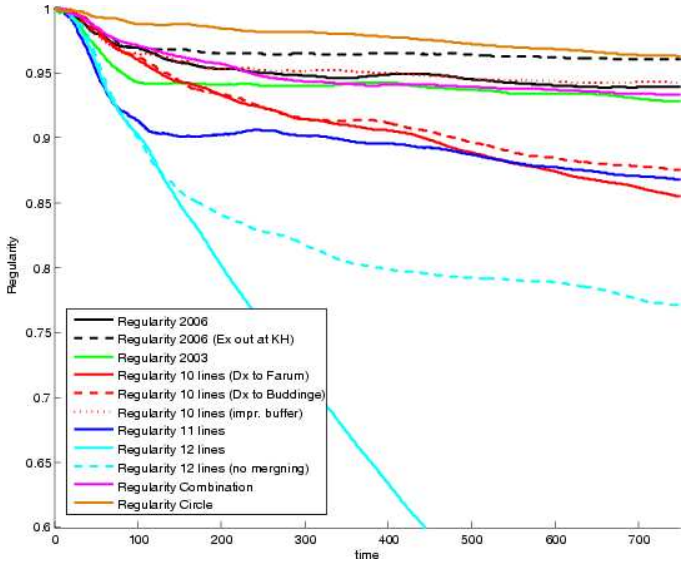


Fig. 6. Regularity of the 12 tested timetables where no recovery is applied

In general the number of lines have a high impact on regularity. Fewer lines implies an increase in regularity. It is, however, possible to improve timetables that has a high number of lines by increasing buffers on terminals. The results show that increased buffers improve the ability to "cope with" delays. An example of this is the timetable with 10 lines, cf. Figure 3.

8.2 Comparing Timetables using Turn-Around Recovery

The regularities of the timetables run with the turn-around recovery method are shown in Figure 7. The threshold for invoking the method has been set to the terminal buffer time used in the time tables.

Results show again that the number of lines significantly influences the level of regularity, however, the effect decreases with increasing number of lines. This is a consequence of more trains reaching the threshold and hence being turned, cf. Figure 8, where regularities of timetables are shown with a threshold for the turn-around recovery set to 5 minutes. The ranking of timetables with respect to level of regularity is here different from that of Figure 7. In addition, an overall better regularity on lines when using buffertimes as threshold can be observed.

8.3 Comparing Timetables using Cancellation of Lines Recovery

As expected, the results show that the cancellation of lines has a very positive effect on regularity. Corresponding to the positive effect on regularity, the recovery

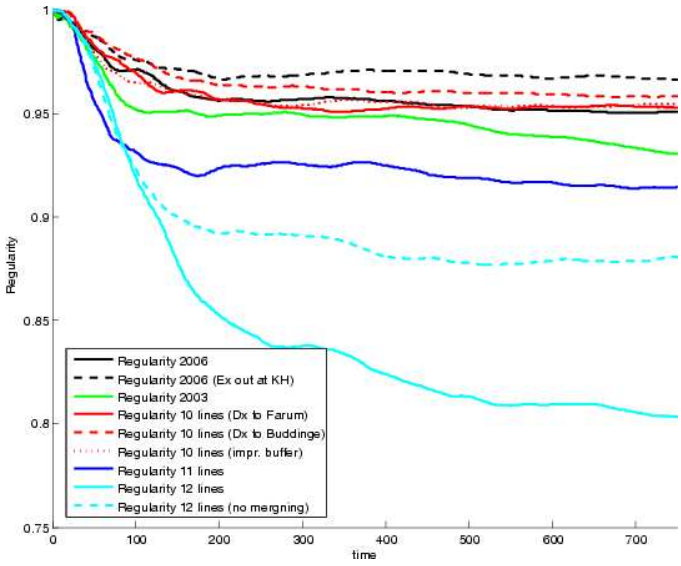


Fig. 7. Regularity of the 12 tested timetables where Turn Around recovery is applied

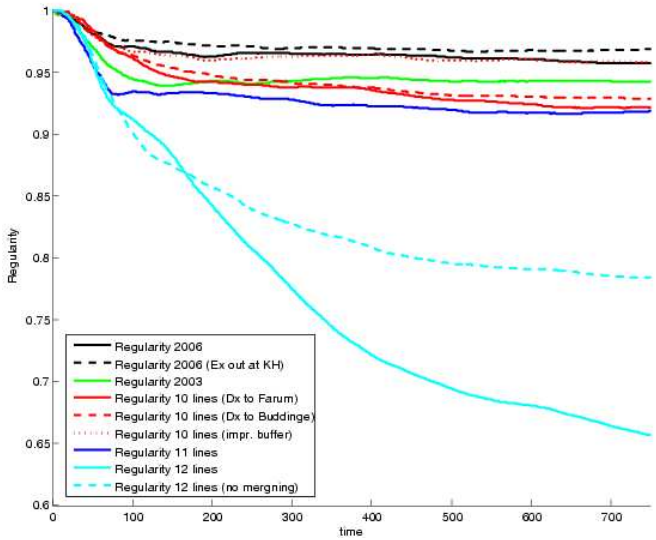


Fig. 8. Regularity of the 12 tested timetables where Turn Around recovery is applied when delay is higher than 5 minutes

method has a negative effect on reliability. That is, the majority of departures may be on time but only when a substantial part of the planned departures have been cancelled. The results for all timetables are given in Figure 9.

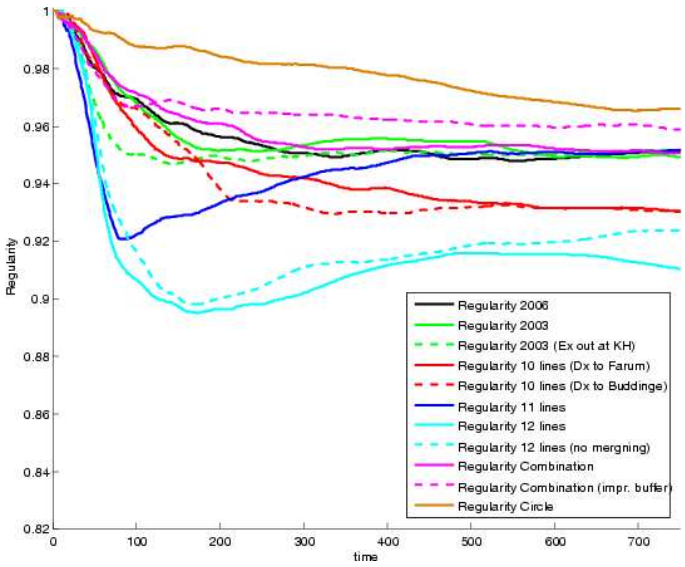


Fig. 9. Regularity of the 12 tested timetables where Cancellation recovery is applied when regularity is under 90%

8.4 Comparing Timetables using Replacement of Trains Recovery

This recovery method does not cancel any departures. Therefore the reliability is 100% in all test results. This also means that the headways are not increased when the recovery method is invoked. As expected this shows that the positive effect on regularity is less than for the other recovery methods.

8.5 Comparing the Effectiveness of Recovery Methods

If we compare the results of the "turn-around" with the "line-cancellation" recovery method, we see that the regularity of the "tun-around" is at the same level as the one of "line-cancellation" for timetables with a low number of lines. For timetables with high numbers of lines, only "line-cancellation" recovery brings up the regularity to a sufficiently high level.

Comparing recovery by replacement with the two other recovery methods, it is evident that the method does not have the same level of effect on the regularity as the two others when it comes to the timetables with many lines.

8.6 Testing the Effect of Large and Small Delays

The test results of running with small and large delays separately are shown in Figure 10 for timetables with 12 lines. Similar results were observed for other timetables.

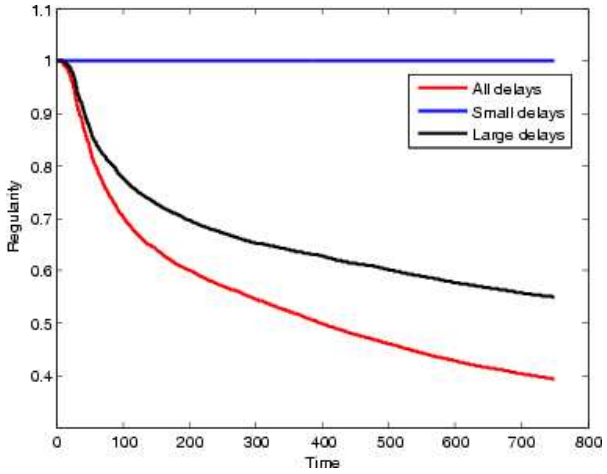


Fig. 10. Regularity when respectively only small delays, only large delays and all delays are applied

The figure shows a clear tendency: Small delays have almost no effect on the regularity when no large delays are present. The size of buffers are relatively large compared to the delays in the system. Large delays have a significant effect on the regularity as expected. When small delays are introduced in addition to the large delays, they have a much larger effect on propagation of delay than hen they occur on their own. It is, however, still obvious that larger delays has the largerst effect on regularity and that these if possible should be eliminated. Nevertheless, a substantial increase in regularity can be achieved through the removal of small delays, which is a much easier task.

8.7 Terminal Buffers

The terminal buffers has a substantial effect on regularity. There is often more available time at end stations than on intermediate stations with respect to the size of buffers. As buffers are larger on terminals, there is a better possibility to decrease an already incurred delay. Regarding the size of terminal buffers it is expected that increasing buffer times at terminals in general implies decreasing delays in the network. Test were run with increasing buffer times to confirm this. The increase in buffer time necessitate that one additional train is set into

rotation on specific lines. Hence the number of trains necessary to cover the line increases as the buffers on terminals are increased, cf. Table 1.

Timetable	Trains Needed
2003, 10 lines	73
2003, 10 lines and improved buffers on terminals	77
Constructed, 10 lines	67
Constructed, 10 lines and improved buffers on terminals	71
Constructed, 12 lines	93
Constructed, 12 lines and improved buffers on terminals	100
Combination	82
Combination, Improved buffers on terminals	88

Table 1. Number of trains running simultaneously in the tested timetables

The results show that in general regularity improves when buffers are increased, but also that there is an upper limit on the amount of buffer time, beyond which no extra regularity is gained, cf. Figure 11 and 12.

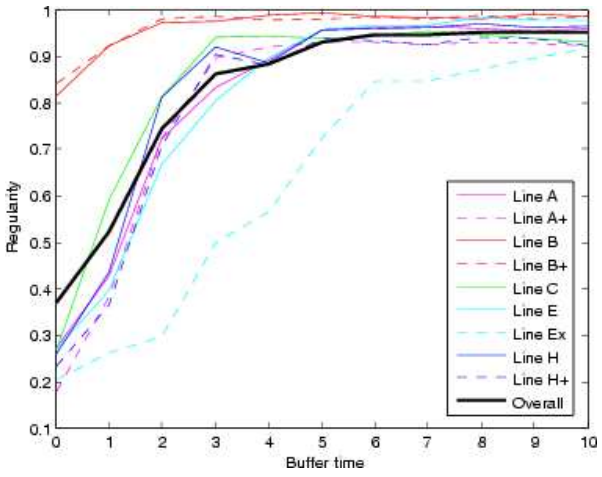


Fig. 11. Regularity on the lines of the timetable with 9 lines with different sizes of buffers on terminals

The improvement of regularity depends heavily on the timetable in question for each individual test. The timetable with 12 lines improves considerably more than the timetable with 9 lines.

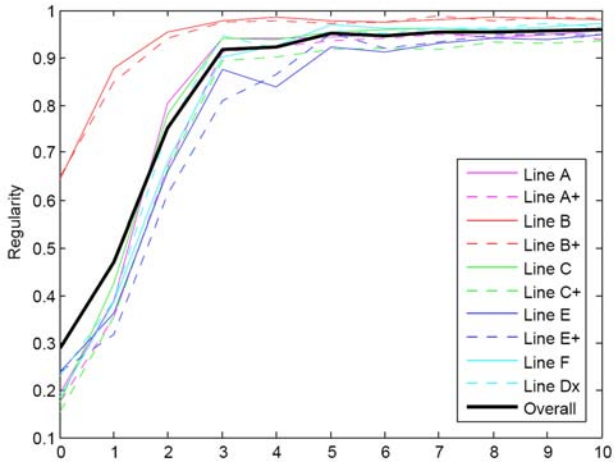


Fig. 12. Regularity on the lines of the timetable with 10 lines with different sizes of buffers on terminals

9 Conclusions and future work

We have presented a simulation model for testing timetable robustness and the effect on robustness of three different recovery strategies. The main results from our tests are that there is a upper limit on the amount of buffer time leading to positive effect on the regularity, and that small delays though insignificant on their own have a significant additional effect when occuring together with large delays. Finally, there is a clear tendency that the recovery methods rendering the largest increase in headways result in the best robustness and thereby the best increase in regularity.

Further work on the simulation model is to implement various others of the presented recovery methods. Also, simulating the operation during non-peak hours including the implementation of rules for change of train-formation is of ovbious interest. Furthermore, including the train drivers in the simulation will enable analysis of the dependency between timetables and crew plans, but will also require substantial additions and changes to the underlying model.

References

1. M. Carey and S. Carville. Exact heuristic measures of schedule reliability. *Journal of Operational Research Society*, 51:666–682, 2000.
2. M. Carey and S. Carville. Testing schedule performance and reliability for train stations. *International Transactions in Operational Research*, 11:382–394, 2004.
3. C. J. Goodman and R. Takagi. Dynamic re-scheduling of trains after disruptions. *Computers in Railways*, IX, 2004. WIT Press.
4. R. M. P. Goverde and M. A. Odijk. Performance evaluation of network timetables using peter. *Computers in Railways*, VIII, 2002. WIT Press.
5. M. Hofman and L. F. Madsen. Robustness in train scheduling. Master's thesis, The Technical University of Denmark, 2005.
6. J. S. Hoogheimestra and M. J. G. Teunisse. The use of simulation in the planning of the dutch railway services. In *Proceedings of the 1998 Winter Simulation Conference*, 1998.
7. T. Kawakami. Future framework for maglev train traffic control system utilizing autonomous decentralized architecture. *Automous Decentralized Systems*, pages 327–333, 1997.
8. W. D. Kelton, R. P. Sadowski, and D. T. Sturrock. *Simulation with Arena*. 3 edition, 2004.
9. L. G. Mattson. Train service reliability: A survey of methods for deriving relationship for train delays. https://users.du.se/jen/Seminarieuppsatser/Forsening-tag-Mattson.pdf.
10. D. Middelkoop and M. Bouwman. Simone: Large scale train network simulations. In *Proceedings of the 2001 Winter Simulation Conference*, pages 1042–1047. Institute of Electronical and Electronic Engineering, Piscataway, New Jersey, 2001.
11. A. Puong and N. H. M. Wilson. A train holding model for urban rail transit systems. In *Proceedings of Conference on Computer Aided Scheduling on Public Transport*, 2004. http://fugazi.engr.arizona.edu/caspt/puong.pdf.
12. Sandblad et al. T9 simulatorsystem inom tågtrafikstyring, en kundskabsdokumentation. Technical report, Upsala Universitet, 2003.
13. J. C. M. Vroman, R. Dekker, and L. G. Kroon. Reliability and heterogeneity of railway services. Technical report, Erasmus University Rotterdam, 2003.