

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems

ATMOS 2026, September 3–4, 2026, L'Aquila, Italy

Edited by

Valentina Cacchiani

Stefan Funke



Editors

Valentina Cacchiani 

DEI, University of Bologna, Italy
valentina.cacchiani@unibo.it

Stefan Funke 

Universität Stuttgart, Germany
funke@fmi.uni-stuttgart.de

ACM Classification 2012

Theory of computation → Design and analysis of algorithms; Mathematics of computing → Discrete mathematics; Mathematics of computing → Combinatorics; Theory of computation → Mathematical optimization; Mathematics of computing → Graph theory; Applied computing → Transportation

ISBN 978-3-95977-453-6

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-453-6>.

Publication date

October, 2026

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists all publications of this volume in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0):
<https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/OASlcs.ATMOS.2026.0

ISBN 978-3-95977-453-6

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

OASlcs – OpenAccess Series in Informatics

OASlcs is a series of high-quality conference proceedings across all fields in informatics. OASlcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Daniel Cremers (TU München, Germany)
- Barbara Hammer (Universität Bielefeld, Germany)
- Marc Langheinrich (Università della Svizzera Italiana – Lugano, Switzerland)
- Dorothea Wagner (*Editor-in-Chief*, Karlsruher Institut für Technologie, Germany)

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

Contents

Preface	
<i>Valentina Cacchiani and Stefan Funke</i>	0:vii
Committees	
.....	0:ix
List of Authors	
.....	0:xi

Invited Talk

Optimizing Last-Mile Deliveries at Amazon	
<i>Renato F. Werneck</i>	1:1–1:1

Regular Papers

Passenger-Based Intermodal Connection Optimization with Elastic Demands	
<i>Boris Grimm, Ralf Borndörfer, Andrea Fraioli, Paula Franke,</i> <i>Giovanni Luca Giacco, Federico Marinucci, and Tommaso Montino</i>	2:1–2:18
Time-Aware A* for Optimal Train Routing on Moving Block Systems	
<i>Stefan Engels and Robert Wille</i>	3:1–3:19
Why Start Over: Incremental MaxSMT for Operational Railway Scheduling	
<i>Thomas Dubach, Florian Flükiger-Fuchs, and Bernardo Martin-Iradi</i>	4:1–4:19
Dispatching-Area Design	
<i>Tomas Lidén, Christiane Schmidt, and Rabii Zahir</i>	5:1–5:17
From Routes to Resources: A Model for Real-Time Train Dispatching with No-Wait Constraints	
<i>Luka Stärk, Ralf Borndörfer, and Niels Lindner</i>	6:1–6:20
Adaptive Metaheuristics for Multi-Objective Berth Allocation and Scheduling	
<i>Konstantinos Karathanasis, Spyros Kontogiannis, Asterios Pegos,</i> <i>Vasileios Sofianos, and Christos Zaroliagis</i>	7:1–7:22
Finding Maximum-Success Disjoint Paths	
<i>Aaron Neugebauer and Marie Schmidt</i>	8:1–8:18
Scheduling Electric-Vehicle Charging Under Grid Capacity and Minimum Charge Rates	
<i>Bruna Cavalcanti Lauro, Matúš Mihalák, Filip Schlemmbach,</i> <i>and Evgueni Smirnov</i>	9:1–9:17
ASPaeroFlow: Decomposition Heuristics for Joint Air Traffic Flow & Capacity Management	
<i>Alexander Beiser, Markus Hecher, Nysret Musliu, Georg Trausmuth,</i> <i>and Stefan Woltran</i>	10:1–10:20

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke

 OpenAccess Series in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Integrated Line Planning and Strategic Ridepooling: Minimizing the Passengers’ Travel Time	
<i>Lena Dittrich, Sven Jäger, and Anita Schöbel</i>	11:1–11:18
A Multimodal Framework for Transport Planning	
<i>Lena Dittrich, Michael Rihlmann, Philine Schiewe, and Anita Schöbel</i>	12:1–12:20
Exploiting Treewidth to Solve the Pricing Problem in Column Generation for Non-Pool-Based Line Planning	
<i>Markus Niebisch and Tom C. van der Zanden</i>	13:1–13:20
Algorithmics for Safe Bicycle Network Design with Bounded Detours in Rural Areas	
<i>Till Fluschnik</i>	14:1–14:22
Large Scale Middle Mile Network Design Through Efficient Local Search	
<i>Carlo S. Sartori, Jian Shen, Anastasia Kireeva, Mattia Neroni, and Philipp Loick</i>	15:1–15:18
 Short Papers	
Computing the Cost of Stability in Periodic Microscopic Railway Timetabling via Logic-Based Benders Decomposition	
<i>Tongcheng Ouyang, Florian Flükiger-Fuchs, Bernardo Martin-Iradi, and Francesco Corman</i>	16:1–16:6
Integrated Location-Routing-Scheduling for Multi-Trip Mobile Parcel Lockers with Time Windows	
<i>Salma Essouaied, Leandro C. Coelho, Christophe Wilbaut, and Raca Todosijević</i>	17:1–17:6

Preface

Transportation systems are characterized by increasing dynamicity, complexity and integration of multiple phases. The effective design, management and optimization of transportation systems call for advanced mathematical models and efficient algorithms.

The Algorithmic Approaches for Transportation Modelling, Optimization and Systems (ATMOS) symposium has been held every year since 2000 and brings together researchers and practitioners interested in modeling and solving optimization problems arising in transportation systems. The ATMOS 2026 symposium was hosted by the University of L'Aquila on September 3–4 2026 in L'Aquila, Italy, as part of ALGO 2026, the major European event for researchers, students, and practitioners in algorithms.

We received a total of forty-one submissions on optimization problems, models and algorithms related to several optimization phases, ranging from strategic network design, to line planning, timetabling/scheduling and operational and real-time rescheduling. The works tackle problems arising in several transport systems, including railway system, maritime and air transport, electric vehicles and bicycle networks, as well as multi-modal systems. They focus both on theoretical complexity aspects and on solving real-life applications, and propose advanced exact and heuristic approaches and mathematical models to address recent challenging aspects such as passengers' behavior and objectives, timetable stability, train dispatching and routing, improving sustainability and reducing environmental impact, middle and last mile delivery, integration of various transport modes and phases and coordination of complex decisions. Each manuscript was reviewed by at least three PC members who evaluated originality, technical quality, and relevance related to the topics of the symposium. Based on the reviews, the program committee selected sixteen submissions to be presented at the symposium. The manuscripts show the wide applicability of algorithmic optimization to transportation problems.

Two submissions stood out during the review process. The paper “Passenger-Based Inter-modal Connection Optimization with Elastic Demands” by Boris Grimm, Ralf Borndörfer, Andrea Fraioli, Paula Franke, Giovanni Luca Giacco, Federico Marinucci and Tommaso Montino received the “Best Paper Award”, and the “Best Student Paper Award” was given to the paper “Finding Maximum-Success Disjoint Paths” by Aaron Neugebauer and Marie Schmidt.

Our schedule of contributed ATMOS talks was complemented by an ALGO 2026 keynote talk given by Renato Werneck (Amazon, California, USA) titled “Optimizing Last-Mile Deliveries at Amazon”.

We would like to thank: the Steering Committee of ATMOS for giving us the opportunity to serve as Program Chairs of ATMOS 2026; all the authors who submitted their papers; the members of the ATMOS 2026 Program Committee and the sub-reviewers for their valuable work in evaluating all the submissions and selecting the papers appearing in this volume; Renato Werneck for accepting our invitation to present an invited talk; the ALGO 2026 Organizing Committee, in particular Stefano Leucci, for hosting the symposium as part of ALGO 2026. We would also like to acknowledge the use of the EasyChair system for the great help in managing the submission and review processes, and Schloss Dagstuhl for publishing the proceedings of ATMOS 2026 in its OASICS series.

August 2026

Valentina Cacchiani and Stefan Funke

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Committees

Program committee chairs

- Valentina Cacchiani, University of Bologna, Italy
- Stefan Funke, University of Stuttgart, Germany

Program committee members

- Thomas Bläsius, Karlsruhe Institute of Technology, Germany
- David Coudert, INRIA Sophia Antipolis, France
- Mattia D’Emidio, University of L’Aquila, Italy
- Twan Dollevoet, Erasmus University Rotterdam, the Netherlands
- Markus Friedrich, University of Stuttgart, Germany
- Andrew Goldberg, Lehigh University, Bethlehem, USA
- Niels Lindner, Zuse Institute Berlin, Berlin, Germany
- Richard Lusby, Technical University of Denmark, Denmark
- Matúš Mihalák, University of Maastricht, the Netherlands
- Federico Perea, University of Seville, Spain
- Dennis Schieferdecker, Apple, San Francisco, USA
- Philine Schiewe, Aalto University, Finland
- Marie Schmidt, University of Würzburg, Germany
- Sebastian Stiller, Technical University Braunschweig, Germany
- Renato Werneck, Amazon, San Francisco, USA
- Christos Zaroliagis, University of Patras, Greece

Steering committee

- Matthias Müller-Hannemann, University of Halle, Germany
- Marie Schmidt, University of Würzburg, Germany
- Anita Schöbel, University of Kaiserslautern-Landau, Germany
- Christos Zaroliagis, University of Patras, Greece (chair)

Organizing committee

- Davide Bilò, University of L’Aquila
- Gianlorenzo D’Angelo, Gran Sasso Science Institute
- Luciano Gualà, University of Rome “Tor Vergata”
- Stefano Leucci, University of L’Aquila
- Guido Proietti, University of L’Aquila

■ List of Authors

Alexander Beiser  (10)
TU Wien, Vienna, Austria

Ralf Borndörfer  (2, 6)
Freie Universität Berlin, Germany;
Zuse Institute Berlin, Germany

Bruna Cavalcanti Lauro  (9)
Department of Advanced Computing Sciences,
Maastricht University, The Netherlands

Leandro C. Coelho  (17)
CIRRELT and Faculté des Sciences de
l'Administration, Université Laval, Québec,
Canada;
Canada Research Chair in Integrated Logistics,
Université Laval, Québec, Canada

Francesco Corman  (16)
Institute for Transport Planning and Systems,
ETH Zurich, Switzerland

Lena Dittrich  (11, 12)
Department of Mathematics,
RPTU University Kaiserslautern-Landau,
Germany

Thomas Dubach  (4)
IVT Transport Systems Group,
ETH Zurich, Switzerland

Stefan Engels  (3)
Chair for Design Automation,
Technical University of Munich, Germany

Salma Essouaied  (17)
CIRRELT and Faculté des Sciences de
l'Administration, Université Laval, Québec,
Canada;
Univ. Polytechnique Hauts-de-France, LAMIH,
CNRS, UMR 8201, Valenciennes F-59313,
France;
INSA Hauts-de-France, Valenciennes F-59313,
France

Till Fluschnik  (14)
Department of Computer Science, Algorithm
Engineering Group, Humboldt-Universität zu
Berlin, Germany

Florian Flükiger-Fuchs  (4, 16)
IVT Transport Systems Group,
ETH Zurich, Switzerland

Andrea Fraioli (2)
IVU Traffic Technologies Italia S.r.l.,
Rome, Italy

Paula Franke  (2)
Freie Universität Berlin, Germany;
Zuse Institute Berlin, Germany

Giovanni Luca Giacco (2)
Transport Data Management, Direzione
Pianificazione Industriale, Trenitalia SpA,
Rome, Italy

Boris Grimm  (2)
Freie Universität Berlin, Germany; Zuse
Institute Berlin, Germany

Markus Hecher  (10)
University of Potsdam, Potsdam, Germany;
CNRS, Artois University (CRIL), Artois, France

Sven Jäger  (11)
Institute of Mathematics,
Technische Universität Berlin, Germany

Konstantinos Karathanasis  (7)
Department of Computer Engineering and
Informatics, University of Patras, Greece;
PIKEI New Technologies, Patras, Greece

Anastasia Kireeva  (15)
ETH Zurich, Switzerland

Spyros Kontogiannis  (7)
Department of Computer Engineering and
Informatics, University of Patras, Greece;
Computer Technology Institute and Press
"Diophantus", Patras, Greece

Tomas Lidén  (5)
Department of Science and Technology (ITN),
Linköping University, Sweden

Niels Lindner  (6)
Zuse Institute Berlin, Germany;
Freie Universität Berlin, Germany

Philipp Loick  (15)
Amazon Science and Technology, Luxembourg

Federico Marinucci (2)
IVU Traffic Technologies Italia S.r.l.,
Rome, Italy

Bernardo Martin-Iradi  (4, 16)
IVT Transport Systems Group,
ETH Zurich, Switzerland

Matúš Mihalák  (9)
Department of Advanced Computing Sciences,
Maastricht University, The Netherlands

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Tommaso Montino (2)

Transport Data Management, Direzione
Pianificazione Industriale, Trenitalia SpA,
Rome, Italy

Nysret Musliu (10)

TU Wien, Vienna, Austria

Mattia Neroni (15)

Amazon Science and Technology,
Barcelona, Spain

Aaron Neugebauer (8)

Department of Computer Science, University of
Würzburg, Germany;
Image Processing Department, Fraunhofer
ITWM-Institute for Industrial Mathematics,
Kaiserslautern, Germany

Markus Niebisch (13)

Department of Data Analytics and
Digitalisation, School of Business and Economics,
Maastricht University, The Netherlands

Tongcheng Ouyang (16)

Institute for Transport Planning and Systems,
ETH Zurich, Switzerland

Asterios Pegos (7)

Department of Computer Engineering and
Informatics, University of Patras, Greece;
PIKEI New Technologies, Patras, Greece

Michael Rihlmann (12)

Fraunhofer Institute for Industrial Mathematics
ITWM, Kaiserslautern, Germany

Carlo S. Sartori (15)

Amazon Science and Technology, Luxembourg

Philine Schiewe (12)

Department of Mathematics and Systems
Analysis, Aalto University, Finland

Filip Schlembach (9)

Department of Advanced Computing Sciences,
Maastricht University, The Netherlands

Christiane Schmidt (5)

Department of Science and Technology (ITN),
Linköping University, Sweden

Marie Schmidt (8)

Department of Computer Science,
University of Würzburg, Germany

Anita Schöbel (11, 12)

Department of Mathematics, RPTU University
Kaiserslautern-Landau, Germany;
Fraunhofer Institute for Industrial Mathematics
ITWM, Kaiserslautern, Germany

Jian Shen (15)

Amazon Science and Technology, London, UK

Evgueni Smirnov (9)

Department of Advanced Computing Sciences,
Maastricht University, The Netherlands

Vasileios Sofianos (7)

Department of Computer Engineering and
Informatics, University of Patras, Greece

Luka Stärk (6)

Zuse Institute Berlin, Germany;
Freie Universität Berlin, Germany

Raca Todosijević (17)

Univ. Polytechnique Hauts-de-France, LAMIH,
CNRS, UMR 8201, Valenciennes F-59313,
France;
INSA Hauts-de-France, Valenciennes F-59313,
France

Georg Trausmuth (10)

Frequentis AG, Vienna, Austria

Tom C. van der Zanden (13)

Department of Data Analytics and
Digitalisation, School of Business and Economics,
Maastricht University, The Netherlands

Renato F. Werneck (1)

Amazon, San Francisco, CA, USA

Christophe Wilbaut (17)

Univ. Polytechnique Hauts-de-France, LAMIH,
CNRS, UMR 8201, Valenciennes F-59313,
France;
INSA Hauts-de-France, Valenciennes F-59313,
France

Robert Wille (3)

Chair for Design Automation,
Technical University of Munich, Germany

Stefan Woltran (10)

TU Wien, Vienna, Austria

Rabii Zahir (5)

Department of Science and Technology (ITN),
Linköping University, Sweden

Christos Zaroliagis (7)

Department of Computer Engineering and
Informatics, University of Patras, Greece;
Computer Technology Institute and Press
“Diophantus”, Patras, Greece

Optimizing Last-Mile Deliveries at Amazon

Renato F. Werneck  

Amazon, San Francisco, CA, USA

Abstract

Amazon delivers tens of millions of shipments worldwide on a daily basis. Planning the last-mile routes that reach final customers requires solving combinatorial optimization problems at massive scale. While designing and engineering efficient algorithms is essential to address this challenge, it is far from sufficient. Drawing on practical examples, this talk argues that properly framing the problem, choosing the right metrics, and navigating real-world operational and organizational constraints are just as critical. In fact, they ultimately shape the algorithms themselves.

2012 ACM Subject Classification Applied computing → Transportation

Keywords and phrases Vehicle Routing, Route Planning, Last-Mile Logistics, Algorithm Engineering

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.1

Category Invited Talk



© Renato F. Werneck;

licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 1; pp. 1:1–1:1



OpenAccess Series in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Passenger-Based Intermodal Connection Optimization with Elastic Demands

Boris Grimm ✉ 

Freie Universität Berlin, Germany
Zuse Institute Berlin, Germany

Ralf Borndörfer ✉ 

Freie Universität Berlin, Germany
Zuse Institute Berlin, Germany

Andrea Fraioli

IVU Traffic Technologies Italia S.r.l., Rome, Italy

Paula Franke ✉ 

Freie Universität Berlin, Germany
Zuse Institute Berlin, Germany

Giovanni Luca Giacco

Transport Data Management, Direzione Pianificazione Industriale, Trenitalia SpA, Rome, Italy

Federico Marinucci

IVU Traffic Technologies Italia S.r.l., Rome, Italy

Tommaso Montino

Transport Data Management, Direzione Pianificazione Industriale, Trenitalia SpA, Rome, Italy

Abstract

A railway network can attract additional passengers by improving the quality of service. One way to do that is to establish new train connections by slight shifts in arrival and departure times. We propose a connection optimization model to find a shifting of a given aperiodic timetable that maximizes the passenger volume, in which the demand is elastic and forecast by an integrated discrete choice logit model. The resulting large-scale mixed-integer non-linear non-convex programming problems can be solved by tailor-made heuristics for the entire Italian railway network with its 2,000 stations and 10,000 daily train trips, uncovering substantial potentials for connectivity improvements and boosts in demand.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Mathematics of computing → Network optimization; Applied computing → Transportation

Keywords and phrases Timetabling, Demand Elasticity, Discrete Choice, Mixed-Integer Nonlinear Programming, Heuristics

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.2

Funding The work for this article has been conducted in the MobilityLab of the Research Campus MODAL funded by the German Federal Ministry of Research, Technology and Space (BMFTR), fund number 05M25KEM. The work of authors 1, 2, and 4 is also supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/2, project ID: 390685689).

Acknowledgements We thank three anonymous reviewers for their dedication and time to review this paper. Their fruitful comments improved our work.



© Boris Grimm, Ralf Borndörfer, Andrea Fraioli, Paula Franke, Giovanni Luca Giacco, Federico Marinucci, and Tommaso Montino;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 2; pp. 2:1–2:18



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The attractiveness of railway transportation to passengers, in comparison to other means of transport, depends on multiple factors such as availability, reliability, ticket price, travel duration, number of transfers, and transfer times. Railway companies are particularly interested in establishing opportunities for transfers, as they allow passengers to travel between origins and destinations that are not connected by direct lines; such connections generate additional business opportunities and generally improve the quality of service. It is therefore highly interesting to predict and ultimately optimize potential increases in passenger demand in dependence of the train connections, which is precisely the aim of connection optimization.

From an operational perspective, transfers are subject to a number of constraints. Indeed, a feasible transfer requires a minimum duration between the arrival of the first train and the departure of the second train, because otherwise it is not possible to change from the first train to the second. On the other hand, long transfer waiting times discourage passengers from traveling by train, such that an upper bound on the transfer time is also appropriate. There is hence a certain time window for a feasible transfer. This time window can in many cases be met by relatively small shifts in the arrival and departure times of the respective trains at the transfer station, and at essentially no additional operational cost to the service provider.

Giacco et al. [10] proposed a connection optimization approach in order to establish more and/or better transfer opportunities for a major Italian railway company. They provide a case study involving roughly 10,000 scheduled train trips between 2,225 stations for an entire day of operation on a nationwide scale. The increase in the number of feasible connections is significant: When allowing only small shifts in departure and arrival times of ± 1 minute, the number of connections can be increased by 5%. When allowing up to ± 8 minute shifts, the number of connections rises by up to 15%. However, a connection is only a transfer opportunity, and some connections are surely more valuable than others. Hence, the question arises of how much of this potential translates into an increase of railway passenger transportation volume, which is the overall goal of every railway operator. An assessment of this potential requires a prediction of the travel choices of potential passengers, which may or may not use the train depending on whether they have a (good) connection or not. Discrete choice logit models are well established to make such predictions. The aim of this paper is to integrate such a logit model into the connection optimization model of Giacco et al. in order to control the elasticity of the demand with respect to the connections, and to optimize the connections such that the number of railway passengers increases.

Previous work

Connection optimization changes the timetable by shifting arrival and departure times; the problem is therefore a timetabling problem and, in particular, a railway timetabling problem. Railway timetabling is one of the central planning problems in railway operations, as timetable quality directly affects infrastructure utilization, operational efficiency, and passenger satisfaction. It has been studied extensively in the Operations Research literature, starting with Szpigel's 1973 paper on railway timetabling on a single-track corridor [24]. Subsequent research extended these optimization approaches from single-track corridors to increasingly complex railway networks. As is highlighted in the survey articles by Caimi, Kroon & Liebchen [6] and Cacchiani & Toth [5], the two most common approaches to timetabling are periodic timetabling, i.e., creating timetables that repeat after a fixed time

interval, and aperiodic timetabling. Periodic timetabling allows for foreseeable planning under predictive circumstances for rolling stock deployment and maintenance scheduling, see, e.g., Peeters [19] or Liebchen [16]. Aperiodic timetables are mostly considered in the context of shorter time periods with unusual circumstances such as construction zones or holidays, or in connection with train routing, see e.g., the work of Caprara, Fischetti & Toth [8] on the train timetabling problem, of Borndörfer et al. [3] on railway track allocation, or the survey article by Lusby et al. [17].

Aperiodic timetables also arise from modifying a given (periodic or aperiodic) timetable by shifting departure and arrival times in order to improve some measure of connectivity; this idea has been explored repeatedly under terms such as timetable synchronization, timetable coordination, schedule optimization etc., sometimes in an intermodal context. For example, Ceder et al. [9], synchronize bus timetables to maximize the number of simultaneous arrivals, Wong et al. [27] synchronize the Hong Kong mass transit rail timetable to minimize the total transfer waiting time, Cao et al. [7] synchronize the Beijing subway timetable, Ke et al. [14, 15] coordinate the high-speed rail timetable on the corridor from Zhengzhou to Beijing to establish additional connections to and from flights, while Polinder, Schmidt and Huisman [20] show that slight deviations from periodic timetables can enable better passenger travel times.

Substantial progress has been made in the treatment of passenger behavior. Many timetabling models consider a fixed passenger routing and aim to optimize the timetable based on this travel behavior [6]. However, given the importance of the timetable for passenger travel decisions, researchers have started to investigate flexible passenger routing of a static demand in response to timetabling [4], [22]. Indeed, fixing passenger routes before timetable optimization may lead to arbitrarily bad results compared to approaches that integrate passenger routing into the timetable optimization [4]. Integrated design approaches for aperiodic timetabling were presented by Schmidt [21], and Schöbel and Scholl [23]. A notable approach is the one of Hartleb and Schmidt ([12]), who incorporated an advanced treatment of passenger routing that includes mode choice. Under the assumption that the travel choice depends on the number of available routes, they approximate a logit distribution of the passenger flow, uncovering improvement potentials over alternative timetabling methods. Indeed, logit models [18, 1, 25] are widely accepted as a reasonable way to anticipate travel choices, and timetabling models that integrate a logit routing can be considered as being particularly realistic.

The above approaches optimize the construction of a timetable while considering passenger behavior. Another line of research studies how already existing timetables can be modified to improve passenger service. This constitutes a switch from a purely operation-driven timetabling approach to passenger-centric scheduling without having to completely change the timetable and thus maintaining reliability for passengers in the process. We mention Binder et al. [2], who proposed a multi-objective algorithm for timetable rescheduling after a disruption, allowing for a consideration of both passenger and operational objectives.

This work

While the literature extensively covers timetabling and integrated models that include passenger response to the timetable, there is little research on refining existing timetables subject to a realistic demand response. Integrated models so far mostly consider changes in route choices subject to a fixed travel demand. The question of whether passengers want to travel at all has been rarely studied, and, as far as we know, not in the context of connection optimization.

We address the two points of connection optimization and demand elasticity by developing an integrated scheduling model that is based on an existing timetable, optimizes the connections, and predicts the associated demand elasticity. We employ a logit approach to model passenger behavior. Different from Hartleb and Schmidt [12], we do not discretize subject to the number of travel alternatives, but use an iterative flow distribution approach that alternates between connection optimization and passenger routing. This method works for large-scale scenarios and is exemplified in a case study for a European passenger railway network of 2,225 stations and a timetable for roughly 10,000 trips. The paper is organized as follows. Section 2 presents a mathematical definition of the problem as well as the development of the associated logit connection optimization model. An iterative flow distribution approach to tackle this problem is presented in Section 3. Section 4 reports and discusses computational results for a large-scale application. Section 5 concludes.

2 Connection Optimization

We discuss the connection optimization problem in three versions of increasing complexity that highlight the major characteristics. The first version, the connection optimization problem (COP), just counts connections; they are the decisive combinatorial part of the problem and decide which journeys are possible and which ones are not. The second version, the passenger connection optimization problem (PCOP), adds a passenger flow to maximize the volume of the satisfied demand; this model is more realistic and the workhorse of this paper. The third version, the logit(-based) connection optimization problem (LCOP), incorporates a discrete choice logit model to address the elasticity of the demand with respect to the service quality; it aims at attracting car drivers to use the train. The notation is summarized in Tab. 5 in the appendix.

2.1 The Connection Optimization Problem

Giacco et al. [10] considered the following basic version of the connection optimization problem. Let $G = (S, A)$ be a directed graph of railway stations S and (train) trips A . The trips can be partitioned into a set of directed paths I which represent trains. Each train $i \in I$ hence does a sequence of trips, and in doing so, visits a set of stations S_i , arriving at and departing from station $s \in S_i$ at default times $\bar{a}_i^s \in \mathbb{N}_0$ and $\bar{d}_i^s \in \mathbb{N}_0$, respectively. These times can be adjusted by timetable shifts $Inf \leq t_i \leq Sup$, resulting in shifted departure and arrival times $a_i^s = \bar{a}_i^s + t_i$ and $d_i^s = \bar{d}_i^s + t_i$, respectively; here $Inf, Sup \in \mathbb{Z}$ are global bounds for the minimum and maximum time shifts and typically $Inf = -Sup$. Let $C \subseteq S \times I \times I$ be a set of potential connections between the trains. A connection between trains i and j at station s is feasible iff

$$m_{ij}^s \leq \Delta_{ij}^s := d_j^s - a_i^s := \bar{d}_j^s - \bar{a}_i^s + t_j - t_i \leq m_{ij}^s + \beta$$

holds, where $m_{ij}^s \in \mathbb{N}_{\geq 0}$ is a minimum connection time between trains i and j at station s and $\beta \in \mathbb{N}_{\geq 0}$ is the width of the connection time window. The minimum connection time m_{ij}^s is mostly dependent on the station's topology, i.e., the distance of the involved platforms, while β is considered to be a parameter of the problem. The *Connection Optimization Problem* (COP) is to determine timetable shifts that maximize the number of feasible connections. This is important in sparse railway networks, where the quality of the connections decides on the passenger volume. If transfers are too short, the passengers cannot reach them, and if they are too long, then they effectively cease to exist, because people will not use them. The

combinatorial structure of high-quality connections therefore determines the connectivity of the network, and the connections can be influenced in a significant way by retiming the arrivals and departures.

The Connection Optimization Problem can be formulated as a mixed-integer linear programming problem in terms of timetable shift variables t_i , connection time variables Δ_{ij}^s , and binary connection variables x_{ij}^s as follows:

$$\max \sum x_{ij}^s, \quad (\text{COP})$$

$$\text{s.t. } \Delta_{ij}^s \geq \text{Inf}_{ij}^s - (\text{Inf}_{ij}^s - m_{ij}^s) x_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (1)$$

$$\Delta_{ij}^s \leq \text{Sup}_{ij}^s - (\text{Sup}_{ij}^s - m_{ij}^s - \beta) x_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (2)$$

$$\Delta_{ij}^s = \bar{d}_j^s + t_j - \bar{a}_i^s - t_i \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (3)$$

$$\text{Inf} \leq t_i \leq \text{Sup} \quad \forall i \in I, \quad (4)$$

$$x_{ij}^s \in \{0, 1\} \quad \forall i, j \in I, s \in S_i \cap S_j. \quad (5)$$

The objective of (COP) maximizes the number of feasible connections. Constraints (1) and (2) determine the feasibility of a connection; here, $\text{Inf}_{ij}^s := \bar{d}_j^s - \text{Inf} - \bar{a}_i^s + \text{Sup}$ is a lower and $\text{Sup}_{ij}^s := \bar{d}_j^s + \text{Sup} - \bar{a}_i^s - \text{Inf}$ an upper bound on the transfer time from train i to train j at station s . The equation (3) computes the transfer time, while (4) limits the time shifts. Constraint (5) sets the domain of the connection variables.

Note that the connection time variables can be eliminated. In fact, Giacco et al. [10] used the constraints

$$1 + \frac{\Delta_{ij}^s - m_{ij}^s}{1440} \geq x_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (6)$$

$$1 + \frac{m_{ij}^s + \beta - \Delta_{ij}^s}{1440} \geq x_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (7)$$

instead of COP (1-3), assuming a time resolution of one minute and a time horizon of one day or 1440 minutes; these are integer equivalent, but weaker from an LP relaxation point of view. There is furthermore no need to stipulate the integrality of the time shifts, because there is always an optimal integer solution if all input data is integer (as we required). We remark that one can replace the uniform shifts t_i by more flexible station-dependent shifts t_{is} , which can be increasing, or coupled, etc., but there is no space to discuss such versions here. We also do not consider headway times for pairs of preceding and succeeding trains sharing a track segment, but associated constraints can in principle be added to the model; for example, Ke et al. [14, 15] considered this degree of freedom.

The COP is theoretically difficult, but computationally tractable for the instances that we target.

► **Theorem 1.** *The Connection Optimization Problem is strongly NP-hard.*

Proof. A reduction of the Max-Cut Problem to the Passenger Connection Optimization Problem, in which all parameters are 0/1, can be found in [13]. ◀

2.2 The Passenger Connection Optimization Problem

A problem of the basic model is that it ignores the importance of the connections for the passengers, as one connection is as good as another. This issue could be mitigated by introducing connection weights, but a more satisfying solution is to compute the actual

passenger flow in order to maximize the satisfied demand, which is the overall purpose of improving the connections. In this vein, we assume that there is a maximal demand given by an od -cube $D \in \mathbb{Q}_{\geq 0}^{S \times S \times H}$ over a discrete (hourly) time horizon H (of one day). For each triple of origin and destination stations $o, d \in S$ and hour h we consider a set P_{odh}^r of railway paths that connect o and d along time-feasible trips and connections between these trips; the set $P^r = \bigcup P_{odh}^r$ of all these paths represents the services that the railway offers. A railway path p is feasible with respect to a timetable shifting if all of its connections are feasible. For a path $p \in P_{odh}^r$ let $y_p \in [0, 1]$ be the fraction of the demand D_{odh} routed on p , and let $\kappa_a \in \mathbb{Q}_{\geq 0}$ be the maximum capacity of trip $a \in A$. A passenger flow is feasible if it ships some or all of the demand on feasible railway paths and in such a way that the capacities on all trips are respected. The *Passenger Connection Optimization Problem* (PCOP) is to determine time shifts such that a weighted sum of the satisfied demand and the number of feasible connections is maximized.

Extending the basic IP model by passenger flow variables y_p , the PCOP can be formulated as a mixed-integer program as follows:

$$\begin{aligned}
 \max \quad & \lambda \sum_{(i,j,s) \in C} x_{ij}^s + \mu \sum_{odh \in S^2 \times H} \sum_{p \in P_{odh}^r} D_{odh} y_p & (\text{PCOP}) \\
 \text{s.t.} \quad & \Delta_{ij}^s + (Inf_{ij}^s - m_{ij}^s) x_{ij}^s \geq Inf_{ij}^s & \forall i, j \in I, s \in S_i \cap S_j, & (8) \\
 & \Delta_{ij}^s + (Sup_{ij}^s - m_{ij}^s - \beta) x_{ij}^s \leq Sup_{ij}^s & \forall i, j \in I, s \in S_i \cap S_j, & (9) \\
 & \Delta_{ij}^s = \bar{d}_j^s + t_j - \bar{a}_i^s - t_j & \forall i, j \in I, s \in S_i \cap S_j, & (10) \\
 & Inf \leq t_i \leq Sup & \forall i \in I, & (11) \\
 & x_{ij}^s \in \{0, 1\} & \forall i, j \in I, s \in S_i \cap S_j, & (12) \\
 & \sum_{p \in P_{odh}^r} D_{odh} y_p \leq D_{odh} & \forall o, d \in Z, o \neq d, h \in H, & (13) \\
 & \sum_{a \in p \in P^r} D_{odh} y_p \leq \kappa_a & \forall a \in A, & (14) \\
 & 0 \leq y_p \leq x_{ij}^s & \forall p \in P^r, (i, j, s) \in p, & (15) \\
 & 0 \leq y_p \leq 1 & \forall p \in P^r. & (16)
 \end{aligned}$$

The objective of PCOP is a weighted sum of the number of connections and the satisfied demand subject to weights $\lambda, \mu \geq 0$. Constraints (8-12) are the same as in COP. Constraints (13-14) define the capacitated passenger flow, while 15 makes sure that only feasible paths are used. The demand constraints (13) could be normalized, and the upper bounds on the passenger flow (16) are redundant, but both constraints will play a role later with different right-hand sides. The PCOP is an integrated connection optimization and passenger routing problem, where the passengers are routed in a system-optimal way as a multicommodity flow over a given set of railway paths. It is clearly more difficult than the COP, but computationally tractable for large-scale instances and reasonable numbers of paths per odh -triple.

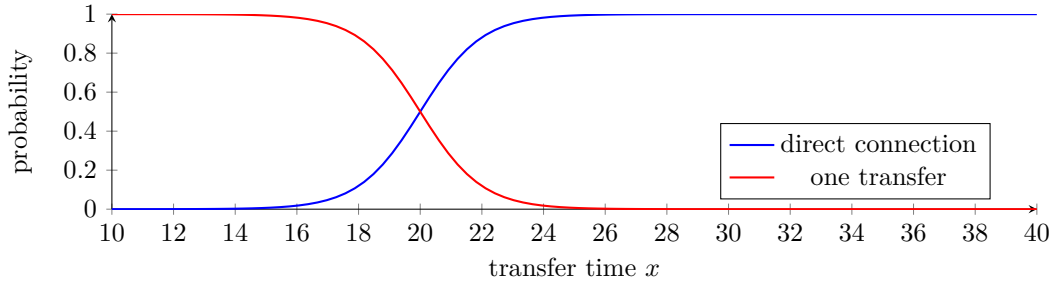
► **Theorem 2.** *The Passenger Connection Optimization Problem is strongly NP-hard.*

Proof. A reduction of the 3-SAT Problem to the Passenger Connection Optimization Problem, in which all parameters are polynomial, can be found in [13]. ◀

2.3 The Logit Connection Optimization Problem

The main drawback of the passenger connection optimization approach is that it computes the passenger flow as a system optimum, while travelers in reality make independent and selfish choices. They may, in particular, choose not to travel at all, or by another mode

such as the car; they may also switch from car to rail due to attractive connections. A widely accepted method to model this behavior in a more realistic way are discrete choice models [18, 1, 25]. These associate a utility with every travel alternative, which reflects its overall attractiveness, and stipulate that a high utility results in a high probability of choice, which is assumed to be feasible. This is a problem in the presence of constraints on the supply, such as capacity constraints. Various options are available to deal with this issue, such as modifying the utility by congestion terms, maximizing the feasible utility, or reallocating excess demand to secondary choices. We will use the latter approach by constructing a capacity-constrained passenger flow that is close to the passenger distribution arising from a logit model. This reflects the behavior of travelers who initially strive for the best trains, and resort to secondary alternatives when these are overcrowded. Logit models often consider zone-based demands, which make the selection of the start and end station a degree of freedom. This is important in regional rail traffic, and simultaneously improves and coarsens the demand data, so we do that, and we will consider the car as an additional mode and source of new railway customers.



■ **Figure 1** Probabilities of two alternative travel choices depending on a transfer time x according to a logit model in which the direct connection has observable utility -150 , the connection with one transfer has observable utility $-130 - x$, and the random variation is Gumbel $G(0, 1)$ distributed.

Let Z be a partition of the set of stations S into zones $z \in Z, z \subseteq S$, let $D \in \mathbb{Q}^{Z \times Z \times H}$ be the associated zone-based demand cube, and let P_{odh}^c and P_{odh}^r denote a set of car and railway paths, respectively, between origin zone o and destination zone d that start in hour h (between h and $h + 1$). Denote by $P_{odh} = P_{odh}^c \cup P_{odh}^r$ the set of all paths connecting the odh -triple odh , and by $P := P^c \cup P^r$ the set of all passenger paths. The paths are evaluated by a utility function $u : P \times \mathbb{Z}^I \rightarrow \mathbb{R}$ that associates with each path and time shift a value that represents the quality of this path for the passengers. A(n unnested) logit model assumes that the percentage passenger flow on a path $p \in P_{odh}$ and a timetable shifting t , i.e., the fraction of the demand D_{odh} routed on p , is

$$\bar{y}_p = \frac{e^{u_p(t)}}{\sum_{p \in P_{odh}} e^{u_p(t)}}. \quad (17)$$

The logit(-induced passenger) flow $\bar{y}$ depends in a non-linear and non-convex way on the shifting of the timetable, see Fig. 1. However, this dependence is relatively small, and does by itself not lead to a substantial flow redistribution. The main impact of the shifting is still the effect on the feasibility of connections, as in the basic model, and this in turn redistributes the logit flow. The *Logit Connection Optimization Problem* (LCOP) is the task to find a shifting for the train timetable that maximizes the number of passengers taking the train

subject to a passenger flow that differs as little as possible from the logit flow. One way to make this goal precise is to formulate the LCOP as a mixed-integer non-linear program adding variables $\bar{y}_p$ for the logit flow and s_p for a path-wise excess over the logit flow:

$$\max \lambda \sum_{(i,j,s) \in C} x_{ij}^s + \mu \sum_{odh \in Z^2 \times H} \sum_{p \in P_{odh}^r} D_{odh} y_p - \nu \sum_{p \in P^r} s_p \quad (\text{LCOP})$$

$$\text{s.t. } \Delta_{ij}^s + (Inf_{ij}^s - m_{ij}^s) x_{ij}^s \geq Inf_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (18)$$

$$\Delta_{ij}^s + (Sup_{ij}^s - m_{ij}^s - \beta) x_{ij}^s \leq Sup_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (19)$$

$$\Delta_{ij}^s = \bar{d}_j^s + t_j - \bar{a}_i^s - t_j \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (20)$$

$$Inf \leq t_i \leq Sup \quad \forall i \in I, \quad (21)$$

$$x_{ij}^s \in \{0, 1\} \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (22)$$

$$\sum_{\substack{p \in P_{odh}^r \\ u_p(t) \leq u(q,t)}} D_{odh} y_p \leq \frac{\sum_{\substack{p \in P_{odh}^r \\ u_p(t) \leq u(q,t)}} e^{u_p(t)}}{\sum_{p \in P_{odh}^r} e^{u_p(t)}} D_{odh} \quad \forall o \neq d \in Z, h \in H, q \in P_{odh}^r, \quad (23)$$

$$\sum_{a \in p \in P^r} D_{odh} y_p \leq \kappa_a \quad \forall a \in A, \quad (24)$$

$$0 \leq y_p \leq x_{ij}^s \quad \forall p \in P^r, (i, j, s) \in p, \quad (25)$$

$$0 \leq y_p \leq \bar{y}_p + s_p \quad \forall p \in P^r, \quad (26)$$

$$\bar{y}_p = \frac{e^{u_p(t)}}{\sum_{p \in P_{odh}^r} e^{u_p(t)}} \quad \forall p \in P^r, \quad (27)$$

$$0 \leq s_p \quad \forall p \in P^r. \quad (28)$$

The first two terms of the objective and the constraints (18-22) and (24-25) are as in the PCOP. The new demand constraints (23) limit the railway passenger flow for each odh -triple to the railway share of the modal split according to the logit model; this share is computed for all levels of utilities in order to enforce a similarity to the logit flow. Constraint (27) computes the logit flow, while (26) and (28) capture the individual deviations from the logit flow for each railway path; here, the constraints (26) replace the upper bounds on the passenger flow in the PCOP. The sum of the deviations is penalized in the new third term of the objective, which is weighed by a factor $\nu \geq 0$. We remark that the LCOP is certainly more complex than its predecessors, but that a formal analysis would require a definition that avoids the computation of irrational numbers.

It is known that a logit flow can also be computed by maximizing the associated entropy or, alternatively, minimizing the perceived travel time, which are concave resp. convex functions, see, e.g., van Lieshout & Dalmeijer [26]. One could therefore alternatively replace the demand constraints (23) and (27) by such an objective, resulting in a MIP with a concave objective. Such a model is usually treated by outer approximation, which in this case will produce cutting planes with the same purpose as our demand constraints. This approach is technically more demanding and not the subject of this paper.

3 Solving the Logit Connection Optimization Problem

LCOP is a mixed-integer non-linear non-convex programming problem of high algebraic complexity. For the scenarios with 2,000 stations and 10,000 trains that we have in mind, it is also very large. As an exact solution appears extremely difficult, if not out of reach, we

propose two heuristic approaches that approximate the LCOP in terms of PCOP problems. The first, which we call mode-based flow distribution, solves a single PCOP to distribute an “idealized logit flow”. The second tries to get closer to the logit distribution by additionally minimizing the total path deviation.

3.1 Mode-Based Flow Distribution

The mode-based flow distribution method attempts to solve the relaxation of the LCOP that arises from deleting the path-based deviation part in the objective as well as the associated constraints (26-28), and from replacing the right-hand sides of the demand constraints (23) by best possible shares. These shares are computed in terms of best possible utilities for the railway paths, which overestimates the demand for rail transport. This produces a high railway passenger volume, which is useful in order to identify timetable optimization potentials. Let $\bar{u}_p$ and $\underline{u}_p$ be the maximal and minimal utility of a passenger path p , i.e., the utility for a (hypothetical) timetable shifting that always produces minimal and maximal connection times, respectively, and let

$$\bar{U}_q := \frac{\sum_{p \in P_{odh}^r : \underline{u}_p \leq \bar{u}_q} e^{\bar{u}_p}}{\sum_{p \in P_{odh}} e^{\underline{u}_p}} \quad (29)$$

be an idealized maximal railway share of the modal-split for the best possible utility level of path q . We remark that in our application $\underline{u}$ and $\bar{u}$ are not as far apart as one might fear. The mode-based approach tries to distribute a flow with these idealized shares according to the following mixed-integer program:

$$\max \lambda \sum_{(i,j,s) \in C} x_{ij}^s + \mu \sum_{odh \in Z^2 \times H} \sum_{p \in P_{odh}^r} D_{odh} y_p \quad (\text{MCOP})$$

$$\text{s.t. } \Delta_{ij}^s + (\text{Inf}_{ij}^s - m_{ij}^s) x_{ij}^s \geq \text{Inf}_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (30)$$

$$\Delta_{ij}^s + (\text{Sup}_{ij}^s - m_{ij}^s - \beta) x_{ij}^s \leq \text{Sup}_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (31)$$

$$\Delta_{ij}^s = \bar{d}_j^s + t_j - \bar{a}_i^s - t_j \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (32)$$

$$\text{Inf} \leq t_i \leq \text{Sup} \quad \forall i \in I, \quad (33)$$

$$x_{ij}^s \in \{0, 1\} \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (34)$$

$$\sum_{\substack{p \in P_{odh}^r \\ \underline{u}_p(t) \leq \bar{u}_q(t)}} D_{odh} y_p \leq \bar{U}_q D_{odh} \quad \forall o \neq d \in Z, h \in H, q \in P_{odh}^r, \quad (35)$$

$$\sum_{a \in p \in P^r} D_{odh} y_p \leq \kappa_a \quad \forall a \in A, \quad (36)$$

$$0 \leq y_p \leq x_{ij}^s \quad \forall p \in P^r, (i, j, s) \in p. \quad (37)$$

This MIP is a PCOP with a lot of demand constraints, namely, for say k paths per odh -triple, k such constraints instead of one in the standard PCOP. For reasonable values of k , this is still tractable.

► **Theorem 3.** *MCOP is a relaxation of LCOP and hence provides an upper bound.*

Proof.

$$\bar{U}_q = \frac{\sum_{p \in P_{odh}^r : \underline{u}_p \leq \bar{u}_q} e^{\bar{u}_p}}{\sum_{p \in P_{odh}} e^{\underline{u}_p}} \geq \frac{\sum_{p \in P_{odh}^r : \underline{u}_p(t) \leq \bar{u}_q(t)} e^{\underline{u}_p(t)}}{\sum_{p \in P_{odh}} e^{\underline{u}_p(t)}} \quad \forall p \in P^r, t \in [\text{Inf}, \text{Sup}]^I. \quad (38)$$

◀

3.2 Iterative Flow Distribution

One might expect that the many demand constraints of the mode-based approach guarantee a good fit with the logit flow not only on a utility-level basis, but also for each individual path. This is, however, not the case, because the logit flow needs to be substantially rerouted in order to avoid congested trains, and this destroys the similarity. The iterative flow distribution method attempts to improve the fit on a path basis by restricting the deviation. We do that by means of upper bounds $\bar{y}_p$ on the path flow, which are initialized with the logit flow y_p . Recall that the logit model assigns too much flow to attractive trains, more than these trains can transport, such that the logit upper bounds for such paths will be inactive for a capacitated passenger flow. The excess logit flow can only be rerouted along other paths, which then have to transport more than the logit flow, but they can't, because their upper bound constraints coincide with the logit flow and are hence tight. We must therefore raise the active upper bound constraints in order to provide capacity for the excess flow, and do so in a uniform way. This allows to reroute some excess flow, but normally not all of it. We therefore raise the active bounds again and repeat until either all excess flow is routed, or all upper bounds become inactive. Denoting the upper bounds by $\bar{y}_p$, the following mixed-integer programming problem is solved in every iteration:

$$\max \lambda \sum_{(i,j,s) \in C} x_{ij}^s + \mu \sum_{odh \in Z^2 \times H} \sum_{p \in P_{odh}^r} D_{odh} y_p \quad (\text{ICOP}(\bar{y}))$$

$$\text{s.t. } \Delta_{ij}^s + (\text{Inf}_{ij}^s - m_{ij}^s) x_{ij}^s \geq \text{Inf}_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (39)$$

$$\Delta_{ij}^s + (\text{Sup}_{ij}^s - m_{ij}^s - \beta) x_{ij}^s \leq \text{Sup}_{ij}^s \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (40)$$

$$\Delta_{ij}^s = \bar{d}_j^s + t_j - \bar{a}_i^s - t_j \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (41)$$

$$\text{Inf} \leq t_i \leq \text{Sup} \quad \forall i \in I, \quad (42)$$

$$x_{ij}^s \in \{0, 1\} \quad \forall i, j \in I, s \in S_i \cap S_j, \quad (43)$$

$$\sum_{\substack{p \in P_{odh}^r \\ u_p(t) \leq u(q,t)}} D_{odh} y_p \leq \bar{U}_q D_{odh} \quad \forall o \neq d \in Z, h \in H, q \in P_{odh}^r \quad (44)$$

$$\sum_{a \in p \in P^r} D_{odh} y_p \leq \kappa_a \quad \forall a \in A, \quad (45)$$

$$0 \leq y_p \leq x_{ij}^s \quad \forall p \in P^r, (i, j, s) \in p, \quad (46)$$

$$0 \leq y_p \leq \bar{y}_p \quad \forall p \in P^r. \quad (47)$$

The update of the bounds is documented in Alg. 1, which raises the bounds uniformly and only so much as to produce a theoretical chance to reroute all of the excess flow. As the bounds only increase, and the number of paths is finite, this method stops after a finite number of iterations. It aims at distributing all the logit flow, keeping the deviation as uniform as possible. The method is also satisfactory from a behavioral point of view, as it mimics the experiences of travellers who run into problems trying to buy tickets for overcrowded trains, or trying to board them in case of time-card owners, and then must change to secondary alternatives, which might again be overcrowded, and so on. ICOP($\bar{y}$) adds the upper bound constraints (47) to the MCOP and is also a PCOP problem. As such, it is computationally tractable for our application.

Algorithm 1 Iterative Flow Distribution.

Input: (LCOP)**Output:** feasible timetable shifting t , connections x , passenger flow y

```

1:  $(t, x, y) \leftarrow \text{argmin (MCOP)}$   $\triangleright$  compute initial timetable shifting and passenger flow
2:  $\bar{y} \leftarrow \bar{y}(t)$   $\triangleright$  compute logit flow associated with this shifting
3:  $\epsilon = \sum_{p \in P^r} \bar{y}_p - \sum_{p \in P^r} y_p$   $\triangleright$  compute total excess flow
4:  $Q \leftarrow \{p \in P^r : y_p = \bar{y}_p\}$   $\triangleright$  get paths with tight upper bounds
5: if  $\epsilon = 0$  or  $Q = \emptyset$  then  $\triangleright$  everything routed?
6:   return  $(t, x, y)$ 
7: end if
8:  $\bar{y}_p \leftarrow \bar{y}_p + \epsilon/|Q|$  for all  $p \in Q$   $\triangleright$  raise active bounds uniformly to route excess flow
9:  $(t, x, y) \leftarrow \text{argmin (ICOP)}$   $\triangleright$  resolve
10: goto 3  $\triangleright$  iterate

```

4 Computational Experiments

We evaluated our solution approach for the LCOP on a real-world timetable of Italy. It contains roughly 10,000 trips, performing 115,000 stops at 2,225 stations. The demand data originates from mobile phone data on (long-distance) public transport, car, and flight movements. It was then aggregated and assigned to origin zones, destination zones, and starting hours. In this way, 800 zones with an hourly granularity over a single day come up. We decided to neglect the air demand as the overall number of travelers was very small, and air travel mostly connected islands with each other and the mainland of Italy. For each of the remaining *odh*-triples with a demand of at least one passenger and each possible shifting interval, a set of at most 5 paths was computed in the railway graph that contains all connections that are potentially feasible with respect to the chosen shifting interval. The six resulting path sets were merged into a single one, such that each *odh*-triple had at most 30 paths to choose from. This set was considered in every run. The path set was constructed in such a way that it is possible to route the railway demand given from the mobile phone data in the unshifted static case, i.e., without additional demand drawn from other modes of transport. This path choice plays a crucial role for the optimization process. Tab. 1 reports the instance characteristics in dependence of a shifting interval. The row headlined *Given paths* shows the total number of paths in the scenario, i.e., ≈ 2.7 million paths. These paths were filtered by checking whether the required connections can be feasible within the shifting window of the instance. This leads to ≈ 1.56 million *Considered paths* for a shifting window of $[0, 0]$ minutes up to ≈ 2.735 million paths for a shifting window of $[-5, 5]$ minutes. The last row shows the number of *odh-triples with demand* larger than a given threshold, namely, in all our instances, a value of 1, which leads to 376,000 considered *odh-triples*. In all our computations, a connection time window of $\beta = 30$ minutes was considered in all runs, and we changed the objective function slightly by scaling the demand induced by an *odh*-triple by the distance in kilometers between the two zones. This makes longer train connections more important than short ones.

In the logit based models, we assume that the observable utility ν_p of a railway path depends affinely on the travel time in high-speed (ht) and low-speed travel time (lt), the number of connections (nc), the waiting time between two trips (ct), access (a) and egress (e) times, the ticket price (tp), and the usage of high-speed trains (hu); these were the attributes for which data was available. Let $b_a(d(o, d))$ be coefficients for each attribute from

■ **Table 1** Demand and path characteristics for all considered experiments.

	[-0,0]	[-1,1]	[-2,2]	[-3,3]	[-4,4]	[-5,5]
Given paths ($\cdot 10^6$)	2.735	2.735	2.735	2.735	2.735	2.735
Considered paths ($\cdot 10^6$)	1.556	1.828	2.125	2.360	2.567	2.735
<i>odh</i> -triples with demand ($\cdot 10^6$)	0.376	0.376	0.376	0.376	0.376	0.376

$\{a, ht, lt, nc, ct, tp, hu, e\}$, depending on the distance $d(o, d)$ of the origin and destination zones o and d . Further, let $v(p, a)$ be the value of attribute a associated with path p , i.e., in case of for example ht the time spent on board of high-speed trains along p . The observable utility of a railway path p is then computed as

$$\nu_p := \sum_{a \in \{a, ht, lt, nc, ct, tp, hu, e\}} b_a(d(o_p, d_p)) v(p, a)$$

and the overall utility as $u_p = \nu_p + \epsilon_p$, where the non-observable utility ϵ_p is Gumbel distributed. In the case of a car path, the observable utility depends on the total travel time and the fuel consumption, which are both independent of the timetable shifting. As we did not have any further information, we consider the observable car utility as constant for any *od*-pair; this can be interpreted as considering exactly one car path for every *odh*-triple. All coefficients associated with the utility functions are industrial data that were calibrated to match the passenger flows of the unshifted timetable. Of course, the observable rail utility increases when the connection time decreases.

All computations were performed on an Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz with 64 GB of RAM using Gurobi 10.0.1 [11] with an LP-IP gap tolerance of 0.01, a run time limit of 3,600 seconds (per iteration), and a maximum of eight threads in parallel.

4.1 Mode-Based Flow Distribution Results

Tab. 2 reports statistics of the solutions computed by the mode-based flow distribution approach. They provide upper bounds on the actual optimization potentials. The first row of the table contains the run time in seconds, and the second the integrality gap. The approach produces nearly optimal solutions within one hour of computation time. Only the two runs for a shifting interval of $[-4, 4]$ and $[-5, 5]$ minutes terminate at the runtime limit, but with reasonably small optimality gaps of 1.6% and 2.5%, respectively. The next row headlined *Aggregated routed pkm* shows the total passenger kilometers, i.e., the routed demand of an *odh*-triple times the distance in kilometers, added over all *odh*-triples. One can see that even the slight shift of plus-minus one minute bears the potential of increasing the aggregated routed passenger kilometers by almost 2.5%, and up to a $\approx 6.7\%$ increase in case of the largest shifting interval $[-5, 5]$; these are significant improvement potentials. Row *Aggregated expected pkm* reports the sum of all idealized maximum demand values $\bar{U}_q D_{odh}$ over all *odh*-triples such that q is feasible. These numbers can be interpreted as the maximum routeable passenger kilometers over all *odh*-triples if there were no train capacities. Comparing the different values for the different shifting intervals reveals that there is also an increase, though a slightly smaller one, ranging from 1.8% for no shifts vs. plus-minus one minute shifts to $\approx 4.9\%$ increase for $[-5, 5]$ minute shifts. Comparing with the values of the flow that gets routed and the expected flow without capacities, it becomes clear that there is a lost potential between 10.3% for $[-5, 5]$ minute shifts up to 11.8% for $[0, 0]$ shifts. The next two rows of the table labeled *Aggregated routed demand* and *Aggregated expected demand* show the numbers for the aggregated routed demand of the computed solutions,

■ **Table 2** Results for the mode-based flow distribution approach.

	[-0,0]	[-1,1]	[-2,2]	[-3,3]	[-4,4]	[-5,5]
Runtime(s)	8.47	26.02	96.79	724.94	3600	3600
Gap	0.0	0.007	0.009	0.009	0.016	0.025
Aggregated routed pkm($\cdot 10^6$)	104.33	106.89	108.93	110.47	111.02	111.34
Aggregated expected pkm($\cdot 10^6$)	118.35	120.51	121.93	123.20	123.77	124.18
Aggregated routed demand	531573	550468	563670	574419	579490	581223
Aggregated expected demand	592074	611250	622890	632757	638196	641971
Average deviation	8.80	7.70	6.90	6.50	6.30	6.20

■ **Table 3** Results for the iterative flow distribution approach after one iteration.

	[-0,0]	[-1,1]	[-2,2]	[-3,3]	[-4,4]	[-5,5]
Total time(s)	70.14	98.83	170.39	732.73	3703.20	3707.14
Aggregated routed pkm ($\cdot 10^6$)	97.54	99.74	101.36	102.59	103.37	103.50
Aggregated expected pkm ($\cdot 10^6$)	118.35	120.52	121.91	123.19	124.05	124.18
Aggregated routed demand	502470	518964	530252	538093	545226	546659
Aggregated expected demand	592074	611389	622726	632491	640047	642400
Number routed paths	371354	405574	425314	437012	456732	449807
Average deviation	0.70	0.80	0.80	0.80	0.80	0.80

and the aggregated values of the theoretically routable demand for the final shifting. Not surprisingly, these numbers follow a pattern that is similar to the travel kilometer values. In a nutshell, there is an even slightly larger potential to increase the demand when increasing the shifting interval, ranging from a $\approx 3.5\%$ to a $\approx 9.3\%$ increase for $[-1, 1]$ and $[-5, 5]$ shifts, and likewise for the demand that could not be routed ranging between $\approx 10.2\%$ and $\approx 9.5\%$. The values in the last row of the table report the *Average deviation* of the routed flow from the logit distribution of the flow over all feasible paths; it is unfortunately quite large.

4.2 Iterative Flow Distribution Results

The characteristics of the solutions computed by the iterative flow distribution approach can be found in Tab. 3 and 4 for one iteration and five iterations. The first row of both tables, headlined *Total time*, shows the total computation time in seconds, i.e., the time that it takes from computing the initial shifting and the associated flow to the final distributed flow. Most of the time is needed in the first iteration; indeed, the four additional iterations require in the worst case approximately 4 minutes of additional computation time.

Tab. 3 reports, as before, the numbers for the *Aggregated routed pkm*, *Aggregated expected pkm*, *Aggregated routed demand*, and *Aggregated expected demand* for each instance. Comparing the numbers for the aggregated routed pkm and the demand with those of the mode-based approach reveals a notable dip that results from enforcing the logit distribution of the flow over all feasible paths of each *odh*-triple with demand. The decrease ranges between 6.5% and 7.1% for the aggregated routed pkm and 5.4% and 6.3% for the routed demand. The values for the aggregated expected pkm and the aggregated expected demand, however, do not decrease at all. This implies that the timetable shifting does not differ too much between the solutions of the mode distribution approach and the iterative distribution approach, such that the passenger flow is simply more realistic. Investigating the numbers for the average deviation of the routed flow from the logit distribution reveals that the iterative approach achieves a significant improvement in that category, as the maximum average deviation is reduced to 0.8 over all instances in comparison to 8.80 for the mode-based approach.

■ **Table 4** Results for the iterative flow distribution approach after five iterations.

	[-0,0]	[-1,1]	[-2,2]	[-3,3]	[-4,4]	[-5,5]
Total time(s)	201.28	260.28	354.65	952.22	3923.97	3936.02
Aggregated routed pkm($\cdot 10^6$)	100.44	102.99	104.68	106.19	107.04	107.19
Aggregated expected pkm($\cdot 10^6$)	118.35	120.52	121.91	123.19	124.05	124.18
Aggregated routed demand	517335	536208	548000	557654	565443	567190
Aggregated expected demand	592074	611389	622726	632491	640047	642400
Number routed paths	362917	388528	405733	415298	432269	425393
Average deviation	0.90	0.90	1.00	1.00	1.00	1.00

Tab. 4 reveals at first glance that the additional iterations to redistribute the unrouted excess flow require only a marginal increase in computation time, while the numbers for aggregated routed pkm and routed demand show a significant increase in comparison to the numbers for the solutions after one iteration of the iterative flow distribution method. The values for aggregated expected pkm and demand do not differ at all from the values after one iteration and only slightly from the numbers computed by the mode-based flow distribution. Comparing the numbers in Tab. 4 for aggregated routed pkm and demand to the other two tables shows that the additional iterations of flow redistribution make up for 3.6% to 3.9% of the aggregated routed pkm, and 2.6% to 2.9% of the aggregated routed demand. Thus the gap between the mode-based upper bound and the result of the iterative flow distribution is closed by at least 52% for the aggregated routed pkm and by 40% for the aggregated routed demand. This is quite a lot.

As more flow is routed on secondary paths, the values for the average deviation from the logit flow on feasible paths increase, but rather marginally, from at most 0.8 for the solutions found after the first iteration to a deviation of at most 1.0 for solutions found after five iterations. can be observed for the number of routed paths in Tab. 3 and 4. Here each path is counted if it bears at least some flow. One can see that the iterative flow distribution approach routes the flow on way more paths, but that this number decreases with an increasing number of iterations, which is a direct consequence of the rerouting procedure and a tradeoff with respect to routing more demand. that there is a steeper improvement for the shifting intervals of $[-1, 1]$, $[-2, 2]$, $[-3, 3]$ minutes, while the additional improvement flattens a bit for the larger intervals. This shows that most of the potential can be harvested with relatively small shiftings.

To provide some insight into the transfer volumes at stations, Fig. 2 plots the aggregated numbers of passengers using newly established connections per station. In each of the four subfigures, there is a circle for each station with newly established connections, centered at the station's location, with a radius corresponding to the number of transferring passengers, i.e., a larger radius means more transferring passengers. Subfigure 2a shows the results for the solution of the mode-based approach and shifting interval $[-5, 5]$. There are only green circles, which correspond to positive numbers of passengers using newly established connections. In contrast to that, the other three subfigures 2b, 2c, and 2d, show differences in numbers of passengers using newly established connections between each combination of the solution of the iterative distribution approach with one (I1) and five (I5) rounds and the mode-based approach (MB), e.g., subfig. 2b (I1-MB) plots in green/red circles how many more/less passengers use newly established connections as a result of I1 in comparison to MB, and so on for the other combinations. All figures show that newly established connections are used to route passengers all over the country and not only at regional clusters. The two subfigures comparing the solutions of the iteratively distributed flow to the mode-based approach solution(2b and 2c) show more green circles, i.e., more stations with

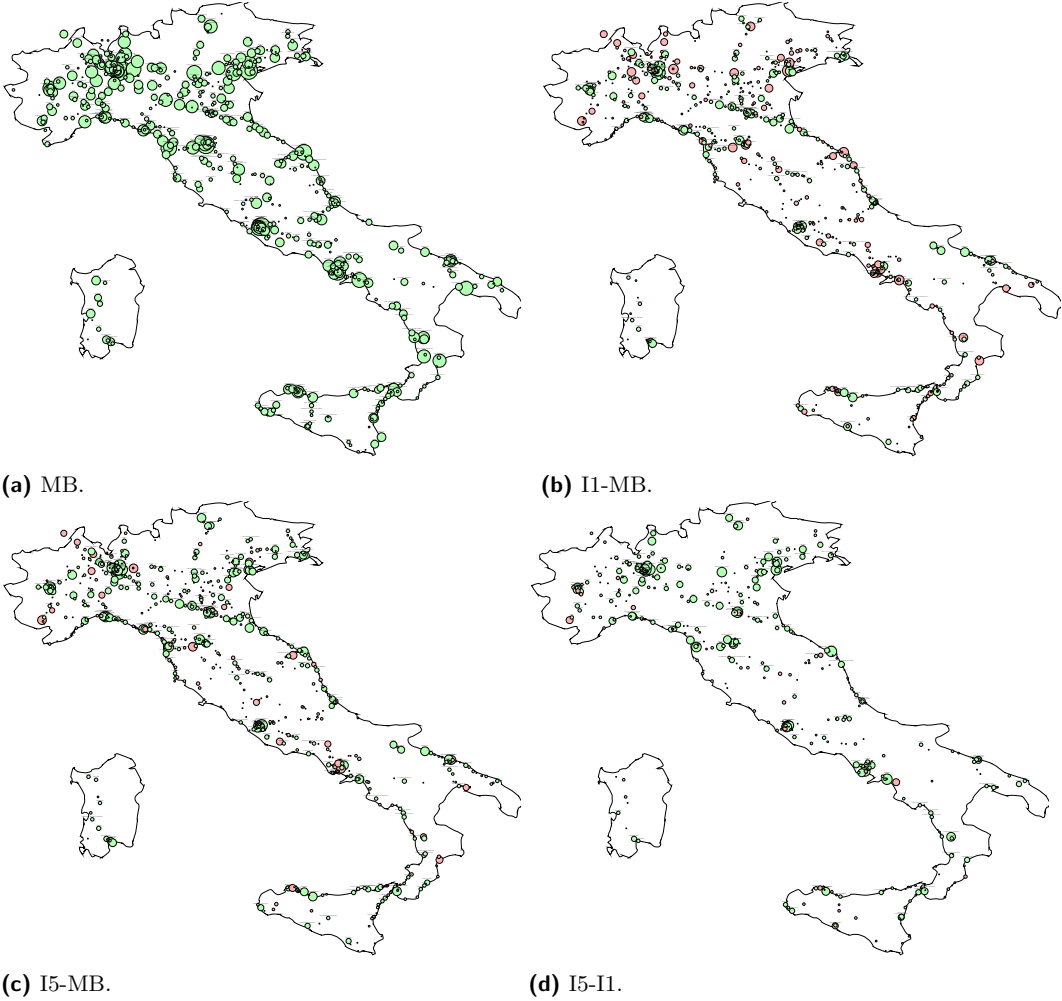


Figure 2 Illustration of the aggregated transfer volumes for newly established connections at stations for solutions of the different approaches considering a shifting interval of $[-5, 5]$.

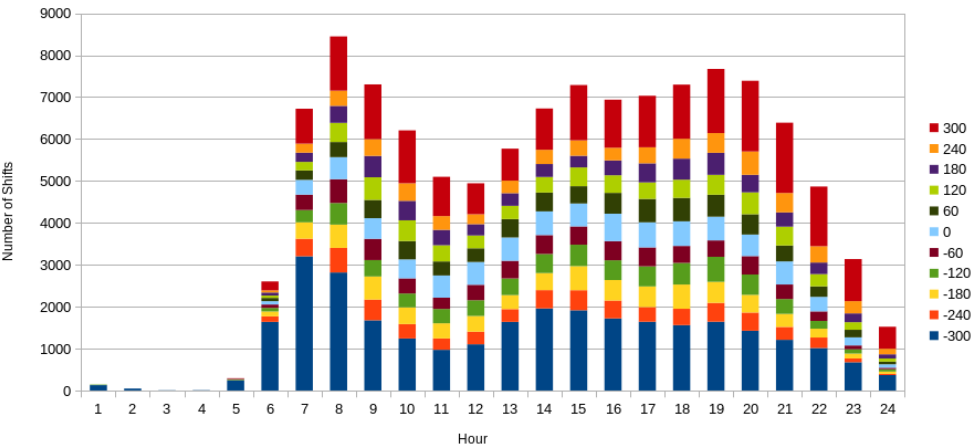


Figure 3 Hourly distribution of time shifts of trains for the solution computed with five iterations of flow distribution considering a shifting interval of $[-5, 5]$.

newly established transfers, which is a consequence of the larger number of paths over which the passenger flow becomes distributed. Looking at the red circles, i.e., stations with less transfers, we see more and larger red circles for Subfig. 2b as compared to Subfig. 2c. The reason for that is that the overall amount of flow is smaller for I1 compared to I5, as well as that I5 routes more flow on paths shared with MB than I1. Finally, Subfig. 2b compares the transfers for the solutions of I1 and I5. The larger number and radii of green circles here show that there is a significantly increased number of passengers using newly established connections for solutions of I5.

Fig. 3 shows the distribution of time shifts of the trains in the optimized timetable computed with five rounds of iterative flow distributions considering a shifting interval of $[-5, 5]$. It is representative for all our approaches: they all show a similar pattern in shifting the largest number of trains by the most negative value, followed by the largest positive value as a runner-up. For most hours, these two shifts comprise more than 50% of all time shifts. Especially in the morning, there is a large number of negative time shifts that decreases to the end of the day, while the number of positive shifts tends to increase toward the end of the day, i.e., the timetable is spread apart. Additionally, one can see that roughly 90% of all trains get shifted in some way. Thus, even if the individual shifts of the trains are small, the number of shifts and their impact on the timetable is large. The overall curve follows the classical steep-morning-wider-afternoon peak “camel distribution”.

5 Conclusion

In this paper, we study the complex task of connection optimization considering elastic demands. To define the problem, we iteratively transformed a basic formulation of the connections problem into a version that includes a logit model to predict passenger behavior. As the problem is non-linear and even non-convex, we presented two approximation heuristics, the mode-based distribution method and the iterative distribution method, based on a mixed-integer linear programming formulation that is used as the backbone of the heuristics. The heuristics are evaluated on real-world data of the Italian long-distance railway network. It turns out that it is possible to solve the model for nationwide scenarios in reasonable time, and that there is a significant potential to increase passenger demand by shifting the timetable by a small margin. In comparison to operating or buying more vehicles, not to speak of building more infrastructure, timetable shifting is a promising way to increase passenger satisfaction and generate higher revenues at a relatively low cost.

References

- 1 Moshe Ben-Akiva and Steven R. Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT Press, Cambridge, MA, 1985.
- 2 Stefan Binder, MY Maknoon, Sh Sharif Azadeh, and M Bierlaire. Passenger-centric timetable rescheduling: A user equilibrium approach. *Transportation Research Part C: Emerging Technologies*, 132:103368, 2021.
- 3 Ralf Borndörfer, Martin Grötschel, Sascha Lukac, May Mitusch, Thomas Schlechte, Sören Schultz, and Andreas Tanner. An auctioning approach to railway slot allocation. *Competition and Regulation in Network Industries*, 1(2):163–196, 2006. ZIB Report 05-45. URL: <http://www.zib.de/PaperWeb/abstracts/ZR-05-45>.
- 4 Ralf Borndörfer, Heide Hoppmann, and Marika Karbstein. Passenger routing for periodic timetable optimization. *Public Transport*, 9(1):115–135, 2017. doi:10.1007/S12469-016-0132-0.

- 5 Valentina Cacchiani and Paolo Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012. doi:10.1016/J.EJOR.2011.11.003.
- 6 Gabrio Caimi, Leo Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6(4):285–312, 2017. doi:10.1016/j.jrtpm.2016.11.002.
- 7 Zhichao Cao, Avishai (Avi) Ceder, Dewei Li, and Silin Zhang. Optimal synchronization and coordination of actual passenger-rail timetables. *Journal of Intelligent Transportation Systems*, 23(3):231–249, 2019. doi:10.1080/15472450.2018.1488132.
- 8 Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and solving the train timetabling problem. *Oper. Res.*, 50(5):851–861, October 2002. doi:10.1287/OPRE.50.5.851.362.
- 9 A Ceder, B Golany, and O Tal. Creating bus timetables with maximal synchronization. *Transportation Research Part A: Policy and Practice*, 35(10):913–928, 2001. doi:10.1016/S0965-8564(00)00032-X.
- 10 Giovanni Luca Giacco and Paolo Dell’Olmo. Complex intermodal network optimization. 2022.
- 11 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL: https://docs.gurobi.com/_downloads/optimizer/en/13.0/pdf/.
- 12 Johann Hartleb and Marie Schmidt. Railway timetabling with integrated passenger distribution. *European Journal of Operational Research*, 298(3):953–966, 2022. doi:10.1016/J.EJOR.2021.06.025.
- 13 Levin Hommel. An approach to railway connection optimization. Master’s thesis, Freie Universität, 2026.
- 14 Yu Ke, Lei Nie, Christian Liebchen, Wuyang Yuan, and Xin Wu. Improving synchronization in an air and high-speed rail integration service via adjusting a rail timetable: A real-world case study in china. *Journal of Advanced Transportation*, 2020(1):5081315, 2020. doi:10.1155/2020/5081315.
- 15 Yu Ke, Lei Nie, and Wuyang Yuan. Joint optimization of flight and train timetables for air and high-speed railway integration services with maximum accessibility. *Transportmetrica B: Transport Dynamics*, 10(1):207–236, 2022. doi:10.1080/21680566.2021.1985009.
- 16 Christian Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435, 2008. doi:10.1287/TRSC.1080.0240.
- 17 Richard M. Lusby, Matthias Larsen, Jesper Ehrgott, and David Ryan. Railway track allocation: Models and methods. *OR Spectrum*, 33:843–883, 2009. doi:10.1007/s00291-009-0189-0.
- 18 Daniel McFadden. Conditional logit analysis of qualitative choice behavior. In Paul Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. Academic Press, New York, 1974.
- 19 Leon W. P. Peeters. *Cyclic Railway Timetable Optimization*. PhD thesis, Erasmus University Rotterdam, Rotterdam, The Netherlands, June 2003. URL: <http://hdl.handle.net/1765/429>.
- 20 Gert-Jaap Polinder, Marie Schmidt, and Dennis Huisman. Timetabling for strategic passenger railway planning. *Transportation Research Part B: Methodological*, 146:111–135, 2021.
- 21 Marie Schmidt. Integrating routing decisions in network problems. In *Operations Research Proceedings 2012: Selected Papers of the International Annual Conference of the German Operations Research Society (GOR), Leibniz University of Hannover, Germany, September 5-7, 2012*, pages 21–26. Springer, 2013. doi:10.1007/978-3-319-00795-3_4.
- 22 Marie Schmidt and Anita Schöbel. The complexity of integrating passenger routing decisions in public transportation models. *Networks*, 65(3):228–243, 2015. doi:10.1002/NET.21600.
- 23 Anita Schöbel and Susanne Scholl. Line planning with minimal traveling time. In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS’05)(2006)*, pages 1–16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2006. doi:10.4230/OASIcs.ATMOS.2005.660.
- 24 Bernardo Szpigel et al. Optimal train scheduling on a single line railway. *Journal of Operations Research*, 1973.

- 25 Kenneth E. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, Cambridge, 2 edition, 2009.
- 26 Rolf N. van Lieshout and Kevin Dalmeijer. A unified approach to evaluation and routing in public transport systems, May 2024. Revised and expanded version of the preprint originally entitled “How (not) to Evaluate Passenger Routes, Timetables and Line Plans”. doi:10.48550/arXiv.2207.09969.
- 27 Rachel C. W. Wong, Tony W. Y. Yuen, Kwok Wah Fung, and Janny May-Yee Leung. Optimizing timetable synchronization for rail mass transit. *Transp. Sci.*, 42(1):57–69, 2008. doi:10.1287/TRSC.1070.0200.

A

 Appendix

■ **Table 5** Definition of used variables and sets.

I	Set of trains.
S	Set of stations.
S_i	Set of stations at which train i stops.
A	Set of trips.
Z	Set of zones.
t_i^s	Integer variable deciding how the stop of train i at station s is shifted.
x_{ij}^s	Binary decision variable if connection between i and j at station s is feasible.
y_p	Rational variable measuring the fraction of demand of D_{od}^h routed on path p .
$u_p(t)$	Function computing the utility of path p for timetable shifting t .
$\bar{u}_p$	Upper bound on the utility on path p for all timetable shiftings.
$\underline{u}_p$	Lower bound on the utility on path p for all timetable shiftings.
$\bar{y}_p$	Rational variable measuring the logit passenger flow on path p .
s_p	Rational variable measuring the deviation from the logit flow on path p .
Inf_{ij}^s	Minimum time shift difference of stops at s of trips i and j .
Sup_{ij}^s	Maximum time shift difference of stops at s of trips i and j .
Inf_i^s	Minimum time shift difference of stop s of trip i .
Sup_i^s	Maximum time shift difference of stop s of trip i .
m_{ij}^s	Minimum transfer time between stops of i and j at s .
β	Maximum connection time.
P_{odh}^r	Set of railway paths between zones o and d departing between hour h and $h + 1$.
P_{odh}^c	Set of car paths between zones o and d departing between hour h and $h + 1$.
P_{odh}	Set of travel paths between zones o and d departing between hour h and $h + 1$.
P	Set of all travel paths.
κ_a	Capacity of trip a .
$\bar{U}_p$	Upper bound on the modal split share for railway paths with utility of p or less.
λ	Non-negative objective weight for connections.
μ	Non-negative objective weight for the railway passenger volume.
ν	Non-negative objective penalty for path-based deviations from the logit flow.

Time-Aware A* for Optimal Train Routing on Moving Block Systems

Stefan Engels¹   

Chair for Design Automation, Technical University of Munich, Germany

Robert Wille   

Chair for Design Automation, Technical University of Munich, Germany

Abstract

Railway networks are operating at their capacity limits. Modern control systems allow for shorter train-following times on existing infrastructure, resulting in faster travel times and higher capacity. As new degrees of freedom emerge, developing effective and flexible algorithms becomes essential to incorporate these advancements into the planning process. Previously, a flexible method for generating provably optimal timetables using A* search has been proposed. However, in practice, the algorithm spends significant time exploring states that yield no new information. This work addresses this issue and shows that transforming the state space into a (directed) tree drastically reduces runtime and enhances scalability, particularly with respect to the number of trains. The implementation is available open-source as part of the *Munich Train Control Toolkit* on GitHub at <https://github.com/cda-tum/mtct>.

2012 ACM Subject Classification Applied computing → Transportation

Keywords and phrases ETCS, European Train Control System, Train Routing, Moving Block, A*, Search Space Reduction, Munich Train Control Toolkit

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.3

Supplementary Material *Software (Source Code)*: <https://github.com/cda-tum/mtct> [2]
archived at `swb:1:dir:2ffb77c8877a842cbc78f45016d4270dd5aeff5d`

1 Introduction

Sustainable transportation relies heavily on railway transportation. At the same time, railway networks are already operating at their capacity limits.

In general, trains cannot operate on sight due to their long braking distances. Hence, control systems have to be implemented to prevent collisions. Previous national rules have been unified across Europe and other regions, most notably, the *European Train Control System* (ETCS, Europe and outside of Europe), *Chinese Train Control System* (CTCS, China), *Positive Train Control System* (PTC, North America), and *Communication Based Train Control* (CBTC, metro systems) [8, 12].

These control systems come in different variants. Modern versions (e.g., moving-block-based) allow for shorter train-following times and, thus, increased capacity. At the same time, train separation is no longer enforced with discrete blocks or even at discrete points, but is continuously monitored.

Since the separation imposed by the control system significantly affects the feasible trajectories of trains, they need to be considered when, e.g., creating timetables. Most research on optimization problems within that context focuses on classical control systems [1]. We are only aware of limited research on modern moving-block systems for timetable optimization and train routing [11, 7, 3, 4].

¹ Corresponding author



© Stefan Engels and Robert Wille;
licensed under Creative Commons License CC-BY 4.0

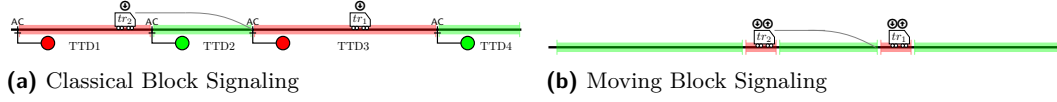
26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 3; pp. 3:1–3:19

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Schematic drawings of different signaling principles [3].

[4] proposes a flexible approach based on search algorithms such as A*, which have already been successfully applied to various design automation tasks, e.g., in nanotechnologies [6] and even within the railway domain [9]. Their approach can use any (possibly black-box) simulator to evaluate given states. By doing so, constraints enforced by the control systems, which are usually hard to model, do not have to be discretized, simplified, and/or approximated.

Even though [4] proposes various strategies to improve runtime, it cannot solve several instances on the benchmark it tested. This is partially due to the search algorithm revisiting many states it has already explored, so it does not gain any new information. In this work, we propose adaptations to the search space that prevent such reexplorations, thereby drastically reducing the search space. The key idea is to make the state expansion time-aware, so that the search focuses only on the earliest pending routing decision. This reduces redundant decisions that do not provide new information, while retaining the optimality guarantee. A case study confirms significant runtime improvements on a benchmark set.

The remainder of this work is structured as follows. Section 2 reviews relevant background on railway systems and search algorithms. Section 3 describes the previous approach by [4], focusing on the details relevant to our proposed changes and explaining what structural properties limit its scalability. Based on this, Section 4 proposes further reductions of the search space and is the main contribution of this work. Finally, a case study (Section 5) demonstrates significant improvements to the scalability with respect to the number of trains. Section 6 concludes this work.

2 Background

This section provides basic concepts and definitions relevant to this work.

2.1 Train Control Principles

Because trains usually have long braking distances, control systems are required to prevent collisions. Classically, such systems rely on predefined fixed blocks. Because those blocks often require *trackside train detection* (TTD) systems, we refer to them as TTD sections. As long as a train is anywhere within a block section, any other train has to be able to come to a full stop before reaching that block section, even if the boundary of the section might be far away from the actual position of the blocking train.

Modern control systems, on the other hand, no longer rely on such fixed sections. Instead, trains are continuously separated by their absolute braking distances and trains “only” have to be able to stop before the previous train’s exact position. Generally, this allows shorter train following times.

► **Example 1.** Consider the network in Figure 1. Classically, it might be separated into 4 TTD sections. In Figure 1a, tr_2 is only allowed to advance to the end of TTD2, since TTD3 is occupied. By contrast, in Figure 1b, tr_2 is allowed to advance all the way to tr_1 (possibly minus some buffer).

2.2 Problem Formulation

In this work, we consider the task of routing trains through a part of a railway network. Trains are released at prespecified times at the boundaries of the considered region. They have certain demands, i.e., which stations they have to service. However, the precise route and timing are not yet decided. More precisely, we solve the following routing task:

► **Problem 2** (Optimal Train Routing (on Moving Block Systems)). Given:

- A railway network with vertices V and (directed) edges E as described in [4, Appx. A].
- A set $\mathcal{T}$ of trains, where w_i denotes the weight of importance of train $i \in \mathcal{T}$
- Demands for every train $i \in \mathcal{T}$ consisting of:
 - entry node $u_{in}^{(i)} \in V$ together with entry time $\tau_{in}^{(i)}$ and initial velocity $v_0^{(i)}$,
 - exit node $u_{out}^{(i)} \in V$ together with earliest exit time $\tau_{out}^{(i)}$ and maximal exit velocity $v_{out}^{(i)}$,
 - list of stations $\mathcal{S}_1, \dots, \mathcal{S}_{m_i} \subset E$ together with stopping requests, i.e., a (desired) arrival time $\tau_j^{(i)}$ and a minimal stopping time $\Delta\tau_j^{(i)}$ for every station $\mathcal{S}_j$.

Task: Find a routing satisfying every train's demand, obeying the constraints imposed by a moving-block control system and earliest departure times, and minimizing the (weighted) exit times and average station delays. To trade-off station and exit delays, we add a constant $\alpha \geq 0$. More precisely, let $t_{out}^{(i)}$ be the exit time and $t_1^{(i)}, \dots, t_{m_i}^{(i)}$ the station arrival times of a given train $i \in \mathcal{T}$. The objective (to be minimized) is given by

$$\sum_{i \in \mathcal{T}} \left(w_i \cdot [t_{out}^{(i)} - \tau_{out}^{(i)}]_+ + \alpha \cdot \frac{w_i}{m_i} \sum_{j=1}^{m_i} [t_j^{(i)} - \tau_j^{(i)}]_+ \right), \quad (1)$$

where $[x]_+ := \max(0, x)$ denotes the ReLU function.

Since we assume that a train is not allowed to leave the network before its earliest exit time, Equation (1) is equal to

$$\sum_{i \in \mathcal{T}} \left(w_i \cdot t_{out}^{(i)} + \alpha \cdot \frac{w_i}{m_i} \sum_{j=1}^{m_i} [t_j^{(i)} - \tau_j^{(i)}]_+ \right) - \sum_{i \in \mathcal{T}} w_i \tau_{out}^{(i)}. \quad (2)$$

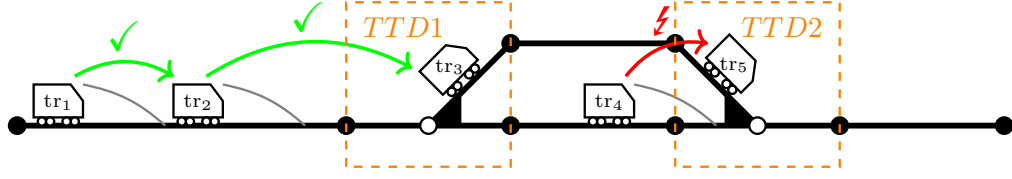
Since $\sum_{i \in \mathcal{T}} w_i \tau_{out}^{(i)}$ is a constant, minimizing Equation (1) is equivalent to minimizing

$$\sum_{i \in \mathcal{T}} \left(w_i \cdot t_{out}^{(i)} + \alpha \cdot \frac{w_i}{m_i} \sum_{j=1}^{m_i} [t_j^{(i)} - \tau_j^{(i)}]_+ \right). \quad (3)$$

For the remainder of the paper, we use the objective in the form presented in Equation (3).

All constraints imposed by the train control system (in our context, a moving-block control system) must be followed. Even though we optimize in the context of a moving-block system, some TTD sections remain on the network. For simplicity and to provide a basic level of flank protection, we assume that all turnouts are surrounded by a classical TTD section. On the contrary, there are usually no other TTD sections in our instances. For the remainder of this work, we can think of TTD sections and turnouts to be somewhat “equivalent”.

► **Example 3.** Consider the network depicted in Figure 2 with 5 trains and relevant braking distances marked with gray lines. tr_1 and tr_2 are on the same track segment. However, this is fine because a moving-block control system is assumed. We only enforce classical TTD



■ **Figure 2** Example network.

separation at the turnouts. Because of that, tr_4 conflicts with tr_5 , because it cannot stop before the TTD section², which tr_5 already entered. On the other hand, tr_2 and tr_3 are safely separated even though a TTD section is enforced around the left turnout.

2.3 (Weighted) A* Algorithm

Let $G = (S, A, c)$ be a directed and weighted graph with vertices S , arcs A , and cost $c: A \rightarrow \mathbb{R}$. Furthermore, we are given an initial vertex $s_0 \in S$ and a goal set $\mathcal{G} \subseteq S$. The goal is to find a minimal cost path from s_0 to any vertex in $\mathcal{G}$. One can also think of the vertices as states, and the arcs as denoting possible actions or transitions with respective costs c .

A* can be used to find such minimal paths. It functions very similarly to Dijkstra; however, instead of expanding the path with the lowest cost, it also adds a guiding heuristic estimating the distance to a goal state, see Algorithm 1, where g denotes the currently best known path cost from s_0 to s and f the cost adjusted with the guiding heuristic.

■ **Algorithm 1** (Weighted) A*-Algorithm [5, 10].

```

C ← ∅; f(s) ← ∞ for all s ∈ V; g(s) ← ∞ for all s ∈ V
add s0 to priority queue (pq); f(s0) ← 0
while pq is not empty:
    select s ∈ argminσ ∈ pq f(σ)
    C ← C ∪ {s}
    if s ∈ G: return s together with observed path
    explore all successors Γ(s)
    for s+ ∈ Γ(s) − C:
        add or update s+ to pq
        f(s+) ← min{f(s+), g(s) + c(s, s+) + ε · h(s+)}
        g(s+) ← min{g(s+), g(s) + c(s, s+)}
return infeasible

```

Special focus should be placed on the choice of the guiding heuristic. If one chooses $h \equiv 0$, Algorithm 1 is basically equivalent to Dijkstra. If h never overestimates the cost of any arc (more precisely, if h is consistent), A* is known to be optimal.

► **Definition 4** (Consistent Heuristic [5]). *A heuristic h is consistent if $h(s_g) = 0$ for every $s_g \in \mathcal{G}$ and $h(s) - h(s^+) \leq c(s, s^+)$ for every $(s, s^+) \in A$.*

The heuristic can be multiplied by a factor $\varepsilon \geq 1$ to get an ε -approximation algorithm. In particular, the following properties and optimality guarantees are well-known:

² If tr_4 were to already be stopped, there would be no conflict.

► **Theorem 5** (ε -Approximation [10]). *If h is consistent, then Algorithm 1 returns a path whose length is at most ε times the minimal path length.*

► **Corollary 6** (Optimality [5]). *If $\varepsilon = 1$ and h is consistent, then Algorithm 1 returns a minimal path.*

3 Previous Work

Previously, an algorithm to solve (a slightly different variant of) Problem 2 has been proposed and implemented [4]. The main idea was that, under the arguably reasonable assumption that trains drive as fast as possible and allowed by the control system, a solution is uniquely defined by the following decisions:

- routing, i.e., the edges used by every train,
- stops, i.e., the exact stop positions within the scheduled stations, and
- ordering, i.e., the order of the trains on every turnout as well as entry and exit vertices.

In that case, any simulator at hand can be used to evaluate a given solution, and A* can be used as a heuristic search algorithm to find an optimal solution.

We summarize the key concepts proposed by [4] in the following subsections. Where applicable, we slightly modify their results to be consistent with this paper.

3.1 State Space

The main idea was to expand routes incrementally. For this, the state space consists not only of all valid solutions, but also of those that are only partially defined and then extended towards fully specified solutions.

3.1.1 States

► **Definition 7** (State [4, Definition 9]). *A state is given by any partial solution. More precisely, it consists of information for every train $i \in \mathcal{T}$:*

- a valid path $e_1^{(i)}, \dots, e_{k_i}^{(i)} \in E$, possibly empty, starting in $u_{in}^{(i)}$,
- a list of positions $s_1^{(i)}, \dots, s_{\hat{m}_i}^{(i)}$ being within the first $0 \leq \hat{m}_i \leq m_i$ scheduled stations $\mathcal{S}_1^{(i)}, \dots, \mathcal{S}_{\hat{m}_i}^{(i)}$, and
- the train ordering on every TTD section as well as on entry and exit vertices.

By design, states do not need to specify paths leading to an exit vertex. If a train's path is not fully specified, it is assumed to travel all the way to its last specified edge and then disappear. At the same time, a train that disappeared within the network blocks its last vertex indefinitely, so that it cannot be overtaken by any other train. Because a train's route might not be fully specified, a partial solution is also not required to stop in all scheduled stations.

► **Assumption 8** (Deterministic State Value). *It is assumed that train movements are uniquely defined by the information contained in Definition 7. This can, e.g., be achieved by the discrete-time simulator proposed in [4, Section 5]; however, any other (black-box) simulator could be used.*

The objective value is then given by

$$g(s) := \sum_{i \in \mathcal{T}} \left(\underbrace{w_i \cdot \hat{t}_{k_i}^{(i)} + \alpha \cdot \frac{w_i}{m_i} \sum_{j=1}^{\hat{m}_i} [\hat{t}_j^{(i)} - \tau_j^{(i)}]_+}_{=: g^{(i)}(s)} \right) = \sum_{i \in \mathcal{T}} g^{(i)}(s), \quad (4)$$

where $\hat{t}_1^{(i)}, \dots, \hat{t}_{\hat{m}_i}^{(i)}$ are the arrival times at the specified stops and $\hat{t}_{k_i}^{(i)}$ is the arrival time at the end of edge $e_{k_i}^{(i)}$. Note that the average stop delay already uses the total number of scheduled stops, even if not all stops are specified yet. This simplifies the definition of a consistent heuristic.

If a state is infeasible (for example, because two trains are blocking each other and cannot reach their final edge), we set $g(s) = \infty$.

► **Definition 9 (Goal States).** The set of goal states $\mathcal{G}$ consists of all states which are fully specified (i.e., for all $i \in \mathcal{T}$, $e_{k_i}^{(i)}$ ends at exit $u_{out}^{(i)}$ and $\hat{m}_i = m_i$) and feasible (i.e., $g(s) < \infty$).

3.1.2 Transitions

Having defined the states, it is necessary to define how the search algorithm can traverse the state space. Given any state s , a successor s^+ is obtained by extending the train path of one train by one or multiple edges. For a specific train, [4] proposes different strategies for choosing the number of edges to extend. One strategy extends by a single edge at a time; another strategy combines edges up to the next turnout or scheduled stop.

However, more important for the remainder of this work is to understand how the train ordering is deduced. For this, [4] assumes that whichever train chooses a certain edge first also travels on that edge first. Since a state is generally expanded on every train, all possible train orderings are eventually expanded.

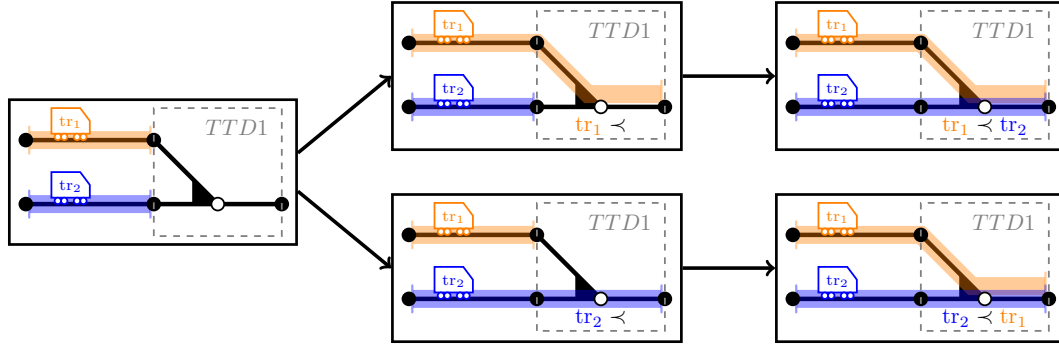
► **Example 10.** Consider Figure 3 with two trains approaching a turnout from different tracks. The left state can be expanded in two ways. One successor state is achieved by moving tr_1 (middle top). By doing so, it is implied that tr_1 moves across the turnout first. This state itself can be expanded by moving tr_2 , in which case the resulting state (top right) assumes that tr_2 waits until tr_1 clears the turnout and then follows. On the other hand, the left state is also expanded by moving tr_2 . Analogously, the bottom right state is reached, in which tr_1 waits for tr_2 to clear the turnout before following.

3.1.3 Cost

Note that we define a constant cost on states only. Thus, the state transition costs used in Algorithm 1 are only given implicitly by $c(s, s^+) := g(s^+) - g(s)$.

W.l.o.g., assume that a single train's path is extended by a certain number of edges. Then every train should either travel identically to the previous state or be slower, because the path extension might lead to further constraints that were not present in the previous state. This motivates the following assumption, which [4] implicitly relies on:

► **Assumption 11 (Monotonicity).** Let s^+ be a state that is created by extending state s , i.e., by adding further edges and/or stops to any of the trains in s . Then $g^{(i)}(s^+) \geq g^{(i)}(s)$ for all $i \in \mathcal{T}$. In particular, $g(s^+) \geq g(s)$.



■ **Figure 3** Train Ordering.

Note that Assumption 11 is not phrased as a theorem, and the justification is not a rigid proof. In fact, depending on the driver's strategy assumed by the simulator, this might not be guaranteed. For example, the Greedy Train Driver Assumption in [4, Assumption 7] does not theoretically guarantee this for all instances and states. At the same time, this did not pose any problems in our case study, since such cases seem unlikely in practice.

3.2 Heuristic

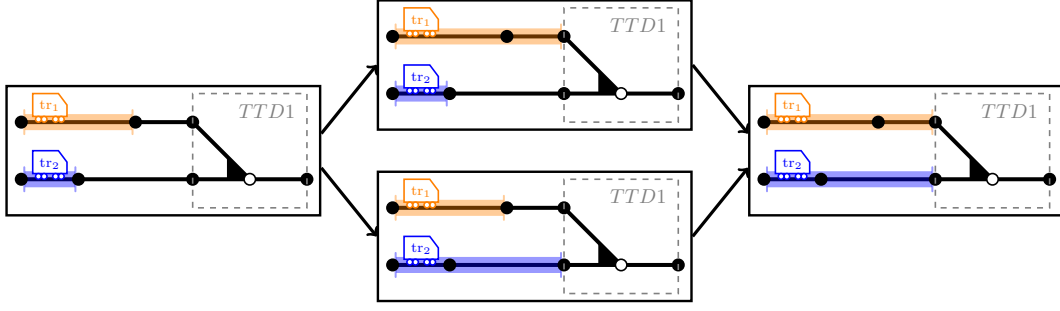
Given any partially defined state, an easy-to-compute guiding heuristic is needed which estimates how far away $g(s)$ is from an optimal goal state, i.e., $h(s) \approx \min_{s_g \in \mathcal{G}} g(s_g) - g(s)$. Obviously, the exact difference is unknown since we are currently searching for the optimal goal state. A natural heuristic is to assume that all trains travel at maximum speed for the remainder of their paths, without any obstacles, i.e., ignoring potential collisions with other trains. [4, Lemma 15] shows that this heuristic is consistent under Assumption 11. Because of this and Corollary 6, they conclude that their proposed method returns optimal solutions to Problem 2 [4, Theorem 16, Corollary 17].

3.3 Problematic State Reexploration

Even though [4] showed that runtime could be improved by reducing the search space using the transition strategies described in Section 3.1.2, the algorithm timed out on many of the tested instances. Adding multiple edges at once and skipping unnecessary states improved the runtime but did not resolve the scalability issue. A major issue with the approach presented in [4] is that many states can be explored multiple times.

► **Example 12.** Consider Figure 4. The right state is expanded from the left state by moving both trains. The algorithm will expand the search space in two ways, reaching the right state. It could either expand tr_1 or tr_2 first. Because both middle states are likely to be considered by the A* search, the state on the right is explored twice. One state was explored without any additional benefit.

Hence, we conclude that it would be desirable to use a tree as the search space, as this would eliminate unnecessary state reexploration. Achieving this is the main contribution of this paper and is described in Section 4.1.



■ **Figure 4** State Reexploration.

4 State Space Reduction

[4] shows that the presented approach using A* search works in principle and finds optimal solutions. However, a naive implementation incurs a high computational cost. To mitigate this, strategies for reducing the search space have been introduced that skip some unnecessary states. This notably improves runtime. At the same time, even the best settings presented in [4] struggle to solve instances with many trains, even if the underlying network structure is rather simple.

This section introduces improvements to this. Recall that generating the set of successor states consists of two steps. First, it is determined which trains are selected³ to advance. Second, for each such train, a set of possible path extensions is generated. We propose improvements for both steps in the following subsections.

4.1 Time-Aware Train Selection

Recall Section 3.3 and that the previous approach by [4] reexplores already expanded states because equivalent decisions are made at several parts of the state space (just in a different order). To prevent this, we aim to make a decision as late as possible. Intuitively, we only want to make the most pressing decision, i.e., the one for the train closest to reaching its specified end-of-route. This makes the train selection somewhat “aware” of the simulated timings of the current state.

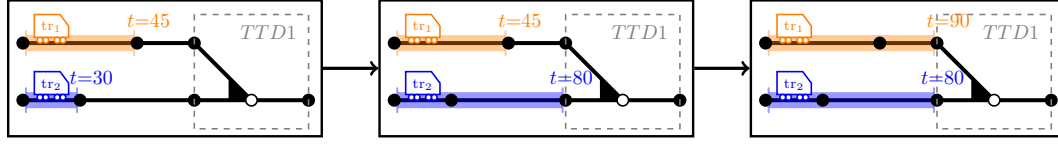
For any given state, we order the trains whose paths are not yet fully specified by the time they reach the end of their currently specified route. We only expand the path of the train that arrives earliest. If multiple trains arrive at the same time, we make a fixed but otherwise arbitrary⁴ choice.

► **Example 13.** Consider Figure 5 with the same situation as in Example 12. Assume that tr_1 needs 45s and tr_2 needs 30s to travel across their first edge. Because the decision for tr_2 is more pressing, it is moved first. Only after that does tr_1 arrive at its route end earliest, so that it is expanded second. In contrast to Example 12, the right state is only explored once, and an irrelevant intermediate state is eliminated.

For simplicity, we have decided to showcase the time-aware train selection strategy with the single-edge path extension. However, it can naturally also be applied to any other extension strategy presented in Section 4.2.

³ The previous approach “selects” all trains whose path has not yet been fully specified.

⁴ In our implementation, we decide to expand trains with higher importance first.



■ **Figure 5** Time-Aware Train Selection.

By ordering the trains by time as proposed in Section 4.1, we ensure that no state is reexplored and the search space becomes a tree. However, the attentive reader might realize that we also cut off valid goal states by doing so. In the previous approach, the train order on a section was determined by whose path was extended onto that section first. Without an additional ordering rule, the order would now be determined by timing, with tracks assigned on a first-come, first-served basis. At the same time, a train may need to wait to let an opposing train clear a section or to let a faster train pass by. Without further refinement, such solutions are pruned and cannot be reached by the search algorithm.

Since it is no longer possible to freely decide which train to advance next, we need to make the ordering decision at a later time. Instead of appending a train to the end of the ordering, we branch into all possible orderings as soon as a new train is expanded into a TTD section. In contrast to the original non-time-aware case, it is now possible to schedule trains ahead of trains already assigned to a track segment.

► **Example 14.** Consider Figure 6a with the same situation as in Example 10. Instead of branching at the left state, tr_2 is expanded onto the shared section first because, in the initial simulation, it arrives there earlier. After that, tr_1 is expanded onto the shared section. To not lose any valid solution, we branch at this point, allowing tr_1 to either go first or second.

We can already exclude some of the orderings, though, which we do not have to expand to. Note that for two trains following each other, a decision on their order is made the first time they enter a shared track segment. As long as their paths do not diverge after that, their order cannot change. Hence, we can forward this choice to later TTD sections without the need for further branching.

► **Example 15.** Consider Figure 6b and assume that tr_3 is expanded. On the right section $TTD2$, it is already decided that tr_1 travels before tr_2 . In theory, tr_3 could go first, second, or third. However, since it shares the path with tr_2 , their relative ordering compared to $TTD1$ cannot change. Hence, tr_3 cannot go last, and this variant is not generated as a successor state.

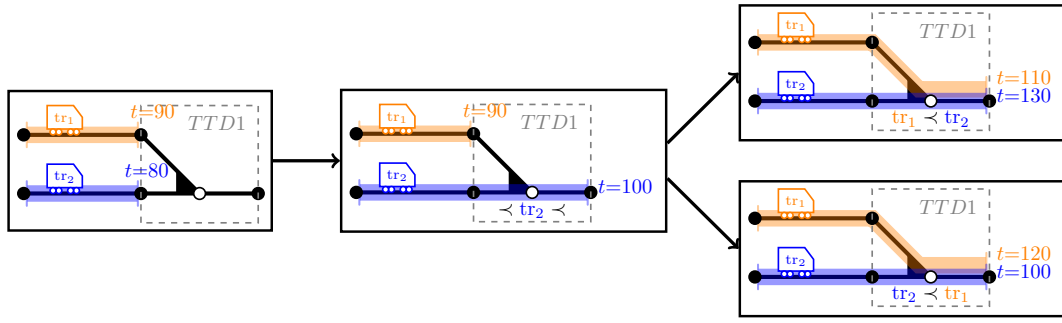
In order to still be able to reach all goal states, we need to slightly modify the simulation assumptions from Section 3.1.1 in two ways:

1. A train that disappeared does *not* block its last vertex.
2. Trains moving in opposite directions on a single track do *not* collide.

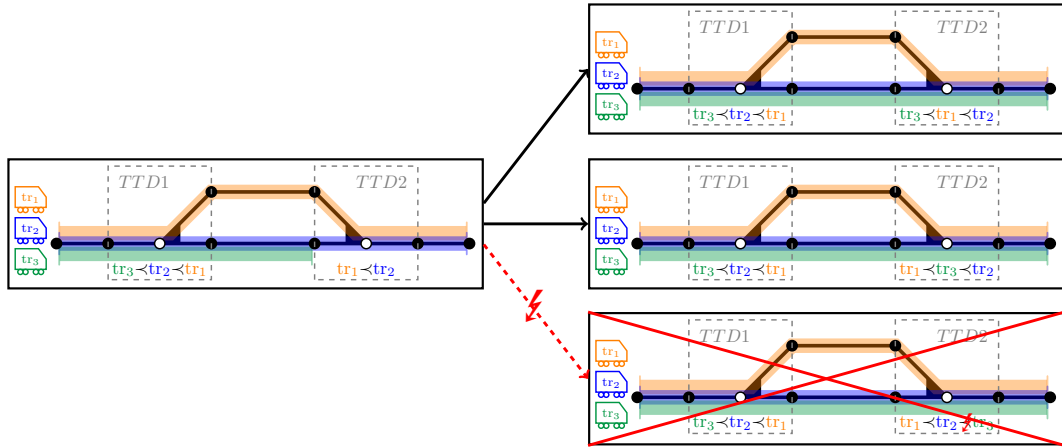
While these assumptions seem counterintuitive, not relaxing the simulation in these points would cut off feasible solutions (see proof of Theorem 17). It is worth noting that these changes do not affect the outcome of goal states, because

1. there are no trains with partial routes that disappear within the network, and
2. the train ordering on the TTD sections adjacent to a single track already prevents head-on collisions.

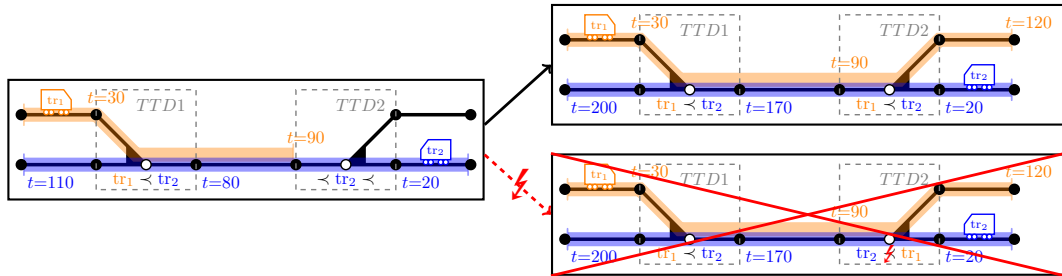
In particular, the relaxed simulation affects only intermediate states that are not returned as solutions; the simulation behavior for returned solutions remains unchanged.



(a) State Transitions.



(b) Deduction of Possible Train Ordering (Same Direction).



(c) Deduction of Possible Train Ordering (Opposite Direction).

■ **Figure 6** Time-Aware Train Ordering.

► **Example 16.**

1. Consider the situation from Example 15, where (without the first relaxation) the left state in Figure 6b would be infeasible, because tr_3 (as going first) would prevent tr_2 from reaching its destination. However, the state might be needed (depending on the train's timing) to reach two valid goal states.
2. Consider Figure 6c. Without the second relaxation, the left state would likely be infeasible due to a head-on collision on the shared track. Again, it might be needed to reach the valid goal state on the right, though. Note that due to the ordering deduction exemplified

in Example 15, no (invalid) successor state with ordering $\text{tr}_2 \prec \text{tr}_1$ on *TTD1* is created. Because of this, the TTD ordering prevents head-on collisions in goal states without further consideration.

It remains to show that the defined new state space is correct (i.e., all goal states can be reached) and prevents the undesired state reexplorations discussed in Section 3.3.

► **Theorem 17.** *In the (time-aware) state space as defined in this section, all feasible goal states can be reached from the initial state.*

Proof. Let $s_g \in \mathcal{G}$ be any feasible goal state and let $B_1^{(i)}, \dots, B_{k_i}^{(i)}$ be the TTD sections traversed by train $i \in \mathcal{T}$ in that goal state. Further let $B_0^{(i)} := u_{in}^{(i)} \in V$ be the train's entry vertex and $B_{k_i+1}^{(i)} := u_{out}^{(i)} \in V$ be the train's exit vertex.

Consider any prefix-state s of s_g , i.e., a partial solution (see Definition 7) where every train's subpath can be extended to the full path in s_g . The orders on all TTD sections and boundary vertices are inherited from s_g by restriction to trains whose partial routes reach the respective element.

Assume that s is not feasible, i.e., the simulation of s reaches a deadlock with at least one blocked train. For a blocked train i , choose one active obstruction $\eta_i = \text{clear}(p_i, j)$, i.e., i is currently blocked because another train j has not yet cleared a point, section, or vertex p_i . Since s_g is feasible, all η_i have been cleared in the goal state, denote the time of clearing in s_g by $t_g(\eta_i)$.

Choose a blocked train i^* with minimal $t_g(\eta_{i^*})$, where $\eta_{i^*} = \text{clear}(p_{i^*}, j)$, i.e., i^* is blocked by train j which has not cleared p_{i^*} during the simulation of state s , hence, j cannot advance. Because trains are assumed to disappear at the end of their partial route without blocking their last vertex indefinitely, j itself must be blocked by some train $k \neq j^5$ which has not yet cleared some p_j . Thus $\eta_j = \text{clear}(p_j, k)$.

We claim that the relative order of the conflicting trains at p_{i^*} and p_j is identical in s and s_g . There are two cases:

1. A train can be blocked by another train because it has not cleared a given TTD section or boundary vertex. In this case, the train order is inherited explicitly by definition.
2. A train can be blocked by another train traveling in the same direction on any intermediate edge. Since trains cannot overtake each other on the same directed track segment, the train order is uniquely determined by the order at the previous TTD section or boundary vertex (which both trains have in common and already passed even in state s)⁶.

Since s_g is feasible and the relative order is identical, j eventually clears p_{i^*} in s_g . However, because j requires p_j to be clear to advance out of p_{i^*} , p_j had to be cleared by k strictly before that. Hence,

$$t_g(\eta_j) = t_g(\text{clear}(p_j, k)) < t_g(\text{clear}(p_{i^*}, j)) = t_g(\eta_{i^*}), \quad (5)$$

which is a contradiction to $t_g(\eta_{i^*})$ being minimal.

In particular, all prefix-states of s_g are valid under the modified simulation assumption for partial states.

⁵ $k = i^*$ is possible, but the argument is independent from whether $k = i^*$ or $k \neq i^*$.

⁶ This argument would not work for trains traveling in opposite directions, because they do not share a common previous TTD section. If their partial routes end between two sections, their order might not be uniquely deducible for state s . Hence, the second relaxation to the simulator described above is crucial for the correctness of the A* search.

It remains to show that there is a path from the initial state to s_g in the time-aware state space that uses only prefix-states. Start with the initial state, which is a prefix-state of s_g . Now consider any reachable prefix-state $s \neq s_g$. The time-aware train selection rule (Section 4.1) selects exactly one train that has not yet been fully specified. Apply the path extension that follows the route and stop choices of this train in s_g and, whenever a newly entered TTD section creates multiple ordering branches, choose the branch matching the ordering of s_g . The resulting successor s^+ is again a prefix-state of s_g . Each such step strictly increases the amount of specified information (at least one additional edge and, if applicable, additional stop information), and there are only finitely many edges and scheduled stops. Hence, s_g is finally reached, and there is a path from the initial state to s_g in the time-aware state space. ◀

► **Theorem 18.** *The (time-aware) state space as defined in this section is a (directed) tree.*

Proof. Assume the contrary, then there is a state s with two distinct successors s_1^+ and s_2^+ and a state $\hat{s}$ such that in the state space there is a path from both s_1^+ and s_2^+ to $\hat{s}$. Since s is only expanded on a single train⁷, say tr_1 , in the time-aware setting, s_1^+ and s_2^+ can only differ by the routing data (i.e., path, stop position, or relative train ordering on some TTD section or boundary vertex) of tr_1 . However, this difference would not be reverted (since the routing data is only extended and not changed in future states) and, hence, be forwarded up to $\hat{s}$, which is a contradiction. ◀

4.2 Path Extensions

[4] proposes two ways to extend paths of a given train. The basic idea is to move trains one edge at a time. At a turnout, this leads to multiple successor states; besides that, the path extension is unique.

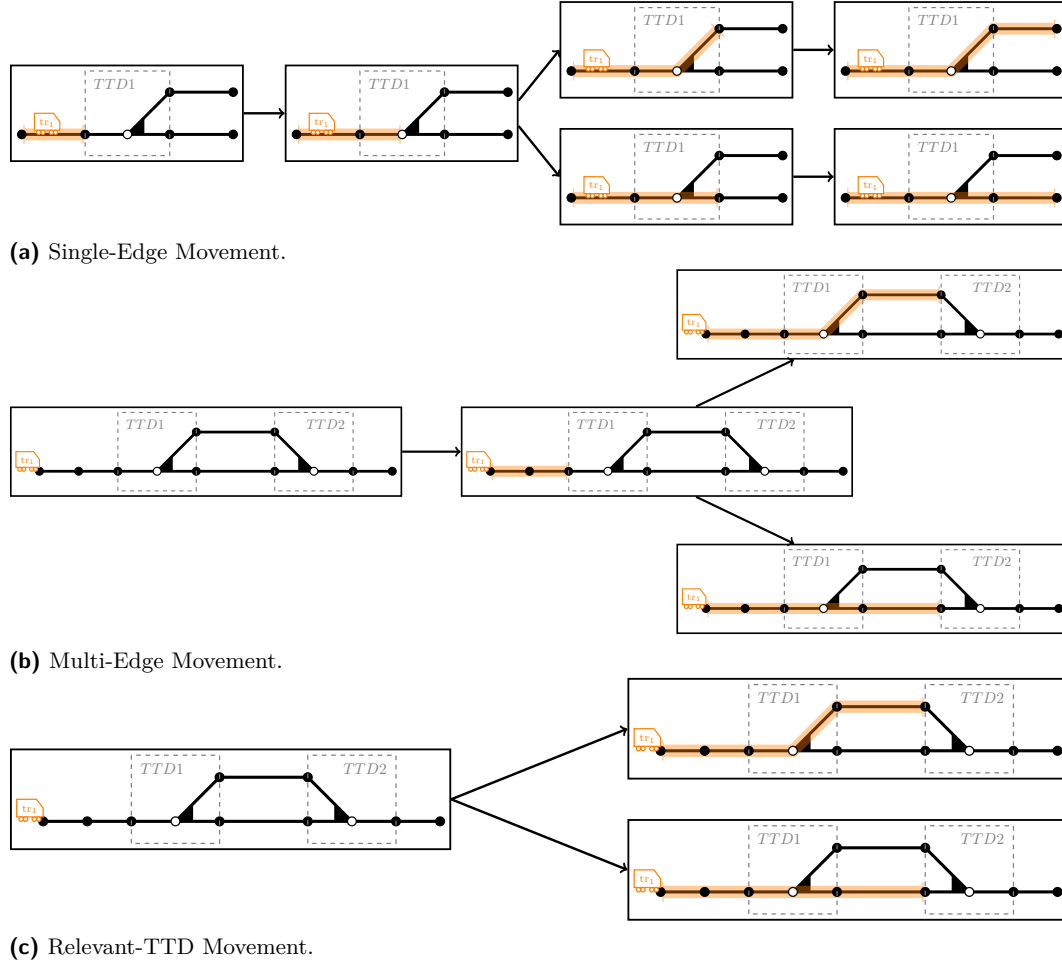
► **Example 19.** Consider Figure 7a with a single train. Neighboring states differ by exactly one edge. At the turnout's vertex, two possible successors exist: the train can either turn left or stay on the main line.

Often, the path extension is unique. Hence, it seems sensible to skip some intermediate steps that do not add value. Recall that in [4] the train ordering is deduced by whichever train extends on an edge first. Thus, we can safely combine multiple edges into a single transition as long as the train ordering is the same across all added edges. On a single track, trains cannot overtake each other. Because of this, [4] proposes to extend train routes up to the next turnout's TTD section.

► **Example 20.** Consider Figure 7b with a single train that has not entered the network yet. First, the train can enter up to TTD1, i.e., the first turnout. It can then transition to either move left or stay on the main line. However, the choice is directly extended up to the second turnout (TTD2).

Looking at Example 20, it is apparent that stopping at TTD1 is not necessary, since any other train trying to cut in must use the edge already blocked by the considered train. If the network consisted of additional trains, these might be moved in intermediate states before the considered train advances. Hence, we propose to further reduce the state space. A train's path extension shall only end at "relevant" TTD sections, i.e., a TTD section that can be traversed by another train without using the entry edge blocked by the current train.

⁷ This did not hold for the state space introduced by [4], which caused the reexploration issue described in Section 3.3.



■ **Figure 7** State Transitions.

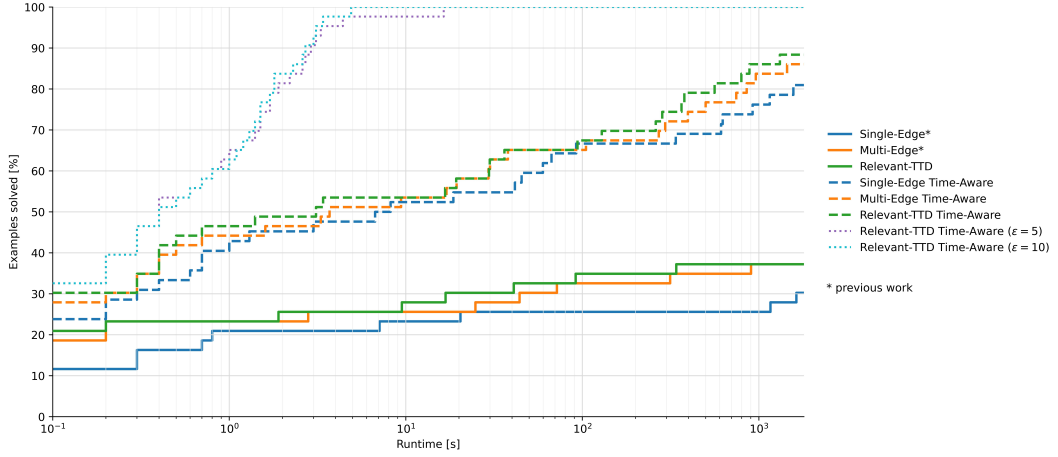
► **Example 21.** Consider Figure 7c with the same initial state as in Example 20. Instead of adding an intermediate state, where the train moves to the first turnout, two successor states are generated directly, both moving to the first “relevant” turnout.

5 Case Study

We extend the implementation from [4] by the adaptations described in Section 4. The proposed method is implemented in C++ and is open-source as part of the *Munich Train Control Toolkit* (MTCT), available on GitHub at <https://github.com/cda-tum/mtct>.

For testing, we use the benchmark from [4], which is also available in the above-mentioned repository. It contains several network families, including instances inspired by the Munich Stammstrecke; these instances have comparatively few routing alternatives but an increasing number of trains. Different variants of the algorithm were tested on an Intel(R) Xeon(R) W-1370P system with a 3.60GHz CPU (8 cores) and 128GB of RAM, running Ubuntu 20.04.

As discussed, the search variants differ in two aspects. First, they differ in the rule used to select the next train whose path is expanded. For this, we compare the strategy from [4] with the time-aware train selection strategy proposed in Section 4.1. In both cases, the relative train orderings are determined by that train selection rule.



■ **Figure 8** Runtime comparison.

For each of the two train selection strategies, we test three path extension strategies: the two already proposed by [4] and the extension discussed in Section 4.2.

- *Single-Edge*: A train’s path is extended by exactly one edge at a time, see Example 19. This strategy was originally proposed by [4].
- *Multi-Edge*: A train’s path is extended by multiple edges at a time until arriving at the next TTD section (usually a turnout), see also Example 20. This strategy was originally proposed by [4].
- *Relevant-TTD*: Similarly to *Multi-Edge*, however, a path extension does not end at a section that cannot be traversed without using the edge on which the train in question approaches it (hence, the train cannot be required to wait there for a train on a different path), see also Example 21.

We ran all instances with a 30-minute (1800-second) timeout. The results are shown in Figure 8. The x-axis denotes runtime in seconds and is log-based, so both easy and hard instances can be compared in a single plot. The y-axis shows the percentage of instances that could be solved within the given time. Hence, in general, lines to the left/top correspond to faster algorithms. The raw data can be found in Appendix A and contains information on runtimes, gaps, the number of (distinct) states simulated, and the number of states that were reexplored.

We were able to reproduce the results reported in [4]. Consider the solid lines in Figure 8 that represent the search algorithm without the time-aware train selection proposed in this work. Again, extending states by more than one edge at a time improved the runtime notably. At the same time, the improvements do not allow solving significantly more instances. Most notably, all variants described in [4] struggle to route trains on instances inspired by the Munich Stammstrecke, even though these instances offer only a limited set of routing choices. This indicates that the previous approach scales badly with the number of trains. In Appendix A, we provide information on how many states were reexplored. In the extreme case, approximately 58% of explored states had been seen and simulated before. This confirms that the problem described in Section 3.3 is not only theoretical, but of practical relevance.

A significantly larger runtime improvement comes from the time-aware train selection strategy (dashed lines) proposed in Section 4.1. On the Munich Stammstrecke instances mentioned above, the resulting runtimes are consistent with the limited number of routing

alternatives. An instance with 16 trains, which timed out in [4], can now be solved in approximately 1 second. While this is an extreme case, it illustrates how strongly reexploring states can affect performance. The same effect, though less pronounced, is also evident across all other tested networks.

All variants discussed so far retain the same *theoretical* optimality guarantee under Assumption 11 [4, Corollary 17]. Consistent with that theory, changing the strategy affected the runtime but did not influence the objective value of the returned solution on the tested instances.

Finally, we ran the best-performing exact strategy with higher weights $\varepsilon \in \{5, 10\}$. The search algorithm is guided more towards goal states, and as a result, the result is no longer guaranteed to be optimal. With this, we were able to find feasible solutions to all tested instances within 20 seconds. Wherever the true optimum was known, we compared the objective values. In the worst case, the objective value doubled. On the other hand, the median increase in objective value was around 9.5% ($\varepsilon = 5$) and 14% ($\varepsilon = 10$) respectively.

6 Conclusions

In this work, we have extended an algorithm for optimal train routing based on A^* [4] by significantly reducing the state space. We conclude that the previous version frequently expanded to states it had already explored before (Section 3.3). Because of this, an alternative state space was proposed by introducing a time-aware train selection rule (Section 4.1). The resulting state space is a (directed) tree, preventing undesired reexploration.

Section 5 showed that the addressed issue had a significant impact on runtimes. Across a small benchmark set, solving times are significantly reduced while still returning solutions with the same objective values as the corresponding exact variants and while retaining the same conditional optimality guarantee. Moreover, the approach remains flexible, and the simulator can (in theory) be replaced by any other simulation system satisfying Assumption 11, e.g., a more efficient event-based simulator. Additionally, this allows easy adaptation to other and future control systems. We provide all implementations open-source as part of the *Munich Train Control Toolkit*, available on GitHub at <https://github.com/cda-tum/mtct>.

At the same time, we still experienced significant runtimes in some instances. Approximate search algorithms on the proposed state space could provide further improvements, but at the cost of losing optimality guarantees. A natural choice is to use a weighted version of A^* that gives more weight to the heuristic and, hence, pushes the algorithm towards goal states more quickly. While this showed a clear additional improvement in efficiency, optimality is no longer guaranteed. A more detailed discussion on this trade-off is out of scope for this paper. Exploring suitable approximate search algorithms for the proposed state space is left for further research.

Overall, this work shows the importance of design choices when defining a state space used with search algorithms. One should take extra care to ensure that, in every iteration, the algorithm extends only towards states that provide new information the search has not revealed before. In our case (since the objective is entirely defined by the state itself, not the transitions), the search space should be a tree. Achieving this requires making counterintuitive design choices (Example 16), but our case study shows that this (likely) pays off significantly.

References

- 1 Ralf Borndörfer, Torsten Klug, Leonardo Lamorgese, Carlo Mannino, Markus Reuther, and Thomas Schlechte, editors. *Handbook of Optimization in the Railway Industry*. Springer International Publishing, 2018. doi:10.1007/978-3-319-72153-8.
- 2 Stefan Engels and Robert Wille. Munich Train Control Toolkit. Software, swhId: swh:1:dir:2ffb77c8877a842cbc78f45016d4270dd5aeff5d (visited on 2026-09-22). URL: <https://github.com/cda-tum/mtct>, doi:10.4230/artifacts.28078.
- 3 Stefan Engels and Robert Wille. Comparing lazy constraint selection strategies in train routing with moving block control. In *Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS)*, volume 39, pages 585–590. Polish Information Processing Society, 2024. doi:10.15439/2024f3041.
- 4 Stefan Engels and Robert Wille. Using A* for optimal train routing on moving block systems. In *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2025, Warsaw, Poland, September 18-19, 2025*, volume 137, pages 14:1–14:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/OASICS.ATMOS.2025.14.
- 5 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968. doi:10.1109/TSSC.1968.300136.
- 6 Simon Hofmann, Marcel Walter, and Robert Wille. A* is born: Efficient and scalable physical design for field-coupled nanocomputing. In *2024 IEEE 24th International Conference on Nanotechnology (NANO)*, pages 80–85. IEEE, 2024. doi:10.1109/nano61778.2024.10628808.
- 7 Torsten Klug, Markus Reuther, and Thomas Schlechte. Does laziness pay off? - a lazy-constraint approach to timetabling. In Mattia D’Emidio and Niels Lindner, editors, *22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2022)*, volume 106. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/OASICS.ATMOS.2022.11.
- 8 Jörn Pachl. *Railway Signalling Principles: Edition 2.0*. Universitätsbibliothek Braunschweig, 2021. doi:10.24355/dbbs.084-202110181429-0.
- 9 Tom Peham, Judith Przigoda, Nils Przigoda, and Robert Wille. Optimal railway routing using virtual subsections. In *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, pages 63–79. Springer International Publishing, 2022. doi:10.1007/978-3-031-05814-1_5.
- 10 Ira Pohl. First results on the effect of error in heuristic search. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence 5*, pages 219–236. Edinburgh University Press, Edinburgh, 1970.
- 11 Thomas Schlechte, Ralf Borndörfer, Jonas Denißen, Simon Heller, Torsten Klug, Michael Küpper, Niels Lindner, Markus Reuther, Andreas Söhlke, and William Steadman. Timetable optimization for a moving block system. *Journal of Rail Transport Planning & Management*, 22:100315, 2022. doi:10.1016/j.jrtpm.2022.100315.
- 12 Lars Schnieder. *Communications-Based Train Control (CBTC)*. Springer Berlin Heidelberg, March 2021. doi:10.1007/978-3-662-62876-8.

A Raw Data

Instance	TA	ε	Single-Edge				Multi-Edge				Relevant-TTD						
			objective	gap [%]	time [s]	simulated	reexplored	objective	gap [%]	time [s]	simulated	reexplored	objective	gap [%]	time [s]	simulated	reexplored
High Speed Track (02 Trains)	no	1	648	0.00	< 0.01	5	0	648	0.00	< 0.01	5	0	648	0.00	< 0.01	5	0
	yes	1	648	0.00	< 0.01	3	0	648	0.00	< 0.01	3	0	648	0.00	< 0.01	3	0
	yes	5	-	-	-	-	-	-	-	-	-	-	648 (+0%)	0.00	< 0.01	3	0
High Speed Track (05 Trains)	no	1	792	0.00	0.01	81	0	792	0.00	0.01	81	0	792	0.00	0.01	81	0
	yes	1	792	0.00	< 0.01	6	0	792	0.00	< 0.01	6	0	792	0.00	< 0.01	6	0
	yes	5	-	-	-	-	-	-	-	-	-	-	792 (+0%)	0.00	< 0.01	6	0
Overtake	no	1	458	0.00	0.76	7958	5781	458	0.00	0.09	952	421	458	0.00	0.08	1010	171
	yes	1	458	0.00	0.01	119	0	458	0.00	< 0.01	53	1	458	0.00	< 0.01	56	2
	yes	5	-	-	-	-	-	-	-	-	-	-	500 (+0%)	9.00	< 0.01	19	2
Simple 2-Track Station	no	1	526	0.00	0.20	3337	1737	526	0.00	0.01	268	49	526	0.00	0.01	255	18
	yes	1	526	0.00	< 0.01	125	0	526	0.00	< 0.01	79	0	526	0.00	< 0.01	73	0
	yes	5	-	-	-	-	-	-	-	-	-	-	526 (+0%)	5.89	< 0.01	33	0
Simple Network	no	1	975	0.00	7.06	33206	24128	975	0.00	0.19	889	352	975	0.00	0.05	195	41
	yes	1	975	0.00	0.17	530	0	975	0.00	0.11	328	0	975	0.00	0.03	134	0
	yes	5	-	-	-	-	-	-	-	-	-	-	1873 (+92%)	47.9	< 0.01	33	0
Simple Network (03 Random Trains)	no	1	900	0.00	0.21	1196	637	900	0.00	0.03	138	40	900	0.00	0.01	63	7
	yes	1	900	0.00	0.01	53	0	900	0.00	< 0.01	33	0	900	0.00	< 0.01	37	0
	yes	5	-	-	-	-	-	-	-	-	-	-	1818 (+102%)	50.5	< 0.01	20	0
Simple Network (06 Random Trains)	no	1	1778	0.00	1632	390578	221637	1778	0.00	24.8	64987	19052	1778	0.00	9.45	23712	4009
	yes	1	1778	0.00	8.16	11309	0	1778	0.00	3.67	5270	0	1778	0.00	0.39	691	0
	yes	5	-	-	-	-	-	-	-	-	-	-	1778 (+0%)	12.2	0.18	312	0
Simple Network (09 Random Trains)	no	1	-	-	timeout	6803640	1721194	2131	0.00	903	1919196	371207	2131	0.00	341	736508	71154
	yes	1	2131	0.00	18.6	17104	0	2131	0.00	9.31	8649	0	2131	0.00	3.31	3343	0
	yes	5	-	-	-	-	-	-	-	-	-	-	2132 (+0%)	5.63	0.29	306	0
Simple Network (12 Random Trains)	no	1	-	-	timeout	7144204	1353229	-	-	timeout	3583778	644052	-	-	timeout	2558439	219886
	yes	1	3485	0.00	1569	900753	0	3485	0.00	745	467117	0	2974	0.00	129	87399	0
	yes	5	-	-	-	-	-	-	-	-	-	-	2988 (+0%)	26.1	1.68	1536	0
Simple Network (15 Random Trains)	no	1	-	-	timeout	6454866	990426	-	-	timeout	5216174	738078	-	-	timeout	3991428	239888
	yes	1	-	-	timeout	847619	0	-	-	timeout	828466	0	3004	0.00	378	290355	0
	yes	5	-	-	-	-	-	-	-	-	-	-	3223 (+7%)	17.0	16.3	4676	0
Simple Network (18 Random Trains)	no	1	-	-	timeout	6046797	581883	-	-	timeout	534558	0	-	-	timeout	4389981	299054
	yes	1	-	-	timeout	912273	0	-	-	timeout	806678	0	-	-	timeout	872613	0
	yes	5	-	-	-	-	-	-	-	-	-	-	3818	20.9	1.14	474	0
Single Track With Station	no	1	470	0.00	< 0.01	74	4	470	0.00	< 0.01	21	0	470	0.00	< 0.01	21	0
	yes	1	470	0.00	< 0.01	12	0	470	0.00	< 0.01	6	0	470	0.00	< 0.01	6	0
	yes	5	-	-	-	-	-	-	-	-	-	-	470 (+0%)	0.00	< 0.01	6	0
Single Track Without Station	no	1	648	0.00	< 0.01	5	0	648	0.00	< 0.01	5	0	648	0.00	< 0.01	5	0
	yes	1	648	0.00	< 0.01	3	0	648	0.00	< 0.01	3	0	648	0.00	< 0.01	3	0
	yes	5	-	-	-	-	-	-	-	-	-	-	648 (+0%)	0.00	< 0.01	3	0
Stammstrecke (01 Random Trains)	no	1	690	0.00	< 0.01	37	0	690	0.00	< 0.01	13	0	690	0.00	< 0.01	10	1
	yes	1	690	0.00	< 0.01	37	0	690	0.00	< 0.01	13	0	690	0.00	< 0.01	10	1
	yes	5	-	-	-	-	-	-	-	-	-	-	690 (+0%)	0.00	< 0.01	10	1
Stammstrecke (02 Random Trains)	no	1	1047	0.00	0.02	2699	1604	1047	0.00	0.15	692	275	1047	0.00	0.13	567	270
	yes	1	1047	0.00	0.02	106	0	1047	0.00	0.01	56	0	1047	0.00	0.01	54	8
	yes	5	-	-	-	-	-	-	-	-	-	-	1047 (+0%)	0.00	0.01	54	8

Instance	TA	ε	Single-Edge			Multi-Edge			Relevant-TTD			
			objective	gap [%]	time [s]	simulated	recexplored	objective	gap [%]	time [s]	simulated	recexplored
Stammstrecke (03 Random Trains)	no	1	999	0.00	20.3	66432	63526	999	0.00	2.73	8900	5173
	yes	1	999	0.00	0.06	147	-	999	0.00	0.02	80	0
	yes	5	-	-	-	-	-	999 (+0%)	0.00	-	-	-
	yes	10	-	-	-	-	-	999 (+0%)	0.00	-	-	-
Stammstrecke (04 Random Trains)	no	1	-	-	timeout	4157345	5658774	1077	0.00	44.1	100167	72883
	yes	1	1077	0.00	0.25	469	0	1077	0.00	0.10	198	0
	yes	5	-	-	-	-	-	1077 (+6%)	6.20	0.04	77	11
	yes	10	-	-	-	-	-	1077 (+6%)	6.20	0.04	77	11
Stammstrecke (04 Trains)	no	1	1142	0.00	1164	2413320	2843988	1142	0.00	71.8	150435	104013
	yes	1	1142	0.00	0.11	269	0	1142	0.00	0.06	112	0
	yes	5	-	-	-	-	-	1284 (+12%)	11.2	0.07	111	21
	yes	10	-	-	-	-	-	1284 (+12%)	11.2	0.07	111	21
Stammstrecke (05 Random Trains)	no	1	-	-	timeout	5664386	6619922	1004	0.00	314	709028	424182
	yes	1	1004	0.00	0.39	806	0	1004	0.00	0.21	389	0
	yes	5	-	-	-	-	-	1145 (+14%)	13.9	0.07	124	31
	yes	10	-	-	-	-	-	1145 (+14%)	15.9	0.07	124	31
Stammstrecke (06 Random Trains)	no	1	-	-	timeout	4099473	3930995	-	-	timeout	3529632	2535153
	yes	1	1208	0.00	0.94	1387	0	1208	0.00	0.47	688	0
	yes	5	-	-	-	-	-	1360 (+13%)	15.7	0.13	164	32
	yes	10	-	-	-	-	-	1360 (+13%)	15.7	0.12	164	32
Stammstrecke (07 Random Trains)	no	1	-	-	timeout	5091200	3306782	-	-	timeout	4102183	2484273
	yes	1	1125	0.00	0.66	956	0	1125	0.00	0.35	464	0
	yes	5	-	-	-	-	-	1253 (+11%)	13.4	0.13	164	26
	yes	10	-	-	-	-	-	1253 (+11%)	13.4	0.13	164	26
Stammstrecke (08 Random Trains)	no	1	-	-	timeout	6432961	3608147	-	-	timeout	3680650	1841907
	yes	1	1118	0.00	0.67	855	0	1118	0.00	0.34	403	0
	yes	5	-	-	-	-	-	1278 (+14%)	14.1	0.21	202	29
	yes	10	-	-	-	-	-	1278 (+14%)	14.1	0.22	202	29
Stammstrecke (08 Trains)	no	1	-	-	timeout	2525016	153747	-	-	timeout	2788064	1523112
	yes	1	1187	0.00	0.51	420	0	1187	0.00	0.27	228	0
	yes	5	-	-	-	-	-	1436 (+21%)	17.5	0.30	230	43
	yes	10	-	-	-	-	-	1436 (+21%)	17.5	0.29	230	43
Stammstrecke (09 Random Trains)	no	1	-	-	timeout	7315637	3245949	-	-	timeout	4612115	2085263
	yes	1	1098	0.00	1.24	1405	0	1098	0.00	0.61	688	0
	yes	5	-	-	-	-	-	1179 (+7%)	10.5	0.23	221	51
	yes	10	-	-	-	-	-	1301 (+18%)	18.8	0.25	225	53
Stammstrecke (10 Random Trains)	no	1	-	-	timeout	6824701	2484781	-	-	timeout	4788169	1965229
	yes	1	1264	0.00	6.64	5288	0	1264	0.00	3.21	2417	0
	yes	5	-	-	-	-	-	1407 (+16%)	19.1	0.37	243	46
	yes	10	-	-	-	-	-	1452 (+15%)	18.2	0.36	247	49
Stammstrecke (11 Random Trains)	no	1	-	-	timeout	6363719	2474871	-	-	timeout	4739460	1660505
	yes	1	1240	0.00	66.9	54523	0	1240	0.00	29.4	23504	0
	yes	5	-	-	-	-	-	1412 (+14%)	19.0	0.38	278	58
	yes	10	-	-	-	-	-	1345 (+8%)	15.0	0.37	278	58
Stammstrecke (12 Random Trains)	no	1	-	-	timeout	6881161	1516836	-	-	timeout	3386717	843681
	yes	1	1253	0.00	41.5	27533	0	1253	0.00	19.3	12328	0
	yes	5	-	-	-	-	-	1484 (+18%)	18.8	0.53	329	71
	yes	10	-	-	-	-	-	1484 (+18%)	18.8	0.53	329	71
Stammstrecke (13 Random Trains)	no	1	-	-	timeout	6933154	2452527	-	-	timeout	4855449	1191177
	yes	1	1329	0.00	59.8	36411	0	1329	0.00	29.9	17775	0
	yes	5	-	-	-	-	-	1452 (+9%)	12.0	0.72	368	81
	yes	10	-	-	-	-	-	1452 (+9%)	12.0	0.72	368	81
Stammstrecke (14 Random Trains)	no	1	-	-	timeout	4191645	1088269	-	-	timeout	3675403	940647
	yes	1	1321	0.00	92.2	40272	0	1321	0.00	37.9	20220	0
	yes	5	-	-	-	-	-	1416 (+7%)	10.7	0.68	349	58
	yes	10	-	-	-	-	-	1416 (+7%)	10.7	0.67	349	58

Instance	TA	ε	Single-Edge				Multi-Edge				Relevant-TTD						
			objective	gap [%]	time [s]	simulated	recexplored	objective	gap [%]	time [s]	simulated	recexplored	objective	gap [%]	time [s]	simulated	recexplored
Stammstrecke (15 Random Trains)	no	1	-	-	timeout	7077858	1337835	-	-	timeout	3238225	953080	-	-	timeout	3096845	1402920
	yes	1	1381	0.00	45.2	23246	0	1381	0.00	16.9	8562	0	1381	0.00	16.6	8015	1656
	yes	5	-	-	-	-	-	-	-	-	-	-	1513 (+10%)	10.5	0.81	335	62
	yes	10	-	-	-	-	-	-	-	-	-	-	1851 (+34%)	26.8	0.91	335	62
Stammstrecke (16 Random Trains)	no	1	-	-	timeout	7039892	1575351	-	-	timeout	3850785	666673	-	-	timeout	3878295	803242
	yes	1	1418	0.00	338	141626	0	1418	0.00	105	43343	0	1418	0.00	93.8	38655	4264
	yes	5	-	-	-	-	-	-	-	-	-	-	1689 (+19%)	19.6	0.98	354	52
	yes	10	-	-	-	-	-	-	-	-	-	-	1743 (+23%)	22.1	1.03	366	52
Stammstrecke (16 Trains)	no	1	-	-	timeout	2221055	644207	-	-	timeout	2141924	568399	-	-	timeout	2023531	738399
	yes	1	1383	0.00	2.95	949	0	1383	0.00	1.51	478	0	1383	0.00	1.35	488	97
	yes	5	-	-	-	-	-	-	-	-	-	-	1754 (+27%)	21.2	1.68	498	99
	yes	10	-	-	-	-	-	-	-	-	-	-	1763 (+28%)	21.6	1.74	486	93
Stammstrecke (17 Random Trains)	no	1	-	-	timeout	2963531	519844	-	-	timeout	3242234	656299	-	-	timeout	3127460	923632
	yes	1	-	-	-	-	-	1517	0.00	961	273975	-	1517	0.00	885	248421	32107
	yes	5	-	-	-	-	-	-	-	-	-	-	1745 (+15%)	18.7	1.46	425	57
	yes	10	-	-	-	-	-	-	-	-	-	-	1745 (+15%)	18.7	1.46	425	57
Stammstrecke (18 Random Trains)	no	1	-	-	timeout	5176165	933973	-	-	timeout	4620655	950424	-	-	timeout	3721597	1021678
	yes	1	-	-	timeout	745673	0	1563	0.00	1445	537518	0	1563	0.00	1316	475233	58127
	yes	5	-	-	-	-	-	-	-	-	-	-	1850 (+18%)	19.4	1.46	460	69
	yes	10	-	-	-	-	-	-	-	-	-	-	1974 (+26%)	24.4	1.44	456	69
Stammstrecke (19 Random Trains)	no	1	-	-	timeout	3948088	891329	-	-	timeout	2983874	621804	-	-	timeout	2910194	745411
	yes	1	1482	0.00	922	313367	0	1482	0.00	396	132275	0	1482	0.00	365	117502	16305
	yes	5	-	-	-	-	-	-	-	-	-	-	1674 (+13%)	14.0	1.39	463	73
	yes	10	-	-	-	-	-	-	-	-	-	-	1629 (+10%)	11.6	1.37	463	73
Stammstrecke (20 Random Trains)	no	1	-	-	timeout	5572922	1203984	-	-	timeout	3700443	532509	-	-	timeout	3436993	1050945
	yes	1	1452	0.00	611	213008	0	1452	0.00	271	92605	0	1452	0.00	261	88205	12584
	yes	5	-	-	-	-	-	-	-	-	-	-	1820 (+25%)	22.5	1.81	484	64
	yes	10	-	-	-	-	-	-	-	-	-	-	1785 (+23%)	20.9	1.76	478	64
Stammstrecke (21 Random Trains)	no	1	-	-	timeout	3377181	703534	-	-	timeout	3683337	539420	-	-	timeout	4214797	876181
	yes	1	-	-	timeout	520273	0	1624	0.00	854	220851	0	1624	0.00	796	216250	38871
	yes	5	-	-	-	-	-	-	-	-	-	-	1873 (+15%)	17.5	2.14	564	100
	yes	10	-	-	-	-	-	-	-	-	-	-	1959 (+21%)	21.2	2.25	568	102
Stammstrecke (22 Random Trains)	no	1	-	-	timeout	4434122	700002	-	-	timeout	3821167	637203	-	-	timeout	2935189	799919
	yes	1	1540	0.00	624	157447	0	1540	0.00	295	73885	0	1540	0.00	284	71149	13204
	yes	5	-	-	-	-	-	-	-	-	-	-	1703 (+11%)	12.4	2.53	608	97
	yes	10	-	-	-	-	-	-	-	-	-	-	1740 (+13%)	14.3	2.51	606	95
Stammstrecke (23 Random Trains)	no	1	-	-	timeout	5328391	577265	-	-	timeout	4583984	549486	-	-	timeout	4612401	897955
	yes	1	-	-	timeout	553884	0	-	-	timeout	476933	0	-	-	timeout	494822	99086
	yes	5	-	-	-	-	-	-	-	-	-	-	1764	10.3	2.85	684	146
	yes	10	-	-	-	-	-	-	-	-	-	-	1815	12.9	2.95	684	146
Stammstrecke (24 Random Trains)	no	1	-	-	timeout	2489865	406614	-	-	timeout	2413180	318846	-	-	timeout	2244219	460228
	yes	1	-	-	timeout	466052	0	-	-	timeout	382398	0	-	-	timeout	367264	71744
	yes	5	-	-	-	-	-	-	-	-	-	-	1835	13.3	2.62	618	113
	yes	10	-	-	-	-	-	-	-	-	-	-	1981	19.7	2.62	605	114
Stammstrecke (25 Random Trains)	no	1	-	-	timeout	4433915	582261	-	-	timeout	2862630	322555	-	-	timeout	3074197	523929
	yes	1	1519	0.00	1153	261561	0	1519	0.00	500	110481	0	1519	0.00	562	122062	23111
	yes	5	-	-	-	-	-	-	-	-	-	-	1653 (+49%)	10.9	3.01	654	106
	yes	10	-	-	-	-	-	-	-	-	-	-	1775 (+17%)	17.0	3.06	662	106
Stammstrecke (26 Random Trains)	no	1	-	-	timeout	3088864	301835	-	-	timeout	3906081	426125	-	-	timeout	2775562	517953
	yes	1	-	-	timeout	474812	0	-	-	timeout	407098	0	-	-	timeout	388653	43300
	yes	5	-	-	-	-	-	-	-	-	-	-	1935	15.3	3.28	598	66
	yes	10	-	-	-	-	-	-	-	-	-	-	1934	15.3	3.30	600	66
Stammstrecke (27 Random Trains)	no	1	-	-	timeout	4331628	462513	-	-	timeout	4019953	517792	-	-	timeout	3304114	568626
	yes	1	-	-	timeout	715059	0	-	-	timeout	583854	0	-	-	timeout	578196	136161
	yes	5	-	-	-	-	-	-	-	-	-	-	2008	14.7	4.40	796	177
	yes	10	-	-	-	-	-	-	-	-	-	-	2043	16.1	4.80	800	181

Why Start Over: Incremental MaxSMT for Operational Railway Scheduling

Thomas Dubach   

IVT Transport Systems Group, ETH Zurich, Switzerland

Florian Flükiger-Fuchs  

IVT Transport Systems Group, ETH Zurich, Switzerland

Bernardo Martin-Iradi   

IVT Transport Systems Group, ETH Zurich, Switzerland

Abstract

Railway timetables are planned periodically, but operational planners continuously need to deviate from this base: extra trains are added, or infrastructure changes are made, requiring local updates to the existing schedule. We study this process as a sequential extra train scheduling problem, in which requests arrive one at a time and must be routed and timed against the existing timetable while minimising deviation from it. Commonly, each admitted train is fixed before the next request arrives, and all accumulated solver knowledge is discarded between calls.

We propose an incremental MaxSAT framework over Integer Difference Logic (MaxSMT) that retains accumulated solver state across successive extra-train additions, rather than solving from scratch after each request. We evaluate the approach on the Rhätische Bahn network using different solving strategies and compare against a MIP baseline. Retained solver state substantially reduces solve time for successive additions, increasingly so as more extra trains are admitted, and is an order of magnitude faster than MIP given the same setup. Beyond solving time, we show that jointly re-optimising over all admitted extra trains yields better cumulative objective values, and that our approach achieves this while prioritising adjustments in the timetable region affected by the new request.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Applied computing → Transportation; Theory of computation → Integer programming

Keywords and phrases MaxSAT, operational railway timetabling, incremental solving

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.4

Funding *Thomas Dubach*: ETH Mobility Initiative – Project LeRaBe

1 Introduction

Periodic timetables are the backbone of modern railway operations. By repeating at regular intervals, they support efficient vehicle and crew rostering, consistent passenger information, and the predictability on which connecting services depend [21, 15]. Yet the periodic structure is an idealisation: in practice, railway operations continuously require local deviations from the periodic base. Extra services are added to accommodate seasonal demand, or infrastructure works impose temporary capacity restrictions. These adaptations are a routine part of the operational planning process, which runs from the moment the yearly timetable is fixed until the operating day [1, 3].

This paper addresses one part of this *operational timetabling* process: given an existing periodic timetable and a sequence of extra train requests, produce a locally adapted aperiodic schedule that accommodates each request while minimising aggregate delays from the periodic base. Each request must be routed and timed on a microscopic infrastructure model without creating resource occupation overlaps with the context trains or with previously admitted extra trains.



© Thomas Dubach, Florian Flükiger-Fuchs, and Bernardo Martin-Iradi;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 4; pp. 4:1–4:19



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This operational planning process has a structure that existing solution methods do not exploit. Requests arrive and are processed sequentially, each producing a locally revised schedule that serves as the reference for the next. When a new request concerns a region of the network already affected by earlier requests, the knowledge developed while solving those earlier problems remains relevant: the infrastructure has not changed, and the combinations of scheduling decisions that were infeasible before are still infeasible now. A solver that is reset between successive requests must rediscover this knowledge from scratch; a solver that retains it can apply it directly to the new problem.

We exploit this structure through an incremental Maximum Satisfiability (MaxSAT) framework over Integer Difference Logic (IDL), which we will call MaxSMT for the remainder of this paper. Building on the SAT+IDL architecture of [14], we add a MaxSAT optimisation layer that minimises timetable delays. The solver is never reset between successive extra train additions. Instead, it carries over its accumulated state: (1) the learnt cuts, which record which combinations of scheduling decisions are proven infeasible, (2) its lower bound proof, which certifies the minimum objective cost reached so far, and (3) the branching decision heuristic, which orients the search toward solutions close to the one previously found.

MIP warm-starting differs: a standard MIP solver can carry over incumbent solutions between solves, but it discards its learnt cuts, its dual bound proof, and the branch-and-bound search tree on each invocation.

The paper makes the following contributions:

- A MaxSMT formulation for operational timetabling on a microscopic infrastructure model, combining Integer Difference Logic for continuous event-time variables with soft clauses penalising delays from a periodic reference timetable.
- An incremental solving architecture in which the full accumulated solver state is retained across successive extra train additions rather than discarded between solves.
- Empirical evidence that retained solver state in MaxSMT substantially reduces median total solve time, with the gain growing as the number of additions increases, while remaining an order of magnitude faster than a state-of-the-art commercial solver under the same solving strategy.
- A finding that allowing joint re-optimisation over the full set of extra train requests, rather than committing to the schedule of each train after a request, yields strictly better cumulative objective values, and that the MaxSMT solver achieves this without sacrificing its speed advantage.
- A finding that the MaxSMT solver, guided by retained search state from the prior solve, tends to concentrate revisions in the timetable region affected by each new request.

The remainder of the paper is structured as follows. Section 2 reviews the relevant literature on operational timetabling and MaxSAT in timetabling, then identifies the research gap this work addresses. Section 3 describes the event-activity model and the MaxSMT encoding. Section 4 presents the incremental solving architecture and the solving strategies evaluated. Section 5 describes the experimental setup on the Rhätische Bahn network, a metre-gauge railway in south-eastern Switzerland. Section 6 reports and analyses the results.

2 Literature Review

We begin by establishing the incremental process of the operational railway planning and situating the present work within the existing timetabling literature. We then review MaxSAT approaches to train scheduling. We conclude by identifying the gap that the present work addresses.

2.1 Operational Timetabling as an Incremental Process

In practice, railway timetables are not constructed in a single pass but emerge through successive refinement across planning horizons of decreasing length and increasing detail [1, 2, 3]. At each stage, the output of the previous stage serves as a starting point that is locally adapted rather than reconstructed from scratch. At the operational layer, this means adjusting an existing timetable to accommodate new constraints or requests (e.g. non-periodic extra services, track possessions, or changed demand) while minimising delays from the base schedule. Each new request triggers a local revision of the schedule constructed so far, making the operational planning process inherently incremental.

Several papers have studied this adjustment problem; we discuss the two most directly relevant. Kloster et al. [12] study an interactive planning workflow in which a route planner makes successive modifications to a yearly timetable, such as adding an extra train service, and calls a solver after each change to restore conflict-freeness. Van Aken et al. [23] address the Train Timetable Adjustment Problem (TTAP): given an existing timetable and a set of planned infrastructure possessions, find an adjusted timetable that minimises cancellations and deviations from the original. Both papers share the delay-minimisation objective that the present work also adopts.

Kloster et al. solve a MIP with lazy constraint generation, operating on a mesoscopic model in which the periodic timetable is projected onto a single operating day; solver state is maintained as constraints are added during lazy generation, but is discarded between successive planning calls. Van Aken et al. extend the PESP formulation with cancellation variables and a row-generation approach for station capacity, operating on a macroscopic network model. Their approach starts with an initial periodic solution and produces a single adjusted timetable that must also be periodic.

2.2 MaxSAT in Timetabling

A growing body of work has established MaxSAT as a competitive solution method for railway timetabling problems where the objective is to find a conflict-free schedule minimising delay from a reference or ideal timetable. Croella et al. [5] apply MaxSAT via the RC2 algorithm to a Dynamic Discretisation Discovery model for real-time train rescheduling. Lemos et al. [13] develop an iterative MaxSAT encoding for scheduling under disruptions. Gouveia et al. [10] address a non-periodic timetabling variant that combines route choice, train cancellations, and delays from an ideal timetable. Matos et al. [18] combine MaxSAT with reinforcement learning to optimise periodic timetables. Fuchs et al. [9] use a binary-search MaxSMT approach to optimize the stability in periodic timetables.

Across this body of work, incremental reuse of solver state takes two forms. Within a single optimisation run, Croella et al. design their variable encoding so that refining the time discretisation only adds new constraints and never invalidates existing ones, which makes the accumulated clause database directly reusable across iterations. Across separate solve invocations for adapted problem instances, only Lemos et al. investigate explicit state retention: they carry the search state into each re-solve triggered by a new disruption, and report an average 25 % speed-up. Unlike Croella et al., however, the search space expansion method they use to handle problem scale arising from discretisation is incompatible with solver state reuse.

2.3 Research Gap

This paper targets the operational layer: given a valid periodic timetable and a sequence of extra train requests, the task is to produce a locally adapted aperiodic schedule that accommodates each request while minimising the delay from the periodic base. The literature reviewed above leaves this problem unaddressed from two complementary directions.

The papers closest to this work at the *problem level* [12, 23] treat each planning call as an independent optimisation: previously adjusted trains are fixed before each new solve, and all solver state is discarded between calls. For Kloster et al., this is an inherent consequence of lazy constraint generation: to handle a request in a different region of the timetable, the solver must restart from scratch to discover the constraints relevant to that region, so no conflict structure accumulated in one call can transfer to the next. Neither paper exploits the incremental structure of the operational planning process at the solver level.

The papers closest to this work at the *methodology level* [5, 13, 10, 18, 9] do not demonstrate the benefits of retaining solver state across a growing sequence of problem additions under an encoding that is complete from the outset. Lemos et al. [13] retain state across re-solves but cannot do so under the search space expansion they require to handle problem scale. The incremental MaxSAT literature establishes that state retention across related instances can yield measurable performance gains [20, 22], but this has not been shown on a practical combinatorial optimisation problem such as operational timetabling.

The present work addresses both gaps simultaneously by encoding the full constraint structure within the MaxSMT solver from the outset, on a microscopic infrastructure model. Because the encoding is complete, learnt cuts accumulate across the entire constraint space with each solve, and successive extra train additions can draw on this full body of accumulated knowledge. We demonstrate empirically that retained solver state outperforms both a warm-start MIP and a cold-start MaxSMT baseline.

3 Problem and Model

We consider the problem of scheduling a set of *extra train requests* against an existing periodic microscopic timetable. A valid periodic timetable for a single operating day is given as input: as supplied, its trains occupy the infrastructure without conflict, and their published arrival times define the reference from which delays are penalised. We refer to these as *context* trains. Context trains are not held to the input schedule: their routes through the microscopic infrastructure may be reassigned, while keeping the sequence of stations served fixed. The event times may move within hard windows around the published arrival and departure times. Departures may only be delayed since passengers plan around them, while arrivals may shift in either direction. Because routes and event times may change, resource occupation must be resolved anew across all trains whenever the timetable is revised, and the conflict-freeness of the input does not carry over.

A set of extra train requests is to be scheduled on top of this context. Each request specifies a sequence of stations together with a desired departure and arrival time at each serviced station; the extra train must be routed through the network so that it serves the requested stations in the given order, without conflicting with context trains or with other extra trains already admitted to the schedule. Its time windows are considerably wider than those of the context trains, since an extra service carries no published time that passengers already rely on. The planning task is to find, for every train, context and extra alike, a route and a set of event times that are jointly feasible, while minimising the aggregate delays from the reference arrival times.

3.1 Event-Activity Model

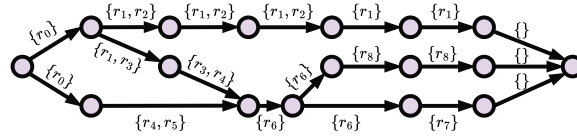
We model the train scheduling problem using the *Event-Activity Network (EAN)* framework; we refer to existing work [8] for the full derivation and give only a brief account here.

The network is a directed graph $G = (\mathcal{E}, \mathcal{A})$ whose components correspond to individual trains, with nodes and edges defined as follows:

Events $e \in \mathcal{E}$ represent scheduled timestamps such as arrivals, departures, or passing times at infrastructure elements. Each event e carries a timing variable $t_e \geq 0$ and hard time bounds $lb_e \leq t_e \leq ub_e$.

Activities $a = (e, e') \in \mathcal{A}$ are directed edges representing precedence relations between pairs of events belonging to a single train, such as running times over track sections or dwell times at stations. Each activity imposes a duration constraint $lb_a \leq t_{e'} - t_e \leq ub_a$ and is associated with a set of resources $R(a) \subseteq \mathcal{R}$. A resource $r \in R(a)$ is occupied during the interval $[t_e, t_{e'} + \delta(r, a)]$, where $\delta(r, a)$ is a release time that enforces the required separation between successive trains. The same resource can appear in the resource sets of multiple activities, whenever those activities must reserve the same physical infrastructure element.

An example EAN for a single train is shown in Figure 1. A feasible solution requires: *route selection*, a single source-to-sink path for each train; *temporal feasibility*, event timings respecting all event and activity bounds along the selected path; and *resource conflict resolution*, no resource occupied by more than one train at any point in time.



■ **Figure 1** The EAN for a single train [8]. Events (nodes) and activities (links) carry time bounds as described in the main text; each activity's resource set $R(a)$ is shown alongside it.

3.2 Encoding with SMT

All constraints in the event-activity model can be reformulated as either clauses over Boolean variables or difference constraints of the form $t_j - t_i \geq l_{ij}$, the latter activated conditionally on the route selection; we refer to [8, 9] for the complete formulation. Event times are integer variables in units of 1 s.

The IDL theory represents the active difference constraints as a directed weighted graph [4]: each timing variable t_i is a node, and each active constraint $t_j - t_i \geq l_{ij}$ adds a directed edge from t_j to t_i with weight l_{ij} . A partial assignment is timing-feasible if and only if this graph contains no positive-length cycle. The theory propagator detects such cycles incrementally as the SAT solver makes Boolean assignments; when a cycle is found, the activation literals of its constituent edges form a *theory lemma*, a clause asserting that this combination of active constraints is infeasible, which is added to the SAT solver's clause database and guides the subsequent search.

4 Algorithm

The central contribution is an incremental solving architecture in which the accumulated MaxSMT solver state is retained and reused across successive extra train additions. As background, we first introduce the core-guided MaxSAT algorithm that underlies the optimisation layer in Section 4.1. Then, Section 4.2 describes the encoding of the delay objective over IDL timing variables; Section 4.3 presents the architecture and the soundness argument for state retention; Section 4.4 defines the six solving strategies compared empirically in Section 6.

4.1 Core-Guided MaxSAT

On top of the SAT+IDL solver we build an optimisation layer using the RC2 algorithm [11], a core-guided MaxSAT solver [19, 17]. MaxSAT partitions constraints into *hard clauses*, which must hold in every solution, and *soft clauses*, each carrying a weight equal to the cost of its violation. The objective is to find a satisfying assignment to the hard clauses that minimises the total weight of violated soft clauses.

RC2 iteratively calls the SAT solver. All soft clauses are initially assumed to hold. If this is infeasible, the SAT solver returns an *UNSAT core*: a subset of soft clauses that cannot simultaneously hold together with the hard clauses. The minimum weight among the clauses in the core is a certificate that the objective must increase by at least that amount, thereby tightening the lower bound. RC2 then encodes the relaxation of the core as a cardinality constraint which asserts that at least one clause in the core may be violated, and permanently adds it to the clause database. Then it probes the solver again at the updated bound. This repeats until the solver returns a feasible solution, which is then provably optimal.

4.2 Timing Objective in MaxSAT

The combination of a MaxSAT solver with an IDL theory lets us encode an objective function directly over the continuous timing variables in the theory, rather than solely over Boolean variables. Thus, though the objective function remains discrete, scheduling itself can be done to the precision-level of the IDL theory. To our knowledge, this combination has not previously been applied in railway scheduling.

The objective minimises the weighted sum of delays from reference times: for context trains, delay from their published arrival times; for extra trains, delay from the preferred arrival time at each serviced station. Both are encoded identically. For each objective event e with timing variable t_e and reference time t_e^* , we introduce n Boolean activation literals $l_1^e, \dots, l_n^e$ together with the following conditionally activated difference constraints:

$$l_k^e \rightarrow t_e \leq t_e^* + k \cdot 60 \text{ s}, \quad k = 1, \dots, n.$$

Each literal l_k^e is added as a soft clause carrying weight 60, so violating it contributes one minute to the objective. The encoding is tightened by adding the implication chain

$$l_k^e \rightarrow l_{k+1}^e, \quad k = 1, \dots, n-1,$$

as hard clauses. The chain is valid because $t_e \leq t_e^* + k \cdot 60 \text{ s}$ implies $t_e \leq t_e^* + (k+1) \cdot 60 \text{ s}$, so setting any literal to true forces all later ones to the same value.

All soft clauses carry the same weight, across events and across the delay minutes of a single event, so the objective is step-wise linear in total delay and indifferent to its distribution; different weights would express other distributional preferences without altering the encoding.

The granularity of 60s per soft clause matches the precision of the published timetable: delays below one minute are not observable to passengers or reflected on public departure boards, so the first minute of delay incurs no penalty. Granularity also controls the encoding size: a finer discretisation would produce more accurate cost distinctions but also lead to a proportionally larger clause database. 60s balances these considerations and matches the objective discretisation used in prior MaxSAT timetabling work [5].

4.3 Incremental Architecture

When a new extra train request is added to an existing feasible schedule, the new solve begins from the full state accumulated during the previous one, rather than discarding it. In the MaxSMT framework, four components make up this retained state:

Conflict clauses. When the SAT solver reaches a contradictory variable assignment, it analyses the chain of decisions that led to the contradiction and derives a *conflict clause*: a constraint that rules out that combination of assignments in any future search.

Theory lemmas. The clauses generated by the IDL propagator when it detects a positive-length cycle in the constraint graph (Section 3.2). They are analogous to conflict clauses, but encode infeasibility of timing combinations rather than propositional contradictions.

Core clauses. When RC2 identifies an UNSAT core among the soft clauses, it encodes the resolution of that core as a cardinality constraint and adds it permanently to the clause database.

Search statistics. The SAT solver maintains an activity score for each variable that reflects how often it appeared in recent conflicts, and uses these scores to decide which variable to branch on next. It also records the last assignment used for each variable. When the solver is not reset, both the activity scores and the saved assignments carry over: the solver begins the new search already biased toward the variables and assignments that were informative in the previous one.

The reason all four components remain valid across solving iterations is that the instance only becomes more constrained between successive solves: each new extra train addition introduces new variables and new constraints, but never removes or relaxes any constraint present in a prior iteration. This monotone strengthening property guarantees soundness. Conflict clauses and theory lemmas are derived by resolution and cycle detection, respectively; both are valid for any instance that includes the constraints used to derive them, and since those constraints are never removed, both remain valid after strengthening. A core witnesses a lower bound on the objective. When the constraints are tightened, the set of satisfying assignments can only shrink, so any previously identified core remains infeasible and its associated lower bound remains valid. Finally, search statistics are heuristic: the solver remains complete regardless of the activity scores and saved variable values it inherits, so carrying them over can only affect efficiency, not correctness.

None of these components is novel in isolation; the contribution lies in their composition within the operational timetabling setting, where the instance changes only locally between successive solves. The claim is that clauses and cores discovered while scheduling earlier extra trains recur when scheduling later ones, and need not be re-derived from scratch.

Here, MIP warm-starting differs. Gurobi retains incumbent solutions between solves but keeps neither the branch-and-bound tree nor any learned cuts, so both the knowledge about infeasibility and the progress toward proving the lower bound are lost on each invocation. The MaxSMT solver retains both, which is the basis for the empirical comparison in Section 6.

■ **Table 1** The four sequential solving strategies, formed by combining the treatment of previously admitted extra trains with the treatment of the solver state between solves. Flexible-persistent is our proposed method; Commit-reset most closely matches the workflow of Kloster et al. [12].

		Solver state between solves	
		Retained	Discarded
Admitted extra trains	Remain free	Flexible-persistent	Flexible-reset
	Committed	Commit-persistent	Commit-reset

4.4 Solving Strategies

Extra train requests may arrive all at once or one at a time, both being operationally realistic. When requests arrive sequentially, there are two independent choices. The first is a problem-level choice: whether an extra train, once scheduled, remains completely free to be rescheduled when later requests arrive (*Flexible*), or is committed and thereafter treated exactly as a context train (*Commit*) keeping route flexibility but with reduced time windows. The second is a solver-level choice: whether the accumulated solver state is retained across solves (*persistent*), as described in Section 4.3, or fully discarded before each solve (*reset*). Combining the two axes gives the four sequential strategies in Table 1.

In addition, we evaluate two *Batch* strategies in which all requests are encoded simultaneously and solved in a single invocation. In the *all-extra-trains* variant, the solver is warm-started from the periodic solution. In the *one-shot* variant, the full periodic timetable is encoded together with all requests and solved from scratch.

The Flexible and Batch strategies yield the same globally optimal objective value: because all extra trains remain free throughout, each solve jointly optimises over the full set of requests, regardless of delivery mode or state-management choice. The Commit strategies trade this guarantee for a smaller search space. Committing each newly scheduled train before the next request arrives removes degrees of freedom and reduces solve time, but forfeits the ability to make joint adjustments when a later request would benefit from revising an earlier decision, potentially leading to a worse objective value.

5 Experimental Setup

All MaxSMT experiments use a custom implementation of RC2 [11] on top of Mini-Card v1.2 [16, 6] extended with IDL theory propagation [14]. The MIP baseline uses Gurobi v13.0.2. The methods are used within the OpenBus toolbox [7]. Both run on a server using 4 cores of an AMD EPYC 7763 CPU allocated 16 GB of RAM. Each experiment is repeated multiple times; run-to-run variation arises from non-deterministic encoding order and server load rather than from solver randomness, as MaxSMT runs deterministically.

The comparison is set up to favour the MIP wherever a choice arises. We use the dedicated formulation of the event-activity model of Fuchs et al. [8], developed for this problem class and competitive on instances of comparable size. Both solvers receive the same problem data, preprocessing, and model strengthenings from that work, and the MIP additionally uses all four cores, whereas MaxSMT is single-threaded. Because Gurobi has been shown to handle discretised objectives in railway scheduling less well than a MaxSAT solver [5], the MIP is also given a continuous relaxation of the discrete objective function: it halts as soon as it finds a feasible solution whose continuous objective value is at most the optimal discretised MaxSMT objective, and therefore never certifies optimality of the discretised problem.

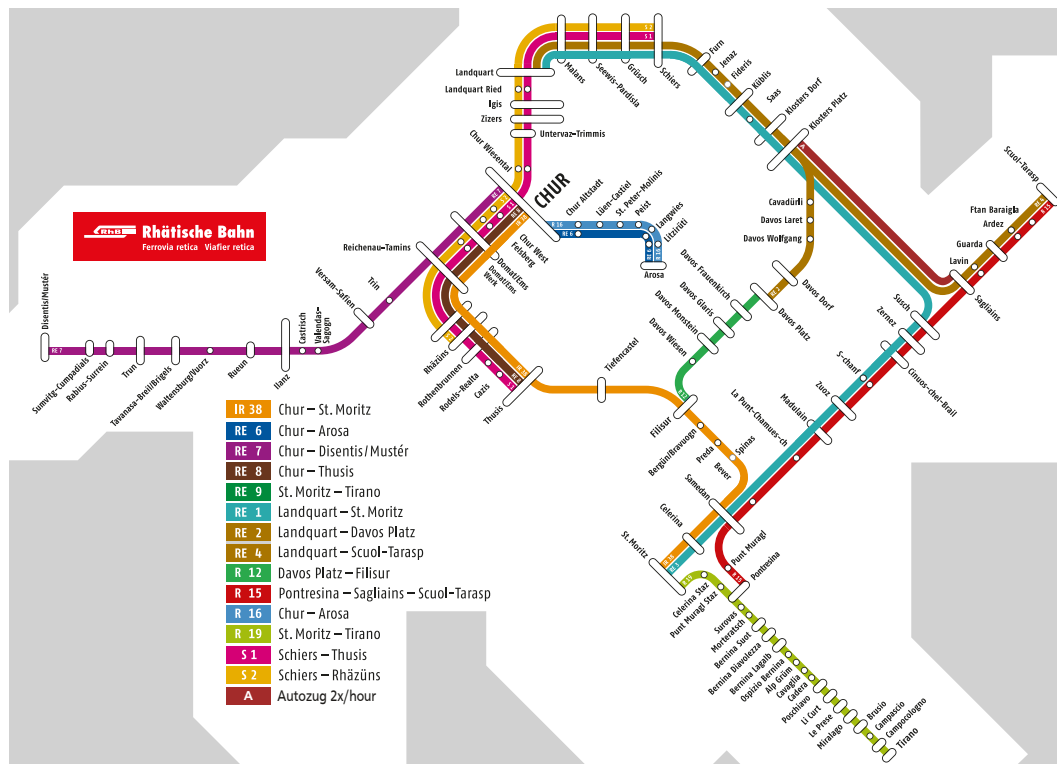


Figure 2 Services in the RhB network. The 14 passenger services run bi-directionally once an hour; the motorail (Autozug) runs twice an hour in each direction. The 240 trains make up the context trains for the experiments.¹

Network and Context Trains. The experiments use infrastructure from the Rhätische Bahn (RhB), a metre-gauge network in the canton of Graubünden, Switzerland, shown in Figure 2. A perfectly periodic solution for the base timetable is provided as input: each train in the timetable is scheduled once an hour in each direction over 8 hours, with the schedule being periodic in both route choice and timings.

The arrival times at stations define the reference from which delays are penalised. Context trains may deviate from their published times within a window of $[0, +5 \text{ min}]$ at station departures and $[-5 \text{ min}, +5 \text{ min}]$ at station arrivals; windows at intermediate events are propagated from these bounds, given minimal travel times [8].

For tractability and interpretability, Sections 6.1–6.3 use only the Chur–Arosa section: the two lines (RE6, R16) visible in the centre of Figure 2, consisting of 8 passing loops connected by single-track corridors. The infrastructure is small enough to visualise the timetable in full, while the single-track structure and two trains per hour per direction produce a high density of resource occupation that prevents trivial scheduling of additional services. Section 6.4 extends to the full RhB network to assess scalability.

Extra Train Requests. Extra train requests are generated by sampling a desired departure time uniformly at random from the 8-hour window and choosing a direction of travel at random. For the Chur–Arosa experiment, all 10 requests are for the same service type: a

¹ Reproduced from: <https://www.rhb.ch/de/streckennetz/>

■ **Table 2** Instance dimensions for the two experimental networks. *Base* is the periodic timetable before any extra train is added; *final* is the instance after all extra train requests have been admitted. Routing decisions count the Boolean route-branching variables; conflicts count pairs of activities competing for the same resource, each introducing an ordering decision.

Network	Trains		Events		Routing dec.		Conflicts	
	Base	Final	Base	Final	Base	Final	Base	Final
Chur–Arosa	32	42	6432	8012	1696	2226	3976	33 255
Full RhB	224	254	110 688	118 613	19 392	21 138	98 035	242 755
<i>Added per extra train request (min–max)</i>								
Chur–Arosa			158		53		837–4339	
Full RhB			125–409		36–91		51–13970	

train with one intermediate stop making three scheduled calls in total. For the full-network experiment, 30 requests are generated across 5 service types, with the service type chosen at random for each request, along with the direction and desired departure time. In both cases, the full set of requests, including their order of addition, is sampled once and held fixed across all runs and all solver configurations, so that differences in solve time reflect solver behaviour rather than instance variability.

Each request specifies a sequence of stations and a desired departure and arrival time at each serviced station, with a time window of $[-3 \text{ min}, +90 \text{ min}]$ around the desired time. The desired arrival times are derived from the sampled departures based on the minimal dwell time at the station. In the Flexible strategies, these windows apply to every admitted extra train throughout, so a train scheduled at an earlier step can still be re-timed anywhere within its original window; in the Commit strategies, each newly scheduled is treated as a context train in subsequent solves, with the context-train windows above applied around the times just scheduled for it. Its reference times in the objective remain those of the original request, so the cumulative objective values reported for the Flexible and Commit strategies measure the same quantity.

Instance Size. Table 2 reports the dimensions of both instances. The full RhB instance contains 254 trains once all requests are admitted, against 160 trains with distinct schedules in Kloster et al. [12] and 135 train lines in the largest instance of Van Aken et al. [23]. Both papers work on mesoscopic and macroscopic models respectively, so the number of events, routing decisions, and resource conflicts per train is considerably smaller than under the microscopic model used here: their largest reported instance comprises 4411 events, where ours has 118 613 events. Though not reported in the two papers, the number of routing decisions and resource conflicts will scale accordingly.

6 Results

We show in Section 6.1 that state retention reduces solve time for both solvers, but not in the same way: MaxSMT benefits consistently across steps, whereas the MIP advantage erodes at the hardest steps, where the warm-start incumbent diverges from optimality. Section 6.2 shows that leaving the time windows of previously scheduled extra trains free between solves yields better cumulative objective values than when reducing them. Section 6.3 shows that MaxSMT reaches these objective values while leaving most of the existing timetable unchanged. Section 6.4 confirms that the conclusions hold on the full RhB network.

■ **Table 3** Median total solve time in seconds and factor relative to the baseline (Flexible-persistent, MaxSMT; 9.3 s), across $n = 50$ runs per condition. *Flexible* keeps all admitted extra trains free, while *Commit* makes each newly scheduled extra train a context train before the next solve; *persistent* retains the solver state between steps, *reset* discards it. *Batch: all-extra-trains* warm-starts from the periodic solution and solves all requests at once. *Batch: one-shot* encodes the full periodic timetable together with all requests and solves once.

Strategy	MaxSMT		MIP (Gurobi)	
	Median (s)	Factor	Median (s)	Factor
Flexible-persistent	9.3	1.0×	559.4	60.2×
Flexible-reset	44.3	4.8×	1742.3	187.3×
Commit-persistent	3.1	0.33×	44.5	4.8×
Commit-reset	5.0	0.54×	36.9	4.0×
Batch: all-extra-trains	14.5	1.6×	577.7	62.1×
Batch: one-shot	14.7	1.6×	1425.8	153.3×

6.1 Solve Time: Persistent vs. Reset

We compare the Flexible-persistent and Flexible-reset strategies for both solvers and report Batch results for comparison with sequential solving. Per-step solve times for the Flexible strategies are shown in Figure 3; aggregate timings for all strategies appear in Table 3.

For MaxSMT, retaining solver state reduces the median total time by a factor of around $5\times$ relative to the Flexible-reset strategy. The benefit is not uniform across steps: early steps, where little state has accumulated, show smaller gains; later steps, where the accumulated conflict structure is richer, show larger ones. This is consistent with the state-retention argument in Section 4.3: the value of retained conflict clauses, theory lemmas, and RC2 core constraints increases as the schedule becomes denser with each addition.

For MIP, the persistent strategy is around $3\times$ faster than resetting the solver state. The MIP therefore also benefits from state retention, though not robustly across all steps. In steps 7 and 9 (the hardest steps as the problem has grown), the benefits of warm-starting the MIP are relatively smaller: the median persistent solve time at step 7 is 29.8 s against 32.4 s for reset, and at step 9, 513.5 s against 570.2 s. These two steps dominate the MIP aggregate solve time and erode the advantage of state retention.

The Batch strategies confirm that the advantage of state retention is not an artefact of sequential delivery. The all-extra-trains batch, which warm-starts from the periodic solution and encodes all requests simultaneously, takes 14.5 s for MaxSMT and 577.7 s for MIP. The Flexible-persistent strategy for MaxSMT is thus measurably faster than Batch, whereas Flexible-persistent MIP and Batch MIP are roughly comparable. Flexible-persistent MaxSMT therefore outperforms both its MIP counterpart and its own Batch baseline.

6.2 Solution Quality: Flexible vs. Commit

Committing previously scheduled extra trains between steps reduces the degrees of freedom in each subsequent solve, and one would therefore expect it to reduce solve time. The per-step solve times for the Commit-persistent and Commit-reset strategies are shown in Figure 4, and the last two rows of Table 3 report the aggregate figures.

The solve-time reduction from committing is substantial for both solvers. Comparing Commit-persistent with Flexible-persistent with MaxSMT, the median total time drops from 9.3 s to 3.1 s, a factor of approximately $3\times$. Comparing the same for the MIP, the reduction

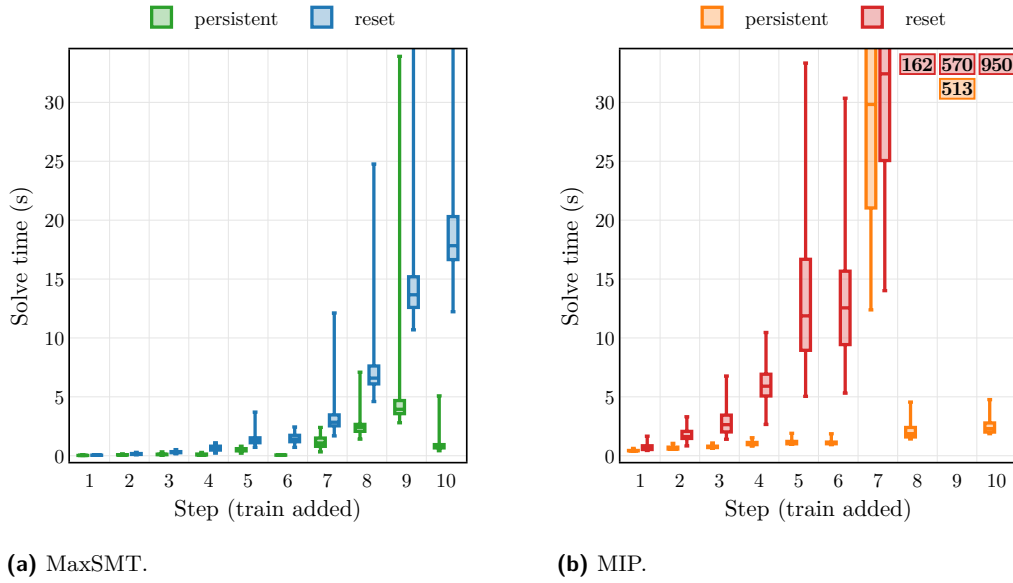


Figure 3 Per-step solve time for the Flexible-persistent and Flexible-reset strategies, for MaxSMT (left) and MIP (right), across $n = 50$ runs. Boxes show the median and interquartile range; whiskers extend to the minimum and maximum. Off-the-chart median values are reported in the boxes at the top of the chart.

Table 4 Cumulative objective value (total delay, in minutes) for the Flexible and Commit strategies. Steps 1–8 are identical; the Commit strategies incur a penalty of 26 min at step 9, which persists to step 10.

Strategy	Step (train added)									
	1	2	3	4	5	6	7	8	9	10
Flexible (min)	1	8	20	26	37	39	53	104	188	219
Commit (min)	1	8	20	26	37	39	53	104	214	245
Gap (min)	0	0	0	0	0	0	0	0	26	26

is larger: from 559.4s to 44.5s, a factor of approximately $12\times$. The MIP Commit-reset strategy (36.9s) is slightly faster than Commit-persistent (44.5s). This reverses the pattern seen in the Flexible case and shows that warm-starting the MIP can become an impediment.

The speed gain from committing comes at a cost to solution quality. Because a committed train cannot be retimed as freely when a new request arrives, the solver loses the ability to make joint adjustments that would reduce the aggregate delay. The effect is visible in Table 4: the cumulative objective values for the Commit and Flexible strategies are identical through step 8, but at step 9 the Commit strategies incur an additional penalty of 26 minutes, which persists through step 10.

6.3 Solution Structure: Locality Bias in MaxSMT

Beyond aggregate solve times and objective values, the two solvers differ in which parts of the timetable they modify when a new extra train request arrives. Figure 5 shows the timetable produced after the seventh step overlaid on the timetable from the sixth step (in grey) for MaxSMT and MIP, side by side, in the Flexible-persistent strategy.

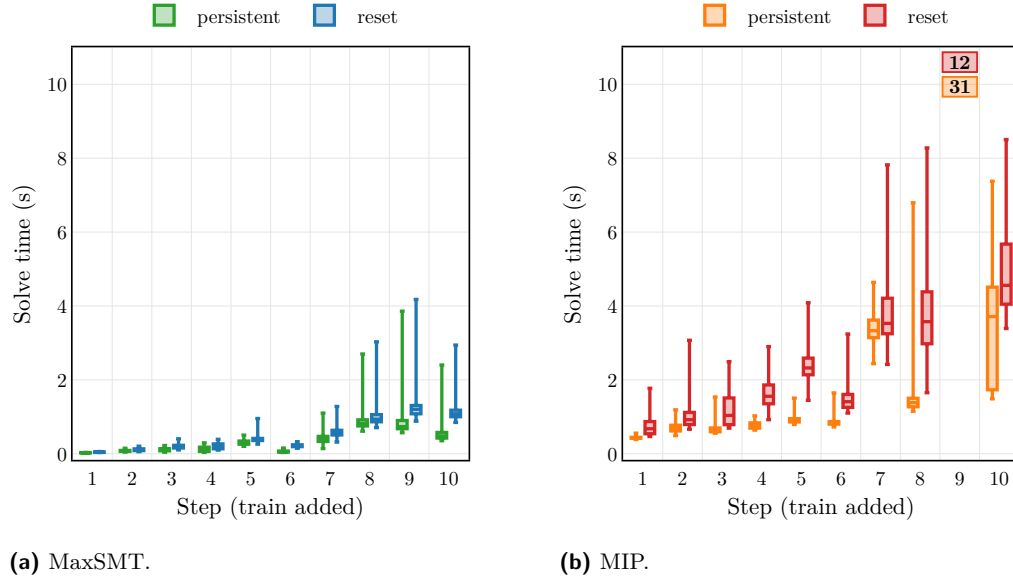


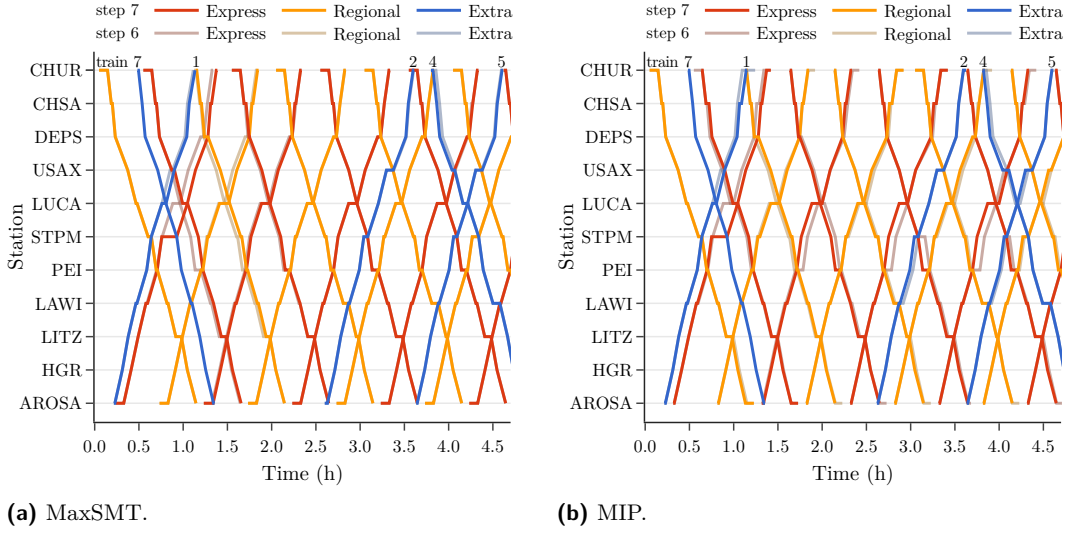
Figure 4 Per-step solve time for the Commit-persistent and Commit-reset strategies, for MaxSMT (left) and MIP (right), across $n = 50$ runs. Boxes show the median and interquartile range; whiskers extend to the minimum and maximum. Off-the-chart median values are reported in the boxes at the top of the chart.

The MaxSMT solution concentrates modifications to the region affected by the new request, with a slight adjustment to another previous extra train. Outside the displayed time window, no further changes are visible. The MIP solution, by contrast, redistributes event times across the full timetable horizon, with changes far from the new train both within and beyond the displayed extract. See the Appendix for the same plot over the full horizon in Figure 7. Across all trains present at both the sixth and seventh step, the MIP changes 230 route branching decisions and shifts 67 station arrival times by more than one minute, while MaxSMT changes 114 route branching decisions and shifts only 12 station arrival times by more than one minute: about half the route modifications and a fifth of the timing changes.

The difference does not reflect a sacrifice of optimality. Both solutions are optimal, and the 60s objective granularity leaves many assignments at the same cost. The MaxSMT solver begins each solve from the variable assignments and activity scores of the previous one, so within this set it tends to return the solution nearest the prior schedule; the MIP solver reconstructs its search tree and has no preference among equally good feasible points. Operationally, the smaller revision is the more useful one: a planner reviewing the updated timetable after a MaxSMT solve sees a change small enough to inspect, and trains running periodically and without conflict keep their periodic timings. Nothing in the formulation enforces this, however: it is a tendency arising from state retention rather than a property of the encoding.

6.4 Scalability: Larger Network

To assess whether the conclusions from the Chur–Arosa experiment hold at larger scale, we repeat the experiment on the full RhB network containing 224 context trains (excludes the RE6 Chur–Arosa express service, as this train only runs occasionally in practice). We schedule 30 extra train requests drawn from the five service types in Table 5, with the



■ **Figure 5** Space–time diagrams for the Chur–Arosa section after the seventh extra train addition (coloured lines) overlaid on the sixth-step solution (grey). MaxSMT (left) concentrates revisions to the region affected by the new request and a slight adjustment to one other extra train; the remainder of the timetable is unchanged. MIP (right) redistributes event times across the full planning horizon. The full-horizon overlay is shown in Figure 7 in the Appendix.

assignment to steps again fixed across all 21 runs. Aggregate solve timings for all strategies appear in Table 6. For interpretability, per-step figures and reported timings refer to the run with the median total solve time across the 21 repetitions in this section.

The patterns observed for Chur–Arosa are preserved and amplified, as shown in Figure 6a. Flexible-persistent MaxSMT remains the fastest strategy, completing all 30 additions in a median of 11.7 min, while Flexible-reset takes nearly 30× as long. State retention thus becomes increasingly valuable as the problem grows. Flexible-persistent also outperforms the Batch strategies by a factor of 2×.

The per-step comparison between Flexible-persistent MaxSMT and MIP is shown in Figure 6b and shows how warm-starting behaves at scale. Recall that the MIP is not required to prove optimality of the discretised problem: it terminates as soon as it finds a feasible solution whose continuous objective value does not exceed the optimal discretised MaxSMT objective (Section 5). On this larger and sparser network, most extra train additions interact weakly with the existing schedule and each other, and the warm-started MIP finds feasible solutions near the warm-start incumbent quickly. Nevertheless, the median total time for Flexible-persistent MIP is 117.7 min, dominated by the few steps where the warm-start incumbent is far from optimality and MIP must fall back on branch-and-bound to find a solution of sufficient quality. Step 24, the longest such step, also stands out in the objective: it is the only step at which the Flexible and Commit strategies diverge, with Commit incurring a penalty of 4 min. Here, the new request conflicts with a previously scheduled extra train and requires joint re-optimisation to resolve. This single step is enough for the Batch strategy to outperform Flexible-persistent MIP in solution time overall. In contrast, Flexible-persistent MaxSMT handles the same step in 67.5 s, within its normal range of 0.1 s to 95.4 s per step; the accumulated solver state already encodes the relevant information from earlier solves, for both feasibility and optimality.

Flexible-reset MIP and Batch one-shot MIP could not be completed within the 10h time limit or exceeded available memory, respectively, and are omitted.

7 Conclusion and Outlook

This paper presented an incremental MaxSMT framework for operational timetabling, scheduling a sequence of extra train requests against an existing periodic timetable while minimising delay from the periodic base. The central contribution is an architecture in which the accumulated solver state is retained across successive extra train additions.

The experimental results on the Rhätische Bahn network support three conclusions.

First, state retention reduces solve time for both solvers, with the benefit growing as the sequence of additions lengthens and the schedule becomes denser. For MIP, the advantage is present but erodes at the hardest steps, where the warm-start incumbent diverges from optimality and branch-and-bound is required; at network scale this erosion is severe enough that solving everything in one batch outperforms incrementally solving overall.

Second, the performance gap between MaxSMT and MIP widens as the sequence of additions grows. The asymmetry is in what each solver keeps: MaxSMT retains its lower bound proof and knowledge about infeasibilities across invocations, whereas MIP discards both on each call.

Third, leaving the time windows of previously scheduled extra trains free between solves yields strictly better cumulative objective values than when reducing them: the Commit strategies incur an objective penalty that the Flexible strategies avoid by exploiting the freedom to jointly revise earlier decisions. The MaxSMT solver achieves this without sacrificing its speed advantage, concentrating revisions to the timetable region of each new request and producing interpretable, minimal changes for a planner to review.

Future work. The incremental architecture as presented relies on the monotone-add-only structure of the problem: solver state is retained because the constraint set never shrinks. This assumption is satisfied by extra train requests but excludes settings where earlier decisions must be retracted, for example, if a request is cancelled after admission or if a track possession modifies the feasibility of previously scheduled services. Recent work on incremental MaxSAT solving [20] provides mechanisms for handling such retractions while preserving accumulated state; extending the present architecture to support constraint relaxation would open the framework to a broader class of operational timetabling scenarios.

The tendency of the MaxSMT solver to revise solutions locally arises from retained search heuristics rather than from the objective. Making it explicit would require a secondary criterion penalising deviation from the base timetable. In a core-guided setting this means a second weight level over a soft-clause set. This would give planners direct control over the stability of successive schedule revisions.

■ **Table 5** Service type for each extra train request in the full-network experiment.

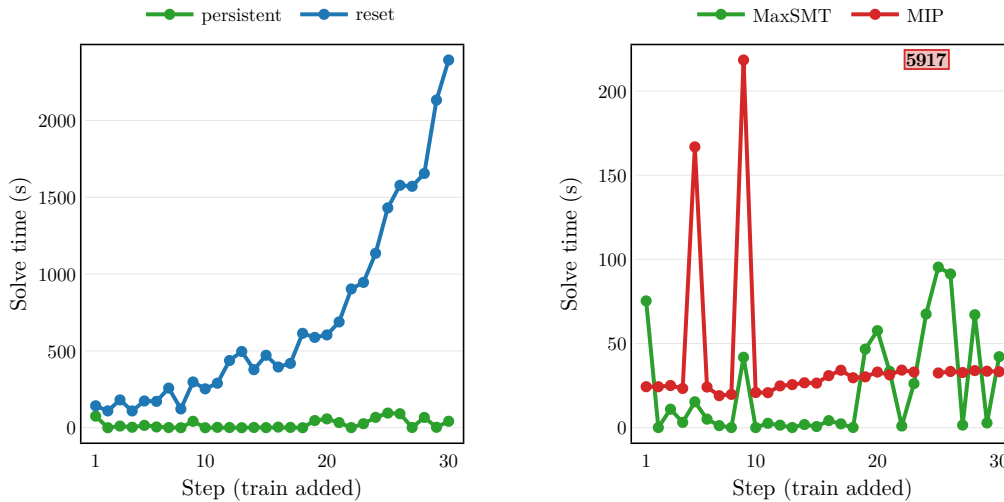
Service type	Steps
Chur – Samedan	3, 4, 7, 15, 16, 18, 19, 20, 26, 28, 30
Samedan – Scuol-Tarasp	2, 5, 6, 9, 10, 12, 14, 21, 22, 25, 27
Landquart – Davos	1, 23, 24
Chur – Illanz	8, 13, 17
Chur – Arosa	11, 29

■ **Table 6** Median total solve time in minutes and factor relative to the baseline (Flexible-persistent, MaxSMT; 11.7 min), across $n = 21$ runs per condition. *Flexible* keeps all admitted extra trains free, while *Commit* makes each newly scheduled extra train a context train before the next solve; *persistent* retains the solver state between steps, *reset* discards it. *Batch: all-extra-trains* warm-starts from the periodic solution and solves all requests at once. *Batch: one-shot* encodes the full periodic timetable together with all requests and solves once.

Strategy	MaxSMT		MIP (Gurobi)	
	Median (min)	Factor	Median (min)	Factor
Flexible-persistent	11.7	1.0×	117.7	10.1×
Flexible-reset	349.1	29.9×	— ^a	—
Commit-persistent	7.2	0.61×	31.6	2.7×
Commit-reset	106.9	9.1×	178.5	15.3×
Batch: all-extra-trains	22.0	1.9×	108.0	9.2×
Batch: one-shot	26.9	2.3×	— ^b	—

^a MIP timed out at 10 h on all runs.

^b MIP exceeded available memory on all runs.



(a) Flexible-persistent versus -reset for MaxSMT. (b) MaxSMT versus MIP for Flexible-persistent.

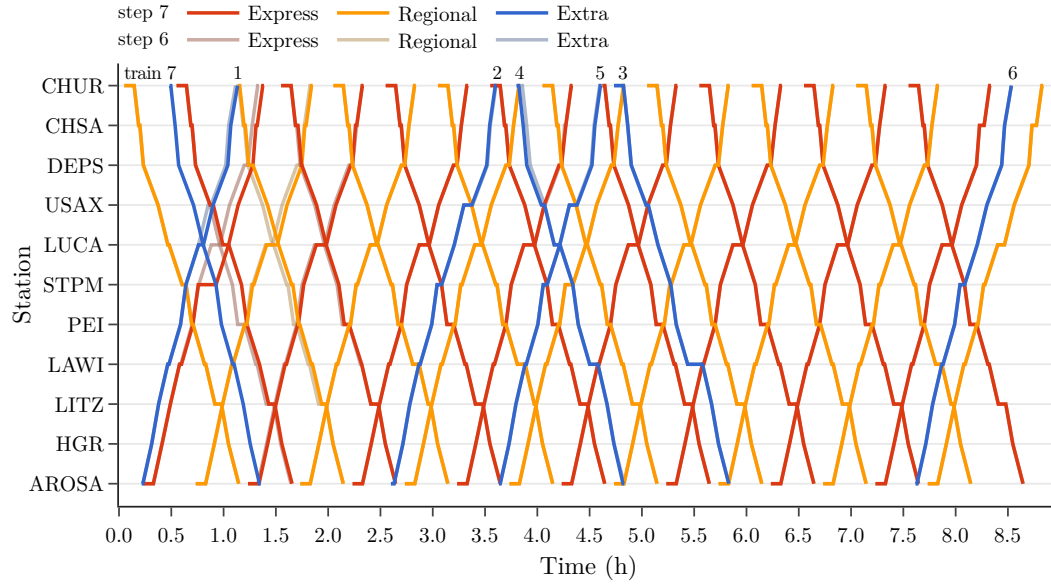
■ **Figure 6** Per-step solve time comparing persistent and reset strategies for MaxSMT (left) and comparing MaxSMT with MIP with the Flexible-persistent strategy (right). Shown are the timings for the run with median total solve time across $n = 21$ runs. Off-the-chart values are reported in the boxes at the top of the chart.

References

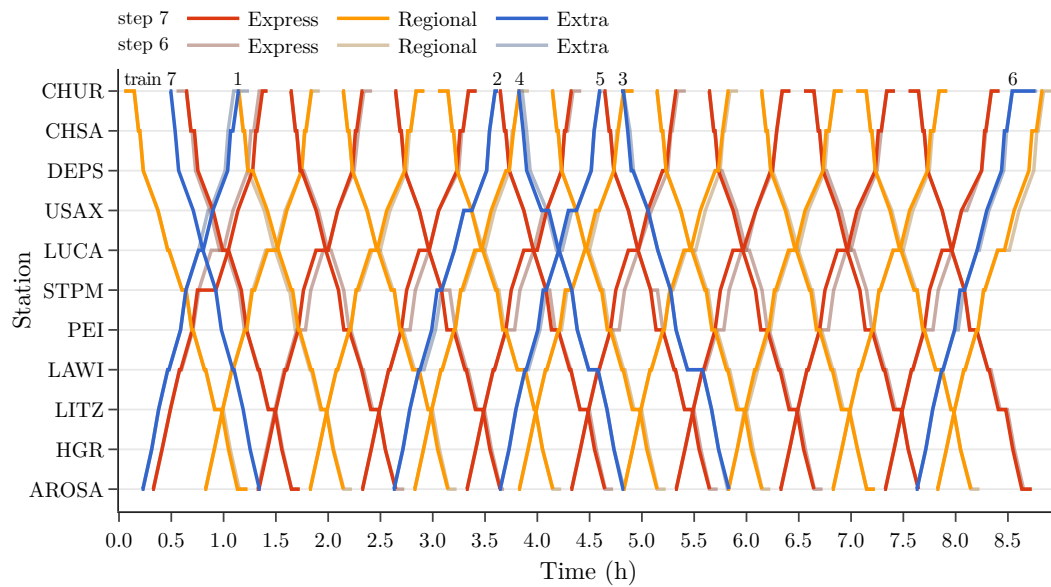
- 1 Gabrio Caimi, Leo Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6(4):285–312, January 2017. doi:10.1016/j.jrtpm.2016.11.002.
- 2 SBB Infrastruktur / SBB Cargo. So entsteht der Fahrplan, April 2014. URL: https://blog.sbbcargo.com/wp-content/uploads/2014/11/A2_Plakat_Fahrplan_Infrastruktur_dt_Web.pdf.
- 3 Pedro José Correia Duarte, Stéphane Dauzère-Pérès, Dennis Huisman, Carlo Mannino, Giorgio Sartor, Norman Weik, and Philipp Widmann. 50 years of OR in railway timetabling and rolling stock planning. *EURO Journal on Transportation and Logistics*, 14:100155, January 2025. doi:10.1016/j.ejtl.2025.100155.
- 4 Scott Cotton and Oded Maler. Fast and Flexible Difference Constraint Propagation for DPLL(T). In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006*, pages 170–183, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11814948_19.
- 5 Anna Livia Croella, Bjørnar Luteberget, Carlo Mannino, and Paolo Ventura. A MaxSAT approach for solving a new Dynamic Discretization Discovery model for train rescheduling problems. *Computers & Operations Research*, 167:106679, July 2024. doi:10.1016/j.cor.2024.106679.
- 6 Niklas Eén and Niklas Sörensson. MiniSat Page, September 2013. URL: <http://minisat.se/>.
- 7 Florian Fuchs and Francesco Corman. An Open Toolbox for Integrated Optimization of Public Transport. In *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–7, June 2019. doi:10.1109/MTITS.2019.8883335.
- 8 Florian Fuchs, Thomas Dubach, Jan Lordieck, Francesco Corman, and Bernardo Martin-Iradi. Handling conflicts in microscopic railway timetabling at scale: A parallelized approach. *Journal of Rail Transport Planning & Management*, 39:100600, September 2026. doi:10.1016/j.jrtpm.2026.100600.
- 9 Florian Fuchs, Bernardo Martin-Iradi, and Francesco Corman. Optimizing periodic stability in railway timetables: A microscopic model for networks with a macroscopic comparison. *Journal of Rail Transport Planning & Management*, 36:100555, December 2025. doi:10.1016/j.jrtpm.2025.100555.
- 10 Filipe Gouveia, Luís Albino, and Ricardo L. Saldanha. A MaxSAT Approach for the Train Timetabling Problem with Route Choice and Other Features. In José Valente de Oliveira, João Leite, João Rodrigues, João Dias, and Pedro Cardoso, editors, *Progress in Artificial Intelligence*, pages 179–191, Cham, 2026. Springer Nature Switzerland. doi:10.1007/978-3-032-05179-0_14.
- 11 Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. RC2: an Efficient MaxSAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):53–64, September 2019. doi:10.3233/SAT190116.
- 12 Oddvar Kloster, Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. An Optimization-Based Decision Support Tool for Incremental Train Timetabling. *Operations Research Forum*, 4(3):65, August 2023. doi:10.1007/s43069-023-00243-2.
- 13 Alexandre Lemos, Filipe Gouveia, Pedro T. Monteiro, and Inês Lynce. Iterative Train Scheduling under Disruption with Maximum Satisfiability. *Journal of Artificial Intelligence Research*, 79:1047–1090, March 2024. doi:10.1613/jair.1.14924.
- 14 Florin Leutwiler and Francesco Corman. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research*, 303(2):525–540, December 2022. doi:10.1016/j.ejor.2022.02.043.
- 15 Christian Liebchen and Rolf H. Möhring. The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables — and Beyond. In Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos D. Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, pages 3–40, Berlin, Heidelberg, 2007. Springer. doi:10.1007/978-3-540-74247-0_1.

- 16 Mark H. Liffiton and Jordyn C. Maglalong. A Cardinality Solver: More Expressive Constraints for Free. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing — SAT 2012*, pages 485–486, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-31612-8_47.
- 17 Ruben Martins, Saurabh Joshi, Vasco Manquinho, and Inês Lynce. Incremental Cardinality Constraints for MaxSAT. In Barry O’Sullivan, editor, *Principles and Practice of Constraint Programming*, pages 531–548, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-10428-7_39.
- 18 Gonalo P. Matos, Lu s M. Albino, Ricardo L. Saldanha, and Ernesto M. Morgado. Solving periodic timetabling problems with SAT and machine learning. *Public Transport*, 13(3):625–648, October 2021. doi:10.1007/s12469-020-00244-y.
- 19 Antonio Morgado, Carmine Dodaro, and Joao Marques-Silva. Core-Guided MaxSAT with Soft Cardinality Constraints. In Barry O’Sullivan, editor, *Principles and Practice of Constraint Programming*, pages 564–573, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-10428-7_41.
- 20 Andreas Niskanen, Jeremias Berg, and Matti J rvisalo. Incremental Maximum Satisfiability. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum f r Informatik. doi:10.4230/LIPIcs.SAT.2022.14.
- 21 P. Serafini and W. Ukovich. A Periodic Job Shop Model. *IFAC Proceedings Volumes*, 22(14):89–94, September 1989. doi:10.1016/S1474-6670(17)54331-1.
- 22 Xujie Si, Xin Zhang, Vasco Manquinho, Mikol   Janota, Alexey Ignatiev, and Mayur Naik. On Incremental Core-Guided MaxSAT Solving. In Michel Rueher, editor, *Principles and Practice of Constraint Programming*, volume 9892, pages 473–482. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-319-44953-1_30.
- 23 Sander Van Aken, Nikola Be sinovi , and Rob M. P. Goverde. Designing alternative railway timetables under infrastructure maintenance possessions. *Transportation Research Part B: Methodological*, 98:224–238, April 2017. doi:10.1016/j.trb.2016.12.019.

A Full Space-Time diagram



(a) MaxSMT.



(b) MIP.

Figure 7 Space-time diagrams for Chur–Arosa case study after the seventh extra train addition (coloured lines) overlaid on the sixth-step solution (grey). MaxSMT (top) revises only the region directly affected by the new request and an adjustment to one other extra train; the remainder of the timetable is unchanged. MIP (bottom) redistributes event times across the full planning horizon.

Dispatching-Area Design

Tomas Lidén 

Department of Science and Technology (ITN), Linköping University, Sweden

Christiane Schmidt 

Department of Science and Technology (ITN), Linköping University, Sweden

Rabii Zahir 

Department of Science and Technology (ITN), Linköping University, Sweden

Abstract

Increasing traffic density and system interdependencies in railway networks can lead to imbalanced workload distribution among train dispatchers. Traditional dispatching-area design, often based on historical and geographical considerations, may no longer reflect current infrastructure complexity and coordination requirements, potentially affecting punctuality and operational robustness.

In this paper, we study the problem of partitioning a set of train stations into a predefined number of dispatching areas, formulated as a multi-objective mixed-integer programming problem. The objective is to minimize a weighted sum of inter-area interaction complexity and intra-area operational complexity. Inter-area complexity captures coordination needs through the number of trains crossing area boundaries, while intra-area complexity is modeled using station-related parameters, including track layout, traffic density, starting and ending services, and traffic heterogeneity (mix of passenger and freight traffic).

We apply the suggested approach to instances from the Malmö dispatching center in the southern part of the Swedish railway network, illustrating how an optimization-grounded sectorization framework can support structured and balanced redesign of dispatching areas. We evaluate the model's robustness by through sensitivity analyses on objective weights, minimum area size, and the number of dispatching areas. These analyses provide insights into how different planning priorities influence the resulting sectorization.

2012 ACM Subject Classification Applied computing → Transportation

Keywords and phrases Railway sectorization, Area design, Train dispatching

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.5

Funding This work is part of the Trafikverket-funded project Dispatching Areas: Combinations and Design (DACoD) (TRV 2023/91772).

Acknowledgements Special thanks to David Dekker for his support and guidance in working with the data used in this paper, which was greatly appreciated.

1 Introduction

A railway network is divided into smaller regions corresponding to different parts of a country, each supervised by a dispatching center. In turn, each region is further subdivided into dispatching areas managed by individual dispatchers to ensure safe and efficient traffic flows. An area may include one or several stations, such as a passenger station, a freight terminal, or simply a section of track equipped with switches. Historically, an area corresponds to a major station together with small adjacent ones. However, an increase in traffic demand, timetable density, and system interdependency may significantly impact the workload level and its distribution over time and across the network. The consequence is that some areas may experience a high traffic density with many conflicting train paths, while others are quite simple to manage. Such imbalanced area design has a direct impact on workload distribution between train dispatchers, potentially affecting punctuality and operational robustness.



© Tomas Lidén, Christiane Schmidt, and Rabii Zahir;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 5; pp. 5:1–5:17



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To address this issue, infrastructure managers aim to redesign operational areas taking into account station complexity, traffic flows between stations, and spatial coherence. The objective is to distribute operational complexity more evenly while maintaining spatial continuity. A balanced area design supports decision making, improves disturbance handling, and contributes to more efficient utilization of human resources.

Partitioning a network into sectors, often called sectorization, can be viewed as a graph partitioning problem. In this paper, we suggest a framework for partitioning a railway network, given as a set of stations, into a predefined number of dispatching areas (sectors). We apply the suggested model on instances from the Malmö dispatching center, which is responsible for the southern part of the Swedish railway network. Today, the considered network is divided into about 15 areas. The present structure has evolved gradually over several decades and originates from an early organization based primarily on historically important main stations. Our goal is to improve the area design based on the actual traffic demand and infrastructure complexity. The contributions of this work are to propose a relatively simple optimization model that supports design as well as redesign of dispatching areas.

Related Work. Sectorization (also called distriction or territory design) is the process of partitioning a larger area into smaller units according to defined goals while satisfying given constraints [13]. Such problems arise in many practical contexts, including electoral districting [3], road-traffic control [6], maintenance-service areas [11], school [7] and healthcare zoning [2], police-coverage planning [4], and air-traffic-control operations [19]. The goal of sectorization is to create manageable working sectors, in terms of workload, while certain constraints are fulfilled [13]. Typical examples of these constraints include equilibrium (balanced area, demand, workload, etc.), compactness (dense sector shape), contiguity (sector convexity), capacity (upper bound), boundary (certain basic units must belong to different sectors) and neighborhood (certain basic units must belong to the same sector). In air-traffic control, a field with many similarities to train dispatching, researchers showed that the workload of a traffic controller depends not only on the number of aircraft in a sector, but rather on their dynamic characteristics, their routes, and their interaction with each other [5]. Sridhar et al. [14] defined airspace complexity through (i) structural features – fixed geometry like airways and intersections –, and (ii) flow complexity – dynamic aspects such as aircraft numbers and interactions. Gianazza [8] suggested a data-driven model for air-traffic-control sectorization based on workload. The explanatory variables in their model included the sector size, number of aircraft within a sector and their average absolute vertical speed, incoming flow, and the number of speed vector intersections (with an angle greater than 20 degrees). Yousefi and Donohue [17] proposed an integer-programming method to improve static airspace sectorization by balancing controller workload. They divided the airspace into hexagonal cells, estimated each cell’s workload through simulation, and then applied an integer-programming-based clustering algorithm to group the cells into sectors with similar overall workload levels. Mitchell et al. [12] examined dynamic airspace design using computational geometry to create sectors that evenly distribute controller workload over time, rather than depending on static boundaries. Xue [16] proposed a two-step approach to sectorize airspace. The first step, consists in generating initial sectors based on Voronoi diagrams, and then a genetic algorithm is used to fine-tune the sectors’ boundaries while optimizing the performance (low coordination between sectors) and balancing workload. Granberg et al. [9] presented a mixed-integer-program (MIP) framework for airspace sectorization in terminal maneuvering areas, incorporating airspace complexity and geometric constraints to support effective conflict management.

Sectorization of train-dispatching areas has been addressed by several researchers. Zhao et al. [18] formulated the train dispatcher desk districting problem as a MIP to create contiguous, feasible, and workload-balanced districts while minimizing inter-desk coordination. In their approach, they balance workload by minimizing the deviation between districts' workload and an average workload value. The workload is estimated using proxies based on section length and train movements, including the number of trains traveling through sections and stopping at stations. In contrast, in the present study we focus on districting stations while omitting the explicit assignment of track sections, which can subsequently be handled through operational rules of thumb, such as dividing sections at their midpoint. Furthermore, workload estimation in our study is based on a weighted combination of five operational characteristics: track-layout complexity, traffic density, starting and ending services, traffic heterogeneity in terms of passenger and freight traffic mix, and inter-areas flow. Instead of minimizing workload deviations from a single average value – not well suited to either our approach or our data – we propose different workload balancing strategies depending on the specific type of workload being considered. To solve instances for a real high-speed railway network, Zhao et al. [18] needed an iterative-search-based heuristic, which resulted in a total computation time below 33 minutes. Wang et al. [15] addressed the same problem as Zhao et al. [18] and generalized it from a railway-dispatcher-specific formulation to a general simultaneous node-and-link districting problem, using column generation and a hybrid exact–heuristic framework to yield near-optimal or optimal solutions for most instances.

In general, sectorization/districting problems are NP-hard [1, 10]. In our approach, we propose an exact method based on a MIP formulation. We formulate the dispatching-area design problem as a multi-objective optimization problem due to its conflicting criteria and use scalarization methods to explore the Pareto-optimal solution space. Since the practical requirements neither impose strict area connectivity (traffic is allowed to leave and re-enter an area) nor the compactness of the areas (given the eel-like shape of the network), we can use a simplified model structure. As a result, we solve real-world instances to optimum or near optimum within acceptable computational times.

The remainder of this paper is organized as follows. In Section 2, we formulate the problem statement. In Section 3, we present the MIP formulation. In Section 4, we describe the experimental setup. In Section 5, we present and analyze the results. Finally, in Section 6, we conclude with future work and recommendations.

2 Problem Statement

Our sectorization problem consists of determining an optimal partition of a set of train stations S into a set of areas A with a predefined cardinality. The problem statement of our dispatching-area-design is as follows:

Given: A set of train stations; a set of adjacent station-pairs; the number of train tracks, movements, and starting and ending train services in each station, as well as the corresponding maximum values among all stations; the number of crossing train movements between each pair of stations; a set of trains operating in the network; a list of trains visiting each station and the number of yearly movements for each train; the number of passenger and freight trains in each station per year; a list of factor weights representing the priorities between the objectives; and a predefined number of areas representing dispatching sectors.

Find: an area partition of the stations into a predefined number of areas such that each station is assigned to exactly one area, inter-area interaction, measured by area-crossing train flows, is minimized, and the resulting intra-area workloads are as balanced as possible.

Workload balance is achieved by minimizing a weighted sum of deviations from predefined reference values, derived from the largest stations, for the total number of tracks in each area, the total number of starting and ending train services in each area, and the total number of train movements in each area. In addition, we promote traffic homogeneity within each area by encouraging each area to be dominated either by passenger trains or by freight trains.

3 Mathematical Formulation

We view the dispatching-area design problem as involving multiple, often conflicting, criteria. Improving one aspect of the design may negatively affect another – for example, reducing inter-area complexity may increase intra-area workload. We therefore formulate the problem as a multi-objective optimization problem, aiming to characterize the trade-offs between the two competing criteria: the complexity within areas and the complexity of inter-area interactions.

The complexity within areas is determined by the complexities of the stations assigned to each area. Each station i is associated with parameters representing its complexities, captured by (i) the number of tracks in the station, (ii) the number of starting and ending train lines, (iii) the number of unique trains visiting the station, and (iv) the heterogeneity of the traffic (passenger versus freight trains). The inter-area interaction complexity is determined by the number of trains flowing from one area j to another. The objective is to find a partition of with the lowest total cost, defined as a convex combination of the complexity factors within each area and the complexity of inter-area interactions.

We model the problem as a MIP:

$$\text{minimize} \quad \sum_{k=1}^5 \alpha_k \cdot \frac{f_k - f_k^{min}}{\bar{f}_k - f_k^{min}} \quad (1)$$

subject to:

$$\sum_j y_{i,j} = 1 \quad \forall i \in S \quad (2)$$

$$f_5 = \sum_{\{i,i'\} \in E} p_{i,i'} \cdot m_{i,i'} \quad (3)$$

$$y_{i,j} - y_{i',j} \leq p_{i,i'} \quad \forall j \in A, \forall \{i,i'\} \in E \quad (4)$$

$$y_{i',j} - y_{i,j} \leq p_{i,i'} \quad \forall j \in A, \forall \{i,i'\} \in E \quad (5)$$

$$\sum_i \ell_{i,k} \cdot y_{i,j} - \ell_k^{max} \leq f_k \quad \forall j \in A, \forall k \in \{1, 2\} \quad (6)$$

$$\ell_k^{max} - \sum_i \ell_{i,k} \cdot y_{i,j} \leq f_k \quad \forall j \in A, \forall k \in \{1, 2\} \quad (7)$$

$$t_{j,n}^{ar} \geq y_{i,j} \cdot t_{i,n}^{st} \quad \forall j \in A, \forall n \in T \quad (8)$$

$$t_{j,n}^{ar} \leq \sum_i y_{i,j} \cdot t_{i,n}^{st} \quad \forall j \in A, \forall n \in T \quad (9)$$

$$t_{max} - \sum_n t_{j,n}^{ar} \cdot t_n \leq f_3 \quad \forall j \in A \quad (10)$$

$$\sum_i y_{i,j} \cdot m_i^p \leq f_4 + M \cdot z_j \quad \forall j \in A \quad (11)$$

$$\sum_i y_{i,j} \cdot m_i^f \leq f_4 + M \cdot (1 - z_j) \quad \forall j \in A \quad (12)$$

$$y_{i,j} \in \{0, 1\} \quad \forall i \in S, j \in A \quad (13)$$

$$p_{i,i'} \in \{0, 1\} \quad \forall \{i, i'\} \in E \quad (14)$$

$$t_{j,n}^{ar} \in \{0, 1\} \quad \forall j \in A, n \in T \quad (15)$$

$$z_j \in \{0, 1\} \quad \forall j \in A \quad (16)$$

$$f_k \in \mathbb{R}^+ \quad \forall k \in \{1, 2, 3, 4, 5\} \quad (17)$$

The main decision variable is the binary $y_{i,j}$, which assigns a station i to area j whenever $y_{i,j} = 1$. The variable f_k (with $k \in \{1, 2, 3, 4, 5\}$) is used to determine the workload balance for each workload factor. We present more details on the variables and parameters used in the model in Table 1.

The first constraint (2) assures that each station is assigned to exactly one area. Constraint (3) defines the fifth component of the objective value, f_5 , which represents the total number of train movements between areas. Constraints (4) and (5) activate the variable $p_{i,i'}$ whenever stations i and i' are assigned to different areas, which would result in a penalization, in the objective function, by a weight factor $m_{i,i'}$ corresponding to the train movements between the adjacent station pair $\{i, i'\}$. Here, $\{i, i'\}$ denotes an element of the set of edges E in the adjacency graph. This component of the objective function is represented by f_5 . Constraints (6) and (7) define the variables f_1 and f_2 as upper bounds on the deviation of the area workloads from their respective maxima, where ℓ_1^{max} and ℓ_2^{max} denote the maximum number of tracks and the maximum number of starting and ending train services in a station, respectively. These parameters, ℓ_1^{max} and ℓ_2^{max} , are derived from observations at large stations, such as Malmö Central. Minimizing deviations from these workload values promotes a balanced allocation of workload across areas. Since high workload levels are typically associated with large stations assigned to specific areas, balancing the workload may require aggregating smaller stations into other areas to reduce workload disparities.

Balancing the number of train movements between areas is achieved by minimizing the deviation between the number of trains (from the set of trains T indexed by n) visiting each area and the maximum number of trains, t_{max} , observed at a station, typically a large one. This deviation's upper bound is determined by the variable f_3 (see Constraint (10)). We use Constraints (8) and (9) to determine the binary variable $t_{j,n}^{ar}$ for whether a train n visited an area j , based on whether the train n visited a station i assigned to the area j .

Given that we aim for traffic that is as homogeneous as possible – that is, areas dominated either by passenger trains or by freight trains – we introduce the integer variable f_4 , representing an upper bound on the minimum number of passenger (m_j^p) or freight (m_j^f) trains across all areas (see Constraints (11) and (12)). By minimizing f_4 , the model seeks to reduce the smaller of the two traffic types within each area. Consequently, stations with relatively low numbers of either passenger or freight trains are encouraged to be grouped into the same area, leading to areas that are more clearly dominated by a single traffic type and therefore more homogeneous. Furthermore, we define the binary variable z_j , which takes the value 1 if area j contains more passenger trains than freight trains, and 0 otherwise.

The objective function is defined as a convex combination of the weighted complexity measures, where the weights are denoted by α_k ($k \in [1, 2, 3, 4, 5]$). Since the complexity measures differ substantially in scale and units, each objective term is normalized by its attainable range. Specifically, the k -th objective is normalized as: $\frac{f_k - f_k^{min}}{f_k - f_k^{min}}$, where f_k^{min} denotes the minimum attainable value of the corresponding k -th complexity measure, and $\bar{f}_k$ denotes the maximum obtained value when each of the other objectives is individually minimized; a detailed definition of these values is provided in Subsection 5.1. By this normalization we ensure that all objective components contribute on a comparable scale, and that the total objective value is in the range $[0, 1]$.

■ **Table 1** Description of the model's variable and parameters.

Notation	Type	Description
$y_{i,j}$	var (binary)	=1 if station i is assigned to area j
$p_{i,i'}$	var (binary)	=1 if adjacent stations i and i' are assigned to different areas
f_1	var	max deviation of the number of tracks from the max
f_2	var	max deviation of the number of starting/ending services from the max
f_3	var	max deviation of the number of train movements from the max
f_4	var	max-min of either passenger or freight trains across all areas
f_5	var	total number of crossing train movements between areas
$t_{j,n}^{ar}$	var(binary)	=1 if train n visits area j
z_j	var (binary)	=1 if area j contains more passenger than freight trains
S	set	set of train stations (indexed by i)
A	set	set of areas (indexed by j)
T	set	set of trains (indexed by n)
$m_{i,i'}$	param	number of trains movements between adjacent stations i and i'
$\ell_{1,i}$	param	total number of tracks in station i
$\ell_{2,i}$	param	total number of starting/ending services in station i
ℓ_1^{max}	param	maximum number of tracks in a large station
ℓ_2^{max}	param	maximum number of starting/ending services in a large station
t_{max}	param	maximum number of train movements in a large station
t_n	param	number of movements for train n
$t_{i,n}^{st}$	param (binary)	=1 if train n visits station i
m_i^p	param	number of movements of passenger trains visiting station i
m_i^f	param	number of movements of freight trains visiting station i
α_1	param	cost associated with the number of tracks in areas
α_2	param	cost associated with the number of starting/ending trains in areas
α_3	param	cost associated with the number of train movements in areas
α_4	param	cost associated with the traffic homogeneity in areas
α_5	param	cost associated with inter-area-crossing train movements
M	param	a big number; given by $\max(\sum_i m_i^f; \sum_i m_i^p)$

4 Experiment Setup

We consider the railway network controlled by the Malmö dispatching center, which covers the southern region of Sweden. The network comprises 178 operation points (*driftplatser* in Swedish), including switches, stations, and shunting areas. For simplicity, we refer to all operation points as stations throughout this paper.

The available data include route information and the annual train timetable for 2026, specifying train IDs, train types (passenger or freight), arrival and departure times at each station, and service frequencies. In addition, data on the number of tracks at each station were provided. Based on these data, we derived the parameters $m_{i,i'}$, $\ell_{1,i}$, $\ell_{2,i}$, t_n , $t_{i,n}^{st}$, m_i^p , and m_i^f . For the workload data representing a large station, the maximum numbers of train movements, tracks, and starting/ending train services were set to $t_{max} = 299020$, $\ell_1^{max} = 27$, and $\ell_2^{max} = 170024$, respectively. Both t_{max} and ℓ_2^{max} were obtained at Malmö Central Station (Mc), whereas ℓ_1^{max} occurred at Malmö Godsbangård (Mgb).

Before conducting our experiments, and to reduce the domain size and break symmetries, we introduce several additional constraints to the MIP formulation. The first group of constraints restricts the assignment of stations to areas based on their geographical location.

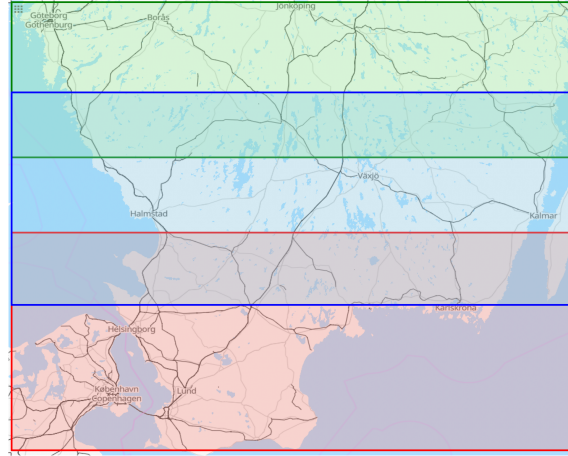
We begin by partitioning the network into three regions: south, middle, and north. Stations in the southern region are prohibited from being assigned to areas 7–14 and may therefore only be assigned to areas in the interval $[0,6]$. Similarly, stations in the middle and northern regions may only be assigned to areas in the intervals $[6,9]$ and $[9,14]$, respectively. Note that we allow some overlap between the admissible area assignments of adjacent regions to provide additional flexibility. We denote the sets of prohibited area assignments by $\overline{A}_s$, $\overline{A}_m$, $\overline{A}_n$ (the complements of the allowed area assignments) for the south, middle, and north, respectively.

Based on their coordinates, we partition the stations into three subsets, S_s , S_m , and S_n , which denote stations in the south, middle and north, respectively. In Figure 1, we illustrate the geographical boundaries of the regions corresponding to S_s , S_m , and S_n . The first group of additional constraints are:

$$\sum_{i \in S_s} y_{i,j} = 0 \quad \forall j \in \overline{A}_s \quad (18)$$

$$\sum_{i \in S_m} y_{i,j} = 0 \quad \forall j \in \overline{A}_m \quad (19)$$

$$\sum_{i \in S_n} y_{i,j} = 0 \quad \forall j \in \overline{A}_n \quad (20)$$



■ **Figure 1** Partition of the network into the three regions, south (red), middle (blue), and north (green).

To further reduce symmetry within each geographical region, we fix the assignments of a small number of major stations to specific areas within their respective admissible area subsets. These fixed assignments serve as anchor points and eliminate equivalent solutions that differ only by permutations of area labels within a region.

Finally, we introduce a third constraints group to prohibit assignments of station pairs to the same area if these stations are located too far from each other. The assumed reference distance is 160 km, which refers to the distance between Kalmar and Rimforsa. This assumption is based on a simple calculation: the total area under study covers approximately $60\,000 \text{ km}^2$, so dividing by the number of areas ($n = 15$) gives an average area of $60\,000/15 = 4\,000 \text{ km}^2$. A circle enclosing this area would have a diameter of $d = 2\sqrt{4000/\pi} \approx 71 \text{ km}$. Since a circle minimizes the maximum internal distance for a given area, this value is a lower bound: any less regular area shape will have a larger diameter. Our threshold of 160 km is more than double this reference value, leaving sufficient margin for irregular area shapes,

or larger areas, while still excluding implausible groupings of distant stations. We generate the binary parameter $d_{i,i'}$ that is equal to 1 if the distance between the stations i and i' is greater than the threshold distance. The new additional constraints are:

$$d_{i,i'} \cdot (y_{i,j} + y_{i',j}) \leq 1 \quad \forall j \in A, \forall i \in S, \forall i' \in S \quad (21)$$

We believe that our assumptions, within the added constraints, are reasonable and not overly restrictive. Large stations are unlikely to be assigned to the same area except in very poor solutions, as such assignments would result in poor objective values with respect to both inter- and intra-area measures. In contrast, we acknowledge that prohibiting the assignment of certain geographically distant station pairs to the same area could affect the intra-area workload. However, such assignments would correspond to operationally undesirable solutions and are therefore unlikely to be of practical interest in the dispatching context.

To model and solve our MIP, we employ the Python programming language together with the Gurobi solver (version 12.0.3). The stopping criteria are either a runtime limit of 24 hours or achieving an optimality gap of 1%. The computational experiments are carried out on a high-performance Sigma server (2019), equipped with Intel Xeon Gold 6130 nodes featuring 32 CPU cores and 384 GiB of memory, provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS).

5 Computational Results

In this section, we present the results of our experiments. We begin with the normalization step, as our multiobjective model involves objectives with different units and scales. We then examine the impact of varying different MIP parameters in the remaining experiments.

5.1 Normalization of the Objective Value

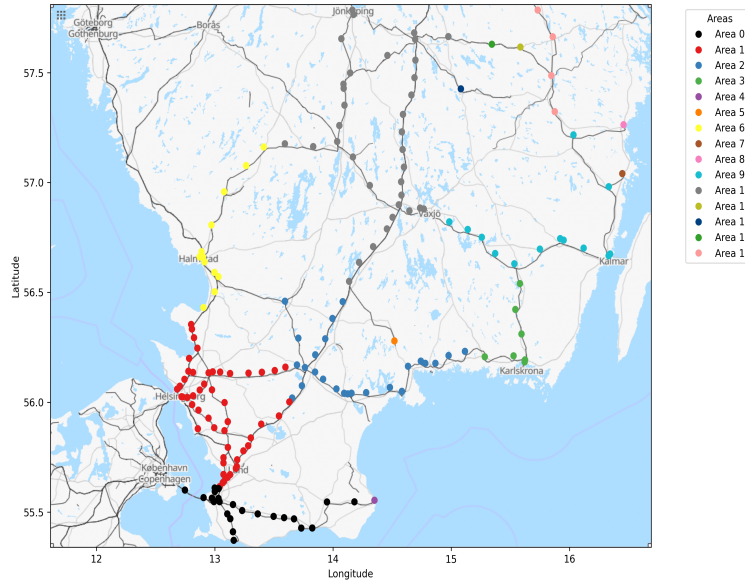
Before the experiments, we normalize the objective function by determining the minimum and maximum attainable values, f_k^{min} and $\bar{f}_k$. We obtain f_k^{min} by solving the MIP for the k -th objective alone, setting its weight to 1 and all others to 0. The resulting objective values and other complexity measures are reported in Table 2. Each instance is denoted by $I_{\alpha_1;\alpha_2;\alpha_3;\alpha_4;\alpha_5}$, where α_k is the weight of the k -th objective term. We define $\bar{f}_k$ as the highest value of the k -th objective across all instances. The last column reports the number of disconnected and large areas. An area is disconnected if the subgraph induced by its stations is disconnected. Although such areas can be operationally feasible – e.g., when a large station belongs to one area and stations on several outgoing lines to another – widely separated connected components are undesirable. We therefore track disconnected areas throughout the experiments.

An area is considered large if it contains at least five stations. Although small areas are acceptable, they are generally less desirable. These characteristics are not explicitly controlled but emerge from the optimization. Thus, reducing disconnected and small areas, particularly those interrupted by multiple other areas, would improve solution quality.

The resulting runtime for all instances was below 23 seconds. The minimum number of tracks is $f_1^{min} = 27$, obtained for the area containing only the Mgb station, a marshaling yard near Malmö. When targeting inter-area flow, all other objectives reach their $\bar{f}_k$ values, showing a clear trade-off between inter- and intra-area complexities. This is further evident from the inter-area-flow values, which are equal to $\bar{f}_5$ for all instances except the last, where this objective is explicitly targeted. Moreover, minimizing inter-area flow results in fully connected areas, as reducing area-border-crossing flows lowers the penalty cost. However, this

■ **Table 2** Normalization results, including runtime (in seconds), objective values f_1 (tracks), f_2 (start-/end train services), f_3 (train movements), f_4 (freight/passenger train movements), f_5 (total inter-area flows), and the number of disconnected and large areas (D;L). $f_k^{\min}$ and $\bar{f}_k$ are in bold.

instance	r (s)	obj	f_1 (track)	f_2 (st/end)	f_3 (mov)	f_4 (m^f/m^p)	f_5 (inter-flow)	#D;L areas
$I_{1;0;0;0;0}$	22	27	27	170024	286612	1088157	7040596	12;12
$I_{0;1;0;0;0}$	15	162647	80	162647	257476	1631859	7040596	14;12
$I_{0;0;1;0;0}$	23	233629	76	169099	233629	1446174	7040596	14;13
$I_{0;0;0;1;0}$	20	109310	38	166579	287412	109310	7040596	14;13
$I_{0;0;0;0;1}$	12	479297	153	170024	298608	3521680	479297	0;8
		$f_k^{\min}$	27	162647	233629	109310	479297	
		$\bar{f}_k$	153	170024	298608	3521680	7040596	



■ **Figure 2** Solution for Instance $I_{0;0;0;0;1}$. The colored dots denote station-area assignments.

connectivity comes at the cost of extreme imbalances in the intra-area measures. Figure 2 illustrates the partition obtained for Instance $I_{0;0;0;0;1}$, highlighting the imbalance in area sizes, with only eight areas being large. This occurs because the solver tends to merge as many neighboring stations as possible into large areas, leaving a small number of stations in smaller areas that only satisfy the partition's cardinality constraint.

5.2 Equal Weights for all Objectives

Now that the objectives are normalized, we proceed with several experiment sets. We first assign equal weights to all objectives. The remaining sets investigate the impact of changing the objectives' weights (Subsections 5.3 and 5.4), restricting area size (Subsection 5.6), and reducing the number of areas (Subsection 5.7).

In the first set, all objectives are assigned equal weights ($\alpha_k = \frac{1}{5}$, $\forall k$). We report the results in the first row in Table 3. For each obtained objective component f_k , we report a percentage value that measures its relative position within the range defined by $f_k^{\min}$ and $\bar{f}_k$.

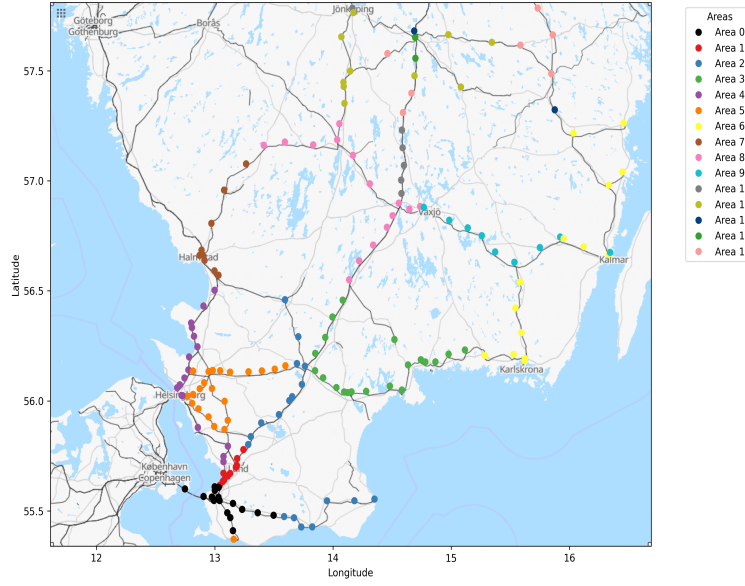
■ **Table 3** Results for decreasing and increasing some targeted weights.

instance	r (m)	obj 10^{-3}	f_1 (track)	f_2 (st/end)	f_3 (mov)	f_4 (m^f/m^p)	f_5 (inter-flow)	#D;L areas
$I_{\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}}$	137	34	33 5%	162647 0%	233640 0%	169841 2%	1173462 11%	10;13
$I_{\frac{4}{400}, \frac{99}{400}, \frac{99}{400}, \frac{99}{400}, \frac{99}{400}}$	50	27	53 21%	162647 0%	233640 0%	163051 2%	1035703 8%	8;13
$I_{\frac{99}{400}, \frac{4}{400}, \frac{99}{400}, \frac{99}{400}, \frac{99}{400}}$	120	42	33 5%	162859 3%	233640 0%	169841 2%	1161709 10%	9;13
$I_{\frac{99}{400}, \frac{99}{400}, \frac{4}{400}, \frac{99}{400}, \frac{99}{400}}$	296	32	27 0%	162647 0%	266063 50%	163880 2%	1097347 9%	6;13
$I_{\frac{99}{400}, \frac{99}{400}, \frac{99}{400}, \frac{4}{400}, \frac{99}{400}}$	7	43	35 6%	162647 0%	233640 0%	300646 6%	1129301 10%	9;13
$I_{\frac{99}{400}, \frac{99}{400}, \frac{99}{400}, \frac{99}{400}, \frac{4}{400}}$	11	14	33 5%	162647 0%	233629 0%	118258 0%	1741982 19%	14;13
$I_{\frac{156}{400}, \frac{61}{400}, \frac{61}{400}, \frac{61}{400}, \frac{61}{400}}$	83	38	33 5%	162647 0%	233640 0%	169841 2%	1182046 11%	9;13
$I_{\frac{61}{400}, \frac{156}{400}, \frac{61}{400}, \frac{61}{400}, \frac{61}{400}}$	65	26	33 5%	162647 0%	233640 0%	169841 2%	1173462 11%	9;13
$I_{\frac{61}{400}, \frac{61}{400}, \frac{156}{400}, \frac{61}{400}, \frac{61}{400}}$	53	26	33 5%	162647 0%	233640 0%	161285 2%	1201512 11%	9;13
$I_{\frac{61}{400}, \frac{61}{400}, \frac{61}{400}, \frac{156}{400}, \frac{61}{400}}$	116	29	33 5%	162647 0%	233640 0%	133680 1%	1294482 12%	10;13
$I_{\frac{61}{400}, \frac{61}{400}, \frac{61}{400}, \frac{61}{400}, \frac{156}{400}}$	91	49	35 6%	162647 0%	233640 0%	235693 4%	1047763 9%	8;13

Comparing each resulting objective component with its corresponding f_k^{min} for $k \in [1, \dots, 5]$, we observe percentage increases of 5%, 0%, 0%, 2%, and 11%, respectively. The increase in inter-area flow is relatively larger than that in intra-area imbalance, which is practically negligible – particularly for f_2 and f_3 , which retain their optimal values. This suggests that the first four components drive the solution consistently toward improved intra-area balance. Furthermore, the resulting partition contains 10 disconnected areas, which remains relatively high given that only the inter-area flow component (f_5) promotes connectivity. Figure 3 illustrates the resulting partition, highlighting both the variation in the number of stations per area and the appearance of disconnected components.

5.3 Decreasing One of the Weights

In this set, we vary the objective weights by reducing one to a low value (0.01) while keeping the others equal. To preserve the convexity of the weighted sum, the remaining weights are increased by $\frac{\delta}{4}$, where δ is the deviation from the initial value $\frac{1}{5}$. For example, reducing the first weight gives $[\frac{1}{5} - \delta; \frac{1}{5} + \frac{\delta}{4}; \frac{1}{5} + \frac{\delta}{4}; \frac{1}{5} + \frac{\delta}{4}; \frac{1}{5} + \frac{\delta}{4}]$. Thus, the weights always sum to 1. To save space in the result tables, we denote instance weights as fractions rather than decimals, using a common denominator whenever possible to facilitate comparison. The results are reported in Rows 2–6 of Table 3.



■ **Figure 3** Solution using equal weights of $\frac{1}{5}$ for all objective components.

Comparing each objective component across instances with its corresponding f_k^{min} , we observe that all components increase when their associated weight α_k is the smallest. However, some components – namely f_1 , f_4 and f_5 – exhibit greater variability across instances, whereas f_2 , f_3 retain their f_k^{min} values in most cases, except when their corresponding weights are the smallest. Moreover, f_1 , f_3 and f_5 experience the largest deviations (+21%, +50%, and +19%, respectively) from their respective f_k^{min} values when their associated weight α_k is the smallest, while f_2 and f_4 remain comparatively close to their minimum values. These observations help identify the objective components that are most sensitive to reductions in their assigned weights and therefore the most relevant for the upcoming experiments.

In the last column, the number of disconnected areas is relatively low for all instances except the last one, $I_{\frac{99}{400}; \frac{99}{400}; \frac{99}{400}; \frac{99}{400}; \frac{4}{400}}$, which has 14 disconnected areas. This suggests that assigning a low weight to f_5 leads to more disconnected areas. Comparing the objective component f_5 for the equal-weight instance $(I_{\frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{1}{5}})$ with the other instances, we observe that f_5 improves whenever an intra-area weight is decreased, while it increases only when α_5 is decreased. This confirms the trade-off between inter- and intra-area workloads.

5.4 Increasing One of the Weights

We adopt the same methodology as in Subsection 5.3, but increase one weight at a time instead of decreasing it, while keeping the same deviation value, $\delta = 0.19$. Emphasizing the first objective yields the weight vector $[0.2 + \delta; 0.2 - \frac{\delta}{4}; 0.2 - \frac{\delta}{4}; 0.2 - \frac{\delta}{4}; 0.2 - \frac{\delta}{4}] = [0.39; 0.1525; 0.1525; 0.1525; 0.1525]$, expressed in fractional form as $[\frac{156}{400}; \frac{61}{400}; \frac{61}{400}; \frac{61}{400}; \frac{61}{400}]$. We report the results in the last five rows of Table 3.

Using instances $(I_{\frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{1}{5}; \frac{1}{5}})$ as a reference, almost all intra-area objective components remain stable. The exceptions are f_1 and f_4 in the last instance, $(I_{\frac{61}{400}; \frac{61}{400}; \frac{61}{400}; \frac{61}{400}; \frac{156}{400}})$, showing a relatively larger increase. In the same instance, f_5 improves as its weight α_5 increases, accompanied by a reduction in the number of disconnected areas, confirming the effect of assigning a higher weight to inter-area flow on maintaining area connectivity. However, the number of large areas remains unchanged across all instances, indicating that this factor is not directly influenced by any of the considered objectives.

To manage the trade-off between inter- and intra-area objectives, the following experiments retain only the most representative intra-area objectives. Based on the results presented above, we set a very low value for the weights α_2 and α_4 and keep only the objectives f_1 and f_3 as representative components of the inter-area objectives.

5.5 The Trade-off between Intra-Area and Inter-Area Objectives

In this experiment set, we retain objectives f_1 , f_3 , and f_5 , setting the weights of the remaining intra-area objectives to 0.01. We begin by assigning equal importance to intra- and inter-area objectives, yielding the weight vector $[\frac{24}{100}; \frac{1}{100}; \frac{24}{100}; \frac{1}{100}; \frac{50}{100}]$. For subsequent instances, we gradually increase α_5 while preserving convexity of the weighted sum, aiming to improve inter-area interaction until full connectivity is achieved across all areas. This approach highlights the trade-off between intra-area workload balance and inter-area connectivity. We report results in Table 4, including percentage scores of the objective components relative to their respective $f_k^{\min}$ and f_k .

■ **Table 4** Results using the weight vector: $[\alpha_1, 0.01, \alpha_3, 0.01, \alpha_5]$, where we gradually increase α_5 against α_1 and α_3 .

instance	r (m)	obj 10^{-3}	f_1 (track)	f_2 (st/end)	f_3 (mov)	f_4 (m^f/m^p)	f_5 (inter-flow)	#D;L areas
$I_{\frac{24}{100}; \frac{1}{100}; \frac{24}{100}; \frac{1}{100}; \frac{50}{100}}$	18	55	35 6%	162889 3%	233640 0%	296445 5%	994792 8%	8;14
$I_{\frac{19}{100}; \frac{1}{100}; \frac{19}{100}; \frac{1}{100}; \frac{60}{100}}$	110	60	35 6%	162889 3%	233640 0%	296445 5%	994792 8%	8;13
$I_{\frac{14}{100}; \frac{1}{100}; \frac{14}{100}; \frac{1}{100}; \frac{70}{100}}$	177	65	35 6%	162889 3%	233640 0%	296445 5%	994793 8%	8;13
$I_{\frac{9}{100}; \frac{1}{100}; \frac{9}{100}; \frac{1}{100}; \frac{80}{100}}$	197	69	35 6%	162889 3%	233928 0%	296445 5%	989691 8%	8;14
$I_{\frac{4}{100}; \frac{1}{100}; \frac{4}{100}; \frac{1}{100}; \frac{90}{100}}$	128	72	59 25%	168227 76%	285503 80%	362536 7%	640258 2%	3;11
$I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$	7	42	143 92%	170024 100%	298272 99%	412464 8%	498353 0%	0;9

In the first four instances, from $I_{\frac{24}{100}; \frac{1}{100}; \frac{24}{100}; \frac{1}{100}; \frac{50}{100}}$ to $I_{\frac{9}{100}; \frac{1}{100}; \frac{9}{100}; \frac{1}{100}; \frac{80}{100}}$, the objective components remain nearly identical, with small differences in total values mainly due to the weight settings. The number of disconnected areas is unchanged, while the number of large areas varies only slightly, from 13 to 14. Only when α_5 reaches $\frac{90}{100}$ do f_1 , f_2 , and f_3 increase substantially, accompanied by improved inter-area flow, confirming the expected trade-off between intra- and inter-area objectives. The homogeneity component, f_4 , is only marginally affected. In the final instance, $I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$, all areas become connected, but at the cost of substantially higher f_1 , f_2 , and f_3 values and fewer large areas.

5.6 The Impact of Restricting the Area Size

Since the number of large areas is not directly steered in our MIP, we decide to add a new constraint that forces all areas to include at least five stations. The new constraint is:

$$\sum_i y_{i,j} \geq 5 \quad \forall j \in A \quad (22)$$

We keep this new constraint in all remaining experiments of this paper. We rerun the same instances reported in Table 4, this time including Constraint (22), and we present the corresponding results in Table 5.

■ **Table 5** Results using same instances (weights) as in Table 4 but with a lower bound on the number of stations per area (at least 5).

instance	r (m)	obj 10^{-3}	f_1 (track)	f_2 (st/end)	f_3 (mov)	f_4 (m^f/m^p)	f_5 (inter-flow)	#D;L areas
$I_{\frac{24}{100}; \frac{1}{100}; \frac{24}{100}; \frac{1}{100}; \frac{50}{100}}$	16	56	35 6%	162889 3%	233640 0%	296445 5%	1001658 8%	8;15
$I_{\frac{19}{100}; \frac{1}{100}; \frac{19}{100}; \frac{1}{100}; \frac{60}{100}}$	53	60	35 6%	162889 3%	233640 0%	296445 5%	1001657 8%	8;15
$I_{\frac{14}{100}; \frac{1}{100}; \frac{14}{100}; \frac{1}{100}; \frac{70}{100}}$	223	65	35 6%	162889 3%	233640 0%	296445 5%	1001657 8%	8;15
$I_{\frac{9}{100}; \frac{1}{100}; \frac{9}{100}; \frac{1}{100}; \frac{80}{100}}$	354	70	35 6%	162889 3%	233928 0%	296445 5%	996219 8%	8;15
$I_{\frac{4}{100}; \frac{1}{100}; \frac{4}{100}; \frac{1}{100}; \frac{90}{100}}$	96	72	61 27%	162889 3%	233928 0%	296019 5%	919008 7%	8;15
$I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$	78	55	62 28%	167362 64%	286330 81%	296019 5%	701051 3%	4;15

Comparing these results with the previous ones, where Constraint (22) was not enforced, we observe that requiring all areas to be large did not significantly affect the individual objectives. Moreover, even increasing α_5 to 0.9 still produces a good solution, with only a modest increase in f_1 . The threshold appears at $\alpha_5 = 0.95$, where we observe a noticeable increase in f_2 and f_3 , accompanied by only a slight improvement in f_5 .

In contrast to Table 4, achieving full area connectivity was not possible even with $\alpha_5 = 0.95$. This is reasonable, since enforcing all areas to be large encourages the model to assign some disconnected stations to the same area. Consequently, achieving full connectivity would require setting α_5 to its maximum value while omitting all intra-area objectives. However, even this may not guarantee full connectivity because of the domain reduction constraints added to the MIP (see Constraints 18, 19, 20, and 21).

5.7 Reducing the Number of Areas

In this set of experiments, we reduce the number of areas used in the model. Instead of partitioning the stations into 15 areas, we decrease this number to 13 and 11. This change requires recomputing the corresponding $f_k^{\min}$ and $f_k^{\max}$ values, making comparisons with the previously computed bounds no longer meaningful. The purpose of these experiments is to investigate whether using fewer areas leads to improvements in the individual objective components. Therefore, rather than comparing normalized objective values, we report the percentage change in each objective component relative to the corresponding instances in Table 5, as shown in Tables 6 and 7.

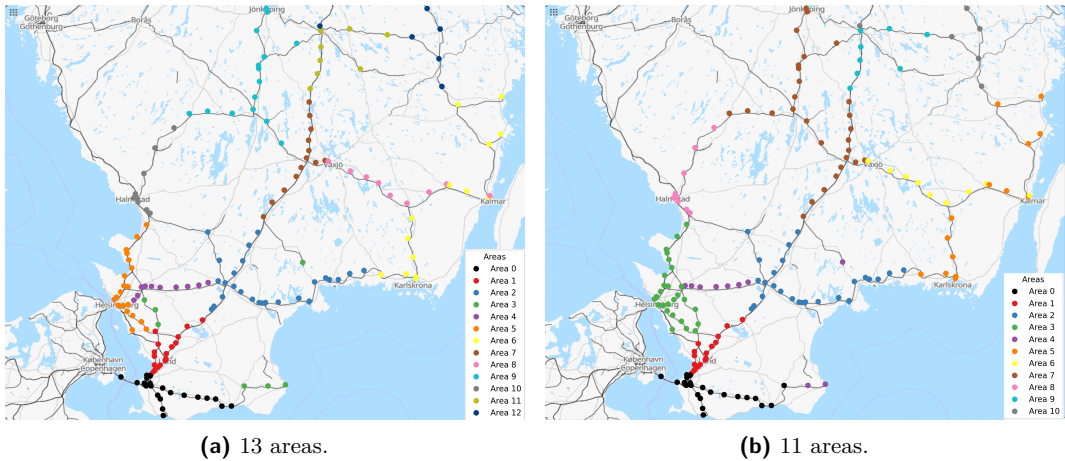
Reducing the number of areas to 13 had a negligible impact on the intra-area objectives, except when α_5 was increased to 0.95. Moreover, the connectivity measure improved remarkably in almost all instances, and the number of disconnected areas decreased to only three in the last instance. The objective f_5 improved in all instances, as expected, since having fewer areas reduces the total number of movements between them. These experiments

■ **Table 6** Results for 13 areas including a percentage comparison with the objective values of the corresponding instances from Table 5. Percentage increases are emphasized in bold text.

instance	r (m)	obj 10^{-3}	f_1 (track)	f_2 (st/end)	f_3 (mov)	f_4 (m^f/m^p)	f_5 (inter-flow)	#D;L areas
$I_{\frac{24}{100}; \frac{1}{100}; \frac{24}{100}; \frac{1}{100}; \frac{50}{100}}$	20	47	35 0%	157750 -3%	233640 0%	296445 0%	879591 -12%	5;13
$I_{\frac{19}{100}; \frac{1}{100}; \frac{19}{100}; \frac{1}{100}; \frac{60}{100}}$	31	49	35 0%	157750 -3%	233640 0%	296445 0%	879591 -12%	5;13
$I_{\frac{14}{100}; \frac{1}{100}; \frac{14}{100}; \frac{1}{100}; \frac{70}{100}}$	28	52	35 0%	157750 -3%	233640 0%	296445 0%	879590 -12%	5;13
$I_{\frac{9}{100}; \frac{1}{100}; \frac{9}{100}; \frac{1}{100}; \frac{80}{100}}$	33	54	35 0%	157750 -3%	233640 0%	296445 0%	879590 -12%	5;13
$I_{\frac{4}{100}; \frac{1}{100}; \frac{4}{100}; \frac{1}{100}; \frac{90}{100}}$	183	54	61 0%	157750 -3%	233640 0%	296019 0%	802831 -13%	4;13
$I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$	22	38	143 130%	165950 -1%	282034 -1%	412464 39%	550718 -21%	3;13

show that reducing the number of areas generally improved connectivity and inter-area flow without a remarkable loss in the intra-area objectives. In Figure 4a, we illustrate the resulting partition for Instance $I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$, where we achieve the highest connectivity without sacrificing the intra-area objectives.

Reducing the number of areas to 11 improved, as also observed with 13 areas, the objective component f_5 and the connectivity measure. In contrast, the objective component f_1 increased in most instances. A plausible explanation is that having fewer areas forces the model to assign more stations per area. In combination with the constraint prohibiting small areas, this likely results in areas with a higher deviation in the number of tracks from the maximum ℓ_1^{max} , which is captured by the objective component f_5 . In Figure 4b, we illustrate the resulting partition for Instance $I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$, using 11 areas.



■ **Figure 4** Resulting partitions for Instance $I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$.

■ **Table 7** Results for 11 areas including a percentage comparison with the objective values of the corresponding instances from Table 5. Percentage increase are emphasized in bold text.

instance	r (m)	obj 10^{-3}	f_1 (track)	f_2 (st/end)	f_3 (mov)	f_4 (m^f/m^p)	f_5 (inter-flow)	#D;L areas
$I_{\frac{24}{100}; \frac{1}{100}; \frac{24}{100}; \frac{1}{100}; \frac{50}{100}}$	5	49	47 32%	157750 -3%	233640 0%	328151 10%	842493 -15%	5;11
$I_{\frac{19}{100}; \frac{1}{100}; \frac{19}{100}; \frac{1}{100}; \frac{60}{100}}$	6	51	47 32%	157750 -3%	233640 0%	319490 8%	853193 -15%	5;11
$I_{\frac{14}{100}; \frac{1}{100}; \frac{14}{100}; \frac{1}{100}; \frac{70}{100}}$	6	51	48 37%	157750 -3%	233640 0%	323361 5%	833488 -17%	4;11
$I_{\frac{9}{100}; \frac{1}{100}; \frac{9}{100}; \frac{1}{100}; \frac{80}{100}}$	6	50	55 57%	157750 -3%	233640 0%	265753 -10%	781092 -22%	4;11
$I_{\frac{4}{100}; \frac{1}{100}; \frac{4}{100}; \frac{1}{100}; \frac{90}{100}}$	3	46	59 -3%	157750 -3%	233640 0%	290021 -2%	766461 -17%	4;11
$I_{\frac{3}{200}; \frac{1}{100}; \frac{3}{200}; \frac{1}{100}; \frac{95}{100}}$	3	38	69 11%	165708 -1%	277652 -3%	366409 24%	625022 -11%	3;11

6 Conclusions and Future Work

In this paper, we presented a MIP, formulated as a multi-objective optimization problem, for partitioning railway stations into a predefined number of dispatching areas for railway dispatching centers. The proposed approach jointly considers two key aspects of operational performance: the coordination effort required between neighboring dispatching areas, represented by inter-area interaction complexity, and the workload within each dispatching area, represented by intra-area operational complexity. By integrating infrastructure characteristics and traffic-related factors into a unified optimization framework, we provide a systematic and quantitative methodology for designing balanced dispatching areas.

We applied the proposed model to instances from the Malmö dispatching center in the southern Swedish railway network, demonstrating its applicability to realistic railway operations. The results reveal a trade-off between the intra-area objectives, which promote a more balanced workload within areas, and the inter-area objective, which promotes lower coordination effort and greater connectivity between areas.

Since our MIP includes multiple objective components, selecting appropriate weights is crucial. Determining these weights is typically the responsibility of network managers, as they reflect the relative importance of each objective. However, in our experiments, we evaluated a range of weight combinations to identify the most representative ones. This analysis can support decision-makers by providing guidance on how different weight settings affect the solutions, making it easier to prioritize among the competing objectives.

Other factors that may affect solution quality are the area size and the number of areas. In several instances, we showed that requiring all areas to be large had no noticeable impact on the different objective components. Furthermore, a moderate reduction in the number of areas improved inter-area measures and area connectivity without adversely affecting the intra-area objectives.

A key contribution of this work is showing that an exact method solves the sectorization problem within 1% of optimality, with all experiments completed in under 360 minutes. In contrast, Zhao et al. [18] aggregate intra-area workload into a single value and minimize deviations from its average, considering only one objective and obtaining heuristic solutions

in under 33 minutes (though this comparison should be interpreted with caution, as computation times are hardware-dependent and were not measured on the same machine). When considering a single objective, our exact method obtains optimal solutions in at most 23 seconds (see Table 2).

In this paper, we showed that optimization-based sectorization can support redesigning dispatching areas by explicitly balancing the trade-off between minimizing coordination across boundaries and maintaining manageable operational complexity within each area. Furthermore, robustness analysis under different traffic compositions and demand levels indicates the framework remains applicable under varying conditions, making it a promising decision-support tool for dispatching area planning.

For future work, several extensions to the proposed framework are possible. In this study, the number of areas is fixed. Allowing the model to decide the number of areas would make it more flexible by jointly optimizing both structure and size. However, this would increase computational complexity and may require decomposition methods or heuristics.

Moreover, since weighted-sum scalarization identifies only supported Pareto-optimal solutions, future work could apply alternative methods – such as the ϵ -constraint method – to characterize the complete Pareto front, including unsupported solutions, offering decision-makers a more comprehensive view of trade-offs between intra- and inter-area objectives.

Finally, applying the method to large national networks (e.g., the Swedish network) is computationally challenging. Aggregation or hierarchical models – merging low-importance stations into larger nodes – may help, while the main network structure is preserved.

References

- 1 Micah Altman. Is automation the answer: The computational complexity of automated redistricting. *Rutgers Computer and Law Technology Journal*, 23(1):81–141, 1997.
- 2 Emna Benzarti, Evren Sahin, and Yves Dallery. Operations management applied to home care services: Analysis of the districting problem. *Decision Support Systems*, 55(2):587–598, 2013. doi:10.1016/J.DSS.2012.10.015.
- 3 Burcin Bozkaya, Erhan Erkut, Dan Haight, and Gilbert Laporte. Designing new electoral districts for the city of edmonton. *Interfaces*, 41(6):534–547, 2011. doi:10.1287/INTE.1110.0544.
- 4 Miguel Camacho-Collados and Federico Liberatore. A decision support system for predictive police patrolling. *Decision support systems*, 75:25–37, 2015. doi:10.1016/J.DSS.2015.04.012.
- 5 Jelena Djokic, Bernd Lorenz, and Hartmut Fricke. Air traffic control complexity as workload driver. *Transportation research part C: emerging technologies*, 18(6):930–936, 2010.
- 6 Hamideh Etemadnia, Khaled Abdelghany, and Ahmed Hassan. A network partitioning methodology for distributed traffic management applications. *Transportmetrica A: Transport Science*, 10(6):518–532, 2014.
- 7 Jacques A Ferland and Gilles Gu  nette. Decision support system for the school districting problem. *Operations Research*, 38(1):15–21, 1990. doi:10.1287/OPRE.38.1.15.
- 8 David Gianazza. Analysis of a workload model learned from past sector operations. In *SID 2017, 7th SESAR Innovation Days*, 2017.
- 9 Tobias Andersson Granberg, Tatiana Polishchuk, Valentin Polishchuk, and Christiane Schmidt. A novel mip-based airspace sectorization for tmas. In *12th International seminar for Air Traffic Management Research and Development (ATM R&D)*, 2017.
- 10 J  rg Kalcsics and Roger Z R  os-Mercado. Districting problems. In *Location science*, pages 705–743. Springer, 2020.
- 11 Alain Labelle, Andr   Langevin, and James F Campbell. Sector design for snow removal and disposal in urban areas. *Socio-Economic Planning Sciences*, 36(3):183–202, 2002.

- 12 Joe Mitchell, Girishkumar Sabhnani, Robert Hoffman, Jimmy Krozel, and Arash Yousefi. Dynamic airspace configuration management based on computational geometry techniques. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7225, 2008.
- 13 Elif Göksu Öztürk, Pedro Rocha, Ana Maria Rodrigues, José Soeiro Ferreira, Cristina Lopes, Cristina Oliveira, and Ana Catarina Nunes. D3s: Decision support system for sectorization. *Decision Support Systems*, 181:114211, 2024. doi:10.1016/J.DSS.2024.114211.
- 14 B Sridhar, K Sheth, and S Grabbe. Measures for air traffic controller workload prediction. In *2nd USA/Europe Air Traffic Management 2000 Research and Development Seminar*, 2000.
- 15 Dian Wang, Jun Zhao, Andrea D'Ariano, and Qiyuan Peng. Simultaneous node and link districting in transportation networks: Model, algorithms and railway application. *European Journal of Operational Research*, 292(1):73–94, 2021. doi:10.1016/J.EJOR.2020.10.013.
- 16 Min Xue. Airspace sector redesign based on voronoi diagrams. *Journal of Aerospace Computing, Information, and Communication*, 6(12):624–634, 2009. doi:10.2514/1.41159.
- 17 Arash Yousefi and George Donohue. Temporal and spatial distribution of airspace complexity for air traffic controller workload-based sectorization. In *AIAA 4th aviation technology, integration and operations (ATIO) forum*, page 6455, 2004.
- 18 Jun Zhao, Dian Wang, and Qiyuan Peng. 2018 train dispatcher desk districting problem in high-speed railway network. In *14th International Conference on Advanced Systems in Public Transport (CASPT)*, 2018.
- 19 Xiang Zou, Peng Cheng, Bang An, and Jingyan Song. Sectorization and configuration transition in airspace design. *Mathematical Problems in Engineering*, 2016(1):6048326, 2016.

From Routes to Resources: A Model for Real-Time Train Dispatching with No-Wait Constraints

Luka Stärk  

Zuse Institute Berlin, Germany
Freie Universität Berlin, Germany

Ralf Borndörfer  

Zuse Institute Berlin, Germany
Freie Universität Berlin, Germany

Niels Lindner  

Zuse Institute Berlin, Germany
Freie Universität Berlin, Germany

Abstract

Optimization for train dispatching promises to reduce delays. Yet scalable applications are challenging. We present an efficient MIP model, based on generalized disjunctive graphs, for the train dispatching problem with no-wait constraints. For each train, we model its possible movements with microscopic detail as a DAG of operations. We select one train path per DAG and associate it with reservation and release times for the exclusive infrastructure it uses, such that the resulting schedule is conflict-free. When the connecting velocity between consecutive operations is positive, no-wait constraints apply. Additionally, we give a procedure to transform realistic infrastructure constraints based on route exclusions into an equivalent representation on resources by means of edge clique covers and incorporation of overlap exclusions. This makes resource-based dispatching models applicable to real-life infrastructure. We embed the MIP into an iterative algorithm, which we evaluate on the DISPLIB benchmark set and on a real-life dataset from German dispatching regions that we publish together with this paper. We prove optimality for 70 of the 112 DISPLIB instances within short running times, and we find 27 new best known solutions within a 10-minute time limit.

2012 ACM Subject Classification Applied computing → Transportation; Mathematics of computing → Combinatorial optimization; Theory of computation → Mixed discrete-continuous optimization

Keywords and phrases Train Dispatching, No-Wait Constraints, Disjunctive Graph, Edge Clique Cover, Microscopic Railway Timetabling

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.6

Supplementary Material *Dataset:* <https://doi.org/10.5281/zenodo.21775303> [17]

Funding The work for this article has been conducted in the Research Campus MODAL funded by the German Federal Ministry of Research, Technology and Space (BMFTR), fund number 05M25KEM.

Luka Stärk: Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/2, project ID: 390685689).

Acknowledgements We thank our industrial partner DB InfraGO AG, in particular the ADA-PMB team, for the kind permission to publish our benchmarking set based on real data. We also thank the three anonymous reviewers, whose comments improved the presentation of this paper.

1 Introduction

Real-time train dispatching is the continuous task of resolving conflicts and minimizing delays in an ever-evolving traffic situation where decisions must be made within seconds. This challenge arises when the carefully planned conflict-free timetable meets the unpredictable



© Luka Stärk, Ralf Borndörfer, and Niels Lindner;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 6; pp. 6:1–6:20



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

reality of delays and disruptions. Dispatchers and signalers are then responsible for making decisions to keep operations running and to prevent delays from spreading to other trains. Most railway networks are separated into dispatching regions where a group of dispatchers supervises the traffic in their region. The primary objective of dispatchers is to minimize delays by reordering and rerouting trains. Optimization-based decision-support tools for train dispatching have the potential to improve this process significantly by increasing the speed and quality of dispatching decisions and by considering solutions and their consequences over longer time periods where delays may propagate.

We consider the train dispatching problem with microscopic feasibility, velocity profiles (acceleration/deceleration curves), and infrastructure use that depends on the position and velocity of a train [18]. For each train, we build a directed acyclic graph of operations. If such an operation reflects a profile ending with positive velocity, no-wait constraints apply. To provide a physically feasible schedule, these constraints are crucial: a stopping train may not only change the release time of the last released resources, but also increase the running time during its acceleration for the next operation after stopping. Furthermore, the set of required resources of an operation depends on the velocity profile for that operation, as the train reserves the next route earlier when driving at higher velocity.

In this work we present three contributions. First, we show how to transform infrastructural exclusions of routes and overlaps into an abstract resource representation. Second, we define TDP-NW, the *Train Dispatching Problem with No-Wait Constraints*, which we model as a MIP based on generalized disjunctive graphs. The core is to select one path of operations per train and the timings along this path such that the selection is conflict-free w.r.t. the resources. In particular, we allow for paths that reflect re-routing of trains. Our resource representation of the route exclusions allows for a simple conflict-detection oracle, by checking for overlapping time intervals per resource. Introducing dedicated reservation and release nodes, whose timing is determined by level-aggregated resource timing constraints, we develop a MIP formulation and an iterative algorithm that is capable of effectively solving realistic instances at microscopic detail. Third, we evaluate our methods on the DISPLIB instances [9] and on a curated dataset of TDP-NW instances based on three German dispatching regions that we make publicly available [17].

We discuss some related work in Section 2. Our transformation from route-exclusion matrices to resources is described in Section 3. The central problem TDP-NW is defined in Section 4, followed by its MIP formulation in Section 5. Our benchmark set and computational results are illustrated in Section 6, before we conclude the paper in Section 7.

2 Related Work

While significant research exists on train dispatching [2], only recently have mathematical optimization algorithms been deployed in real-life applications, such as those of [18, 3]. For the few real-time dispatching algorithms we can see in the industry, their major challenge is scalability in time and space, while ensuring a short optimization time. The spatial dimension refers to the size of the infrastructure segment under consideration.

Definitions of train dispatching problems with and without rerouting in the literature mostly apply formulations of Job-Shop Scheduling Problems with blocking constraints [14]. Different optimization techniques have been explored for solving this problem. The first custom branch-and-bound algorithm restricted to reordering was proposed by D’Ariano et al. [4]. Many approaches use MIP formulations that rely on the big- M trick for disjunctive constraints. To avoid big- M constraints, a non-compact MIP formulation has been developed

that iteratively adds constraints for paths and cycles [10]. The literature becomes sparser when it comes to rerouting. Only recent works have investigated the problem with routing alternatives [11, 9, 16, 1, 5]. Rerouting alternatives are crucial for real-time train dispatching optimization. However, they may lead to increased complexity. One way to reduce this complexity in practice is to use a macro-micro decomposition, where a subnetwork, such as a station, is collapsed into a macroscopic resource that can accommodate more than one train at a time [13].

3 From Routes to Resources

As in related work based on job-shop scheduling, our train dispatching model in Section 4 relies on an infrastructure model, where a train reserves and releases exclusive *resources* along its path. Automatic train protection systems such as ETCS (European Train Control System) implement exclusive infrastructure usage defined for *routes*. A route, also called *block* on the free track, is a directed path between two main signals. A train reserves a route, e.g., before it reaches the EBI (emergency brake intervention point) towards the main signal, and releases or partially releases a route once the train passes a release contact with its last axle. The information which routes cannot be used simultaneously is often given by *route-exclusion matrices*, defining infrastructure resources only implicitly.

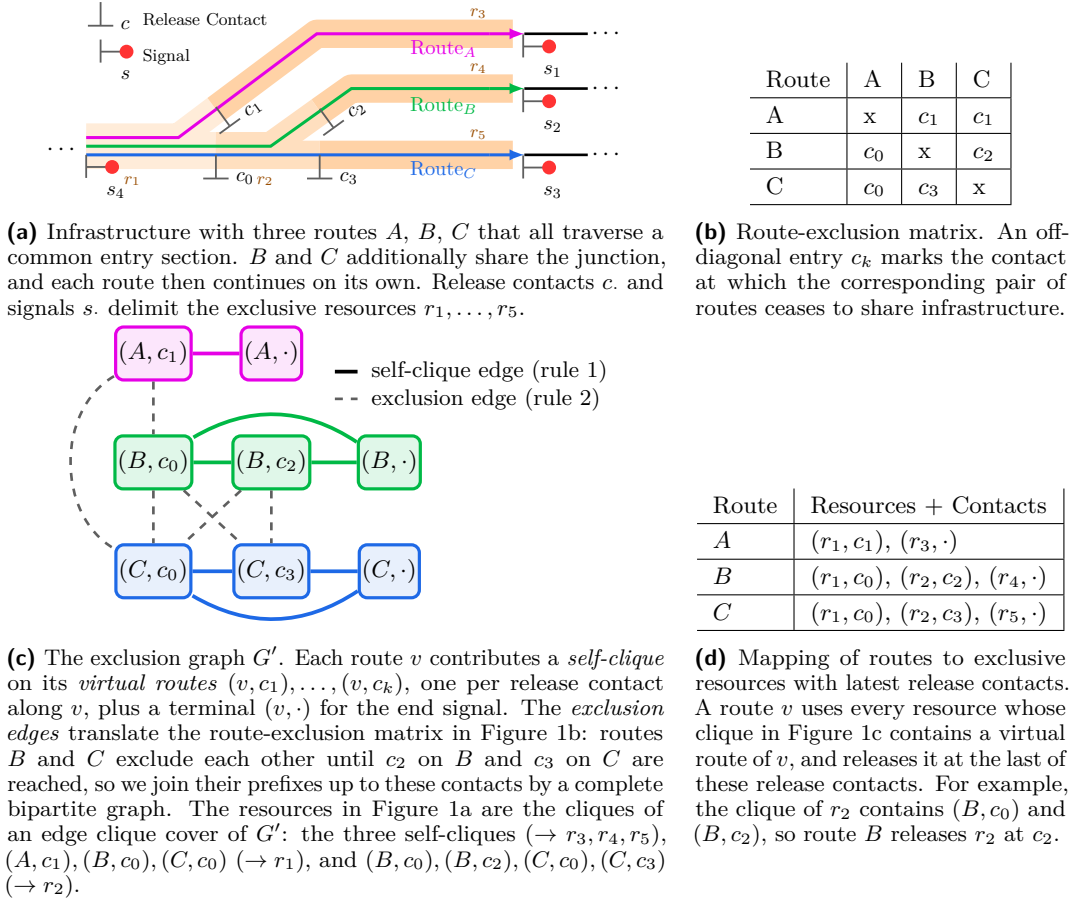
In this section, we describe a procedure to convert route-exclusion matrices to *resources*, hence offering a flat representation of exclusive infrastructure, in which every resource may be used by at most one train at a time. This not only enables us to encapsulate a microscopic level, but also simplifies the check for conflicting resource allocations. Instead of evaluating the route-exclusion matrix for every pair of excluding routes, we collect all overlapping time intervals per resource.

Example. Before providing the formal details, we will first illustrate the conversion of routes to resources in a small example, see Figure 1.

Overlaps. Route exclusions can mostly be derived from physical track intersections. However, there are special cases such as exclusions due to flank protection and overlap exclusions [15]. A train that will stop in front of a main signal additionally claims an *overlap*: a safety margin of track beyond the signal that must be free before the train is allowed to reserve this route, so that it can come to a safe halt even if it overruns the signal. The overlap is released once the train comes to a halt. Figure 2 shows an example situation of two disjoint parallel routes *A* and *B*, e.g., two station tracks. The sections from the signals through the switch up to the release contact c_0 are the *overlaps* of routes *A* and *B*: they are the safety overlap a train of route *A* or *B* claims when it intends to stop at s_1 or s_2 , and at the same time the shared first part of routes *C* and *D*. These two overlaps will become an *overlap resource* r_o in the resource representation for the overlap routes of *A* and *B*. Unlike an ordinary resource usage, an overlap reservation blocks other trains from using r_o as part of their *route*, but it does not prevent another train from claiming the same section as its *overlap*.

3.1 Minimum Resource Representation via Edge Clique Cover

We formalize the problem of computing a minimum set of resources from a route-exclusion matrix as follows. We are given a finite set of release contacts C and a symmetric directed graph $G = (V, A)$, where the nodes V are the routes and the arcs A encode the exclusion relations. The graph is symmetric in the sense that $(u, v) \in A \iff (v, u) \in A$. An arc



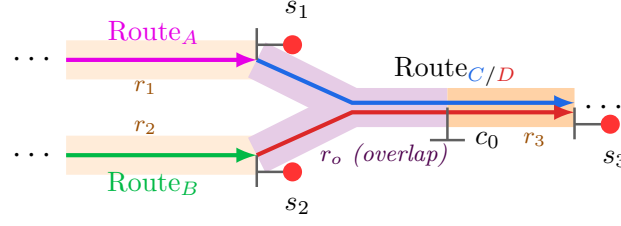
■ **Figure 1** Route to resources conversion for three routes A , B , C sharing track sections.

labeling $\rho: A \rightarrow C \cup \{\cdot\}$ gives, for an arc (u, v) , the release contact $\rho(u, v)$ at which a train on route u (partially) releases its exclusion with route v ; the special symbol $\cdot$ denotes a part of a route that is held until its end signal (more accurately until a release contact after the main signal is passed by the train, which is an implementation detail). Note that the graph is symmetric in structure but the release contacts of the arcs are not, i.e., in general $\rho(u, v) \neq \rho(v, u)$. For each route v we are further given the sequence $\sigma(v) = (c_1, \dots, c_k, \cdot)$ of its partial release contacts, ordered along the direction of travel, and where we append a terminal element “ $\cdot$ ”. We write $c \preceq_v c'$ for the induced order. Finally, let $V_o \subseteq V$ be the set of overlap routes and $V_r = V \setminus V_o$ be the set of regular routes, where each overlap route is associated to exactly one regular route.

Virtual routes. We construct an auxiliary undirected graph $G' = (V', A')$. For every regular route $v \in V_r$ we create a *virtual route* $(v, c) \in V'$ for each $c \in \sigma(v)$. For convenience we write $\text{pre}(v, c) := \{(v, c') \in V' : c' \preceq_v c\}$ for the *prefix* of virtual routes of v up to and including c .

Translating exclusions. The edges A' are derived from A by two rules:

1. (*self*) For every regular route $v \in V_r$, all of its virtual routes $\text{pre}(v, \cdot)$ form a clique in G' .
2. (*exclusion*) For every exclusion pair $\{(u, v), (v, u)\} \subseteq A$ between regular routes, with release contacts $c_u = \rho(u, v)$ and $c_v = \rho(v, u)$, we add all edges of the complete bipartite graph between $\text{pre}(u, c_u)$ and $\text{pre}(v, c_v)$.



■ **Figure 2** Overlap routes and resources. Routes A and B end at signals s_1 and s_2 ; the continuation routes C and D run from s_1 and s_2 through the switch to the common exit signal s_3 . The release contact c_0 lies behind the switch. Each incoming route is associated with a resource (r_1, r_2) ; the section from the signals to c_0 is an overlap resource r_o for A and B ; the outgoing track from c_0 to s_3 is r_3 .

Because each prefix is already a clique (rule 1), rule 2 turns the entire shared region $\text{pre}(u, c_u) \cup \text{pre}(v, c_v)$ into a clique, representing the whole piece of infrastructure two routes share. Note that we can include the prefixes, as the full route must be reserved, and then it can be partially released at the release contacts. To preserve all exclusions, every edge of G' has to be covered by at least one resource, which is an *edge clique cover* of G' : a set of cliques such that every edge is contained in at least one of them. To obtain a representation with a minimum number of resources, we are therefore interested in finding an edge clique cover of G' of minimum size.

Realizing overlap exclusions. An overlap exclusion is an extension beyond the end signal of the regular route, excluding routes to which the regular route has no exclusive relationship. Overlap exclusions apply when a train plans to stop at the end of the route. To cover the overlap exclusions between routes, we create a set S_{v_o} of excluded virtual routes for each overlap route $v_o \in V_o$. If there is an exclusion with route u , we add the virtual routes $\text{pre}(u, c)$ with $c = \rho(u, v_o)$ to S_{v_o} . However, an overlap route usually has several overlap exclusions, so that an overlap route may be connected to several cliques in an edge clique cover, but not necessarily to all virtual routes in any such clique. If the overlap exclusions of v_o do not exclude the whole clique, we cannot assign the clique (resource) to the overlap route; instead we need to search for a clique that is a subset of the excluded virtual routes, or we create a new sub-clique. As an overlap route has no virtual routes of its own, it is never a member of a clique, but only receives cliques as overlap resources. Since there are no overlap-overlap exclusions, we do not need to worry about accidentally creating a new exclusion between overlap routes when assigning cliques as overlap resources. Later, when checking for resource conflicts between trains by comparing the reservation intervals, we simply ignore intersecting intervals where both usages are overlap usages. The problem of finding a minimum representation of overlap resources on top of an edge clique cover can hence be described as follows, where each overlap route forms a *target set* of its excluded virtual routes:

► **Definition 1** (Minimum Sub-Clique Cover Extension). Let $G' = (V', A')$ be the exclusion graph with an edge clique cover $\mathcal{C}$, and let $S = \{S_1, \dots, S_m\}$ be a family of target sets $S_i \subseteq V'$. For each S_i find a set H_i of cliques of G' such that

- every clique is internal to its target set, i.e. $C \subseteq S_i$ for all $C \in H_i$.
 - the union of cliques covers the target set, $\bigcup_{C \in H_i} C = S_i$, i.e. $\forall v \in S_i \exists C \in H_i : v \in C$.
- Cliques already present in $\mathcal{C}$ may be reused at no cost. Minimize the number of newly created sub-cliques, counted globally across all targets, $|\bigcup_i H_i \setminus \mathcal{C}|$.

3.2 Algorithm From Routes To Resources

Our algorithm from routes to resources is a two-stage optimization with the objective to minimize the number of resources to represent the given route exclusions. We first compute a small edge clique cover of G' ; in practice our instances are solved completely (and optimally) using only the data-reduction rules 1 and 2 of [6]. Rule 1 removes isolated vertices and vertices with only covered edges, and rule 2 applies if, for an uncovered edge (u, v) , the common neighborhood of u and v induces a clique C ; that clique C is then added to the cover. As in Figure 1c, each clique becomes one exclusive resource, and a route v receives a resource for every clique that contains one of its virtual routes. The release contact recorded for that resource on v is the latest contact c with (v, c) in the clique, cf. Figure 1d. Then, we solve the Minimum Sub-Clique Cover Extension problem greedily by covering each target set S , corresponding to an overlap route v_o , with cliques, either reusing an existing clique that is contained in S , or otherwise creating a new sub-clique from the intersection of S with an existing clique that covers the most still-uncovered routes of S , until v_o shares a common clique with every excluded virtual route in S . The cliques (resources) referenced this way are labeled as overlap resources for v_o .

To put our approach into perspective, Table 1 shows, for three dispatching regions, the size of the exclusion graph G' and the number of cliques (resources) found. The edge clique cover is solved completely by the reduction rules alone, and the overlap exclusions add a moderate number of extra sub-cliques on top of it. This comes as no surprise, since the exclusions are local relations, and G' typically consists of nearly disconnected components, representing, e.g., stations.

■ **Table 1** Size of the exclusion graph G' and number of exclusive resources (cliques) for three dispatching regions, showing the number of routes, overlap routes, and virtual routes $|V'|$. The edge clique cover (ECC) is computed by the data-reduction rules of [6]. Covering the overlap exclusions adds the sub-cliques, yielding the total number of exclusive resources. Note that these numbers do not reflect a whole dispatching region but only the routes used in a specific instance. The instances used as examples are `fbzm_large_dh_90_0`, `mbzs_large_dh_60_0`, `rbzs_dh_90_0` [17].

Region	Routes	Overlap routes	$ V' $	ECC cliques	Sub-cliques	Total
FBZM	1160	1342	4547	946	525	1471
MBZS	336	248	934	289	156	445
RBZS	841	851	3330	723	388	1111

4 Problem Definition

We consider the *Train Dispatching Problem with No-Wait Constraints* (TDP-NW), which can be seen as a variation of the DISPLIB problem defined in [9] and to some degree as a generalization. An instance consists of a set of trains I that share a common set of resources $U = \{0, \dots, |U| - 1\}$. Each train must be routed and scheduled through its own graph of operations, and the schedules of different trains interact only through their competition for the exclusive resources. The task is to choose, for every train, a path P^i together with a timing $t_v \in \mathbb{R}$ of its operations $v \in P^i$ that avoids resource conflicts while minimizing delay and operation costs.

Operations. The possible movements of a train i are modeled by a directed acyclic graph $G^i = (V^i, A^i)$ whose nodes $v \in V^i$ are the *operations* of the train and whose arcs $(v, u) \in A^i$ encode the successor relation. Each train has a single source operation v_0^i and target operation $v_{n_i}^i$, and a train path is a $v_0^i - v_{n_i}^i$ path in G^i . An operation v has the following attributes:

- a *duration* $\delta_v \in \mathbb{R}_{\geq 0}$, the running time of the operation;
- optional *start-time bounds* $[\alpha_v, \beta_v]$ on the time $t_v \in \mathbb{R}$ at which the operation v starts;
- a *penalty* $\lambda_v \in \mathbb{N}_0$ incurred whenever v lies on the chosen path;
- a *no-wait* flag: if set, every selected successor u must start exactly when v ends, $t_v + \delta_v = t_u$; otherwise waiting is allowed, $t_v + \delta_v \leq t_u$;
- a set of *resource usages*.

Reservations, releases, resource usages. An operation v interacts with the exclusive resources through *resource usages*. A resource usage refers to a resource $r_x \in U$ and may carry a *reservation* offset $\delta_v^{\text{rv}} \in \mathbb{R}$ and/or a *release* offset $\delta_v^{\text{rl}} \in \mathbb{R}_{\geq 0}$, given relative to the start time t_v of the operation: at $t_v + \delta_v^{\text{rv}}$ the train reserves r_x , and at $t_v + \delta_v^{\text{rl}}$ it releases it again. We assume that, along any route, every reservation of a resource has a matching release downstream (or the resource stays reserved until the target operation, modeling indefinite occupation), and every release has a matching reservation upstream. A resource that is never reserved on the chosen path is unused. After a release it may happen that the same resource is reserved again, for instance when a train reverses direction. A usage may additionally be flagged as an *overlap* (see Section 3).

Objective components. Each train has a set $\mathcal{D}^i$ of *objective components*. An objective component is defined for a set of mutually alternative operations, i.e., operations forming a source-to-target node-cut of G^i , such that exactly one of them lies on any train path, or the source or target itself. Each objective component c has a *threshold* $\theta_c \in \mathbb{R}$ and a *coefficient* $\gamma_c \in \mathbb{R}_{\geq 0}$. Let v be the operation of the component on the chosen train path. The component's cost is then

$$\text{cost}_c := \gamma_c \cdot \max(0, t_v + \delta_v - \theta_c). \quad (1)$$

Note that the delay is measured at the arrival of v and not the departure, which is a difference compared to the DISPLIB definition [9]. The reason is that we have different durations for the arrival operations but a single threshold per objective component, e.g., the trains' planned arrival time. We cannot measure the delay at the successor operations, since a train may wait longer than the operations already account for, e.g., longer than the minimum dwell time at a commercial stop. Alternatively, one could introduce an operation for the stop with the duration of the minimum dwell time.

► **Definition 2** (Train Dispatching Problem with No-Wait Constraints (TDP-NW)). *Select, for each train, a path from source to target in the DAG and assign finite start times to the selected operations such that:*

1. **Time windows** – start time bounds $\alpha_v \leq t_v \leq \beta_v$ are respected.
2. **Successor constraints** – for no-wait arcs: $t_v + \delta_v = t_u$; otherwise: $t_v + \delta_v \leq t_u$.
3. **Exclusive resource usage** – each resource $r_x \in U$ is used by at most one train at a time.
4. **Optimality** – the sum of the costs of all objective components and all operation penalties is minimized.

Even the decision variant of TDP-NW is NP-complete: restricted to a single path per train, it is job-shop scheduling with blocking and no-wait constraints, for which deciding whether a feasible solution exists is NP-complete [14].

5 Methodology

In the following we describe how we model the TDP-NW as a compact MIP using the terminology of *generalized disjunctive graphs* [12]. We consider a generalized disjunctive graph $D = (V \cup V_{rv} \cup V_{rl}, A \cup A_{lb} \cup A_{rv} \cup A_{rl} \cup A_d)$, where V are the operations, V_{rv} and V_{rl} the reserving and releasing nodes, respectively. Moreover, A are the successor arcs, A_{lb} the lower-bound arcs, and A_{rv} and A_{rl} the reserving and releasing arcs, and finally A_d are the disjunctive or alternative arcs with both endpoints in $V_{rv} \cup V_{rl}$. The disjunctive graph is generalized in the sense that we select a source-to-target path for each train in addition to the disjunctive arcs. Given such a selection, the maximum bound applies among the incoming arcs of each node, as in the standard disjunctive graph.

Rather than tracking exclusive resource usage for each operation, we reason over each *resource conflict* through dedicated reservation and release time variables for each $v^{\text{in}} \in V_{rv}$ and $v^{\text{out}} \in V_{rl}$, see Figure 3. This construction was inspired by [12], where auxiliary variables model how resource conflicts depend on routing decisions in a disjunctive program, but this yields one variable per alternative operation. The same idea of auxiliary variables for a resource's reservation and release has been developed by [5], using the notion of chunks of operations with the same resource. For our MIP model we anchor start times at the *level* (topological depth) of each train's DAG rather than at individual operations, see Figure 4. Both choices substantially reduce the number of continuous variables, especially since the instances under study contain many alternative operations with very similar resource usages.

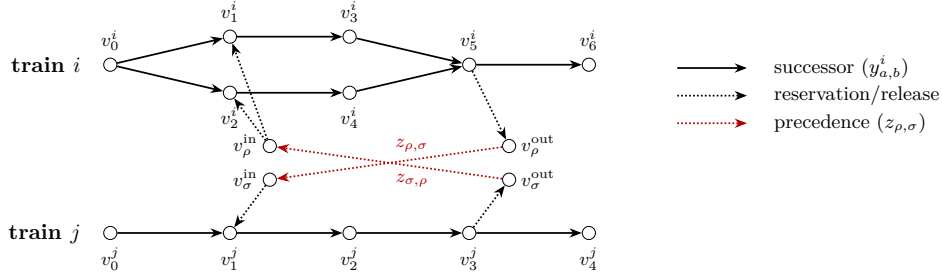


Figure 3 Resource reservation (v_ρ^{in}) and release (v_ρ^{out}) nodes for two conflicting resource usages ρ, σ of trains i, j .

Additional notation. We reuse the notation of Section 4.

- For a vertex v of a directed graph, $N^+(v) = \{u : (v, u) \in A\}$ denotes its *out-neighborhood* and $N^-(v) = \{u : (u, v) \in A\}$ its *in-neighborhood*.
- $S_a^i = \{v_b^i : (v_a^i, v_b^i) \in A^i\}$, $P_a^i = \{v_b^i : (v_b^i, v_a^i) \in A^i\}$ – the successors/predecessors of v_a^i .
- $A^{i,0} \subseteq A^i$ – the *waiting* arcs (e.g., leaving an operation with zero velocity); the remaining arcs $A^i \setminus A^{i,0}$ are *no-wait* arcs and enforce $t_b^i = t_a^i + \delta_a^i$.
- $\ell(v_a^i) \in \{0, \dots, L^i\}$ – the *level* of v_a^i , its topological depth in G^i ; all operations of the same commercial stop share a level. $V_l^i = \{v_a^i \in V^i : \ell(v_a^i) = l\}$ is the set of operations at level l , and $V_l^{i,nw} \subseteq V_l^i$ those with a no-wait out-arc. A level l is *optional* if some arc skips it, see also Figure 4.
- A *resource usage* of train i is a triple $\rho = (r, o, k)$, where $r(\rho) \in U$ is the resource, $o(\rho) \in \{0, 1\}$ its overlap flag, and $k(\rho) \in \mathbb{N}$ its *ordinal*: among all source-to- v -paths reaching a reserving operation v of the pair (r, o) , the ordinal is the maximum number of

re-reservations on any path. The ordinal distinguishes repeated reservations, e.g., a track section released and reserved again. We write R^i for the set of resource usages of train i , $\mathcal{R} = \bigsqcup_{i \in I} R^i$ for all usages across trains, and $i(\rho)$ for the train of a usage $\rho \in \mathcal{R}$.

- For each $\rho \in \mathcal{R}$, we dynamically create a single *reserving node* $v_\rho^{\text{in}} \in V_{rv}$ and *releasing node* $v_\rho^{\text{out}} \in V_{rl}$. The *reserving* and *releasing* operations, which anchor the reservation and release time of ρ , are denoted $\mathcal{R}_\rho^{\text{in}} = N^+(v_\rho^{\text{in}})$ and $\mathcal{R}_\rho^{\text{out}} = N^-(v_\rho^{\text{out}})$, respectively.
- $\mathcal{K}$ – the set of resource conflicts. A conflict $\{\rho, \sigma\} \in \mathcal{K}$ couples two usages $\rho, \sigma \in \mathcal{R}$ of different trains, $i(\rho) \neq i(\sigma)$, on the same resource $r(\rho) = r(\sigma)$ that are not both overlaps, i.e. $\neg(o(\rho) \wedge o(\sigma))$.
- $\mathcal{D} = \bigsqcup_{i \in I} \mathcal{D}^i$ – the objective components of Section 4 across all trains, where each component c belongs to a train $i(c)$. We write component $c \in \mathcal{D}$ as $(\ell_c, \mathcal{V}_c, \theta_c, \gamma_c)$: its anchor level ℓ_c , the operation cut $\mathcal{V}_c \subseteq V_{\ell_c}^{i(c)}$, the threshold θ_c , and the unit delay cost $\gamma_c \geq 0$.

5.1 MIP Model

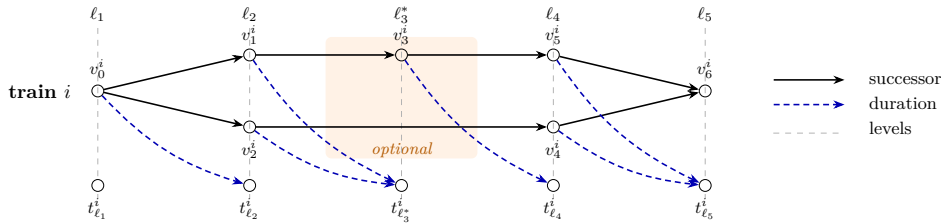
We express the disjunctive graph and the timings of the solution using the following variables:

- $t_l^i \in \mathbb{R}$, $\alpha_l^i \leq t_l^i \leq \beta_l^i$ – start time of level l of train i , where $\alpha_l^i := \min_{v \in V_l^i} \alpha_v$ and $\beta_l^i := \max_{v \in V_l^i} \beta_v$ are the level start-time bounds.
- $y_{a,b}^i \in \{0, 1\}$ – equals 1 iff arc $(v_a^i, v_b^i) \in A^i$ is used on its selected train path.
- $t_\rho^{\text{in}}, t_\rho^{\text{out}} \in \mathbb{R}$ – the reservation/release time of usage ρ , for each ρ appearing in a conflict.
- $x_\rho \in \{0, 1\}$ – equals 1 iff train $i(\rho)$ realizes usage ρ on its chosen path; introduced only when ρ is *optional* (otherwise $x_\rho \equiv 1$).
- $z_{\rho, \sigma} \in \{0, 1\}$ – equals 1 iff ρ is released before σ is reserved, for each conflict $\{\rho, \sigma\} \in \mathcal{K}$ (one per ordered pair).
- $w_c \in \mathbb{R}_{\geq 0}$ – the delay incurred by component $c \in \mathcal{D}$.

Unlike the classical disjunctive formulation, we do not introduce an explicit operation-selection variable x_a^i . As the flow constraints below force exactly one out-arc for any selected non-target operation, we write the selection of v_a^i as

$$x_a^i := \sum_{(v_a^i, v_b^i) \in A^i} y_{a,b}^i \in \{0, 1\}, \quad (2)$$

which equals 1 iff v_a^i is selected (the target $v_{n_i}^i$ is always selected). This eliminates the variables x_a^i and their linking constraints from the model.



■ **Figure 4** Level structure and duration constraints for train i . Level ℓ_3^* (shaded) is optional.

Path selection and level duration constraints. The flow constraints select one source-to-target path per train:

$$\sum_{v_b^i \in S_0^i} y_{0,b}^i = 1 \quad i \in I, \quad (3)$$

$$\sum_{v_a^i \in P_{n_i}^i} y_{a,n_i}^i = 1 \quad i \in I, \quad (4)$$

$$\sum_{v_a^i \in P_c^i} y_{a,c}^i = \sum_{v_b^i \in S_c^i} y_{c,b}^i \quad v_c^i \in V^i \setminus \{v_0^i, v_{n_i}^i\}, \quad i \in I. \quad (5)$$

For the duration constraint, we apply ideas of [5] on maximal and non-maximal clusters, which correspond to our definition of levels and optional levels. For each consecutive level pair $(l, l+1)$ of train i , the level start times respect the durations of the selected operations, allow waiting at waiting arcs, and are tight on no-wait arcs. Writing x_a^i as in (2):

$$t_{l+1}^i - t_l^i \geq \sum_{v_a^i \in V_l^i} \delta_a^i x_a^i \quad l \in \{0, \dots, L^i - 1\}, \quad i \in I, \quad (6)$$

$$t_{l+1}^i - t_l^i \leq \sum_{v_a^i \in V_l^{i, \text{nw}}} \delta_a^i x_a^i + M \left(1 - \sum_{v_a^i \in V_l^{i, \text{nw}}} x_a^i \right) \quad l \in \{0, \dots, L^i - 1\}, \quad i \in I, \quad (7)$$

$$t_{l+1}^i - t_l^i \leq M \sum_{v_a^i \in V_l^i} x_a^i \quad l \text{ optional}, \quad i \in I. \quad (8)$$

Constraint (7) activates the no-wait equality $t_{l+1}^i = t_l^i + \delta_a^i$ exactly for the selected no-wait operation, and (8) collapses an unused optional level onto its predecessor ($t_{l+1}^i \leq t_l^i$), keeping the no-wait chain transitive across skipped levels. A level start time t_l^i carries only the loosest lower bound among the operations sharing level l . When a selected operation v_a^i has a tighter bound $\alpha_a^i > \alpha_l^i$, we enforce it conditionally with the exact gap $M_a^i = \alpha_a^i - \alpha_l^i$, so the constraint is inactive unless v_a^i is selected:

$$t_{\ell(a)}^i \geq \alpha_a^i - M_a^i (1 - x_a^i) \quad v_a^i \in V^i : \alpha_a^i > \alpha_{\ell(a)}^i. \quad (9)$$

For the instances under study no symmetric constraint is needed for the upper bound, since upper bounds are only used for the source node v_0^i , so that $\beta_a^i = \beta_{\ell(a)}^i$. In general, the analogous constraint can be added for operations with $\beta_a^i < \beta_{\ell(a)}^i$.

Resource timing constraints. Consider a usage $\rho \in \mathcal{R}$ that appears in a conflict, and let $\mathcal{A}_\rho^{\text{res}} \subset A^{i(\rho)}$ be a $v_0^i - v_\rho^{\text{in}}$ -cut, the *indicator arcs* whose selection leads to the reservation of ρ . The resource indicator variable is then

$$x_\rho = \sum_{(v_a^i, v_b^i) \in \mathcal{A}_\rho^{\text{res}}} y_{a,b}^i \quad \rho \in \mathcal{R}, \quad (10)$$

for each $\rho \in \mathcal{R}$ whose $\mathcal{A}_\rho^{\text{res}}$ is not a $v_0^i - v_{n_i}^i$ -cut; otherwise we set $x_\rho \equiv 1$. The reservation and release times of a usage ρ are relative to the start times of the operations that reserve it first and release it last, and both the reserving operations $\mathcal{R}_\rho^{\text{in}}$ and the releasing operations $\mathcal{R}_\rho^{\text{out}}$ may lie on different levels of G^i . Let $L_\rho^{\text{rv}} = \{\ell(v_a^i) : v_a^i \in \mathcal{R}_\rho^{\text{in}}\}$ and $L_\rho^{\text{rl}} = \{\ell(v_a^i) : v_a^i \in \mathcal{R}_\rho^{\text{out}}\}$ be the levels with reserving and releasing operations. We anchor the timing per level: for every releasing level $l \in L_\rho^{\text{rl}}$ the release is bounded below by the start of level l plus the release offset of the releasing operation selected at that level, and symmetrically for every

reserving level $l \in L_\rho^{\text{rv}}$ the reservation is bounded above. A big- M term deactivates each constraint whenever no operation of ρ is selected at its level:

$$t_\rho^{\text{out}} \geq t_l^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{out}} \cap V_l^i} \delta_a^{\text{rl},i} x_a^i - M \left(1 - \sum_{v_a^i \in \mathcal{R}_\rho^{\text{out}} \cap V_l^i} x_a^i \right) \quad l \in L_\rho^{\text{rl}}, \rho \in \mathcal{R}, \quad (11)$$

$$t_\rho^{\text{in}} \leq t_l^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{in}} \cap V_l^i} \delta_a^{\text{rv},i} x_a^i + M \left(1 - \sum_{v_a^i \in \mathcal{R}_\rho^{\text{in}} \cap V_l^i} x_a^i \right) \quad l \in L_\rho^{\text{rv}}, \rho \in \mathcal{R}, \quad (12)$$

where $i = i(\rho)$. If $|L_\rho^{\text{rv}}| = 1$, then no big- M term is needed in (12). In Section 5.3 we show how to replace these by-level constraints with a single constraint for most usages ρ .

Resource conflict constraints. For each conflict $\{\rho, \sigma\} \in \mathcal{K}$, the orientation variables $z_{\rho, \sigma}$ and $z_{\sigma, \rho}$ select an order only if both usages are realized, as proposed before in [9]:

$$x_\rho \geq z_{\rho, \sigma} + z_{\sigma, \rho}, \quad x_\sigma \geq z_{\rho, \sigma} + z_{\sigma, \rho}, \quad (13)$$

$$x_\rho + x_\sigma - 1 \leq z_{\rho, \sigma} + z_{\sigma, \rho} \leq 1, \quad (14)$$

where a mandatory usage substitutes $x. \equiv 1$. In particular, if both usages are mandatory, then (14) forces $z_{\rho, \sigma} + z_{\sigma, \rho} = 1$. The chosen order separates the conflicting reservations, enforced by big- M constraints that are non-binding when the orientation is inactive:

$$t_\sigma^{\text{in}} - t_\rho^{\text{out}} \geq -M(1 - z_{\rho, \sigma}) \quad \{\rho, \sigma\} \in \mathcal{K}, \quad (15)$$

$$t_\rho^{\text{in}} - t_\sigma^{\text{out}} \geq -M(1 - z_{\sigma, \rho}) \quad \{\rho, \sigma\} \in \mathcal{K}. \quad (16)$$

Objective. Each objective component lower-bounds its delay variable by the arrival time beyond the threshold:

$$w_c \geq t_{\ell_c}^{i(c)} + \sum_{v_a^{i(c)} \in \mathcal{V}_c} \delta_a^{i(c)} x_a^{i(c)} - \theta_c \quad c \in \mathcal{D}. \quad (17)$$

The objective minimizes the fixed operation penalties and the weighted delays:

$$\min \sum_{i \in I} \sum_{v_a^i \in V^i} \lambda_a^i x_a^i + \sum_{c \in \mathcal{D}} \gamma_c w_c. \quad (18)$$

5.2 Algorithm

■ **Algorithm 1** Conflict-driven decomposition for TDP-NW.

```

1: Initialize the disjunctive graph  $D$  from the train DAGs  $\{G^i\}_{i \in I}$ 
2:  $\mathcal{C} \leftarrow \emptyset$  ▷ active conflict set
3:  $\mathcal{S} \leftarrow \{\{i\} : i \in I\}$  ▷ one subproblem per train
4: while true do
5:   for  $S \in \mathcal{S}$  do ▷ in parallel
6:      $\text{SOLVEMIP}(S \mid \mathcal{C}, D)$ 
7:   end for
8:    $\mathcal{C}_{\text{new}} \leftarrow \text{CONFLICTDETECTION}(D)$  ▷  $D$  optimal w.r.t.  $\mathcal{C}$ 
9:   if  $\mathcal{C}_{\text{new}} = \emptyset$  then
10:    return current solution
11:   end if
12:    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\text{new}}$ 
13:   merge the subproblems in  $\mathcal{S}$  that are newly linked by  $\mathcal{C}_{\text{new}}$ 
14: end while

```

The orientation and separation constraints (13)–(16) couple two trains through a resource conflict $\{\rho, \sigma\} \in \mathcal{K}$, and in practice most conflicts are not relevant for solving the optimization problem. We solve the model iteratively, and in a decomposed fashion: the trains are grouped into independent subproblems, each subproblem is the compact model of Section 5.1 restricted to its own trains, and the constraints of a conflict are only added once that conflict is actually observed in an intermediate infeasible solution.

Conflict Detection. We implemented an oracle that, given the selected and timed operations for each train, returns for each resource $r \in U$ all pairs (ρ, σ) of conflicting usages, using algorithmic ideas of maximal cliques in interval graphs as done before in [8]. Due to potential sub-cliques introduced to cover overlap exclusions, see Section 3, a conflict between the same two trains may appear for two equivalent resources from the perspective of the two trains. For the conflict discovery and the given usage intervals, we test for domination among conflicts between the same two trains. A conflict $C' = \{\rho_i, \rho_j\}$ is dominated by another conflict $C'' = \{\sigma_i, \sigma_j\}$ of the same train pair if each train occupies its resource of C' within the time it occupies its resource of C'' , that is $[t_{\rho_i}^{\text{in}}, t_{\rho_i}^{\text{out}}] \subseteq [t_{\sigma_i}^{\text{in}}, t_{\sigma_i}^{\text{out}}]$ and $[t_{\rho_j}^{\text{in}}, t_{\rho_j}^{\text{out}}] \subseteq [t_{\sigma_j}^{\text{in}}, t_{\sigma_j}^{\text{out}}]$. The first processed conflict wins in tie-breaking situations. We collect the conflicts for each train pair by resource $r \in U$ in ascending order, so that resources from sub-cliques are processed last and we only create a conflict for a sub-clique resource if one of the participating trains uses that resource as an overlap.

Starting from one subproblem per train and an empty conflict set, Algorithm 1 repeatedly solves all subproblems and inspects the combined solution for conflicts $\{\rho, \sigma\} \in \mathcal{K}$ that are violated and not yet represented. Each such conflict is added to $\mathcal{C}$ and the subproblems of the respective trains are potentially merged. The algorithm terminates when no new conflict is found, at which point the combined solution is feasible and optimal for the full model.

5.3 Strengthened Model

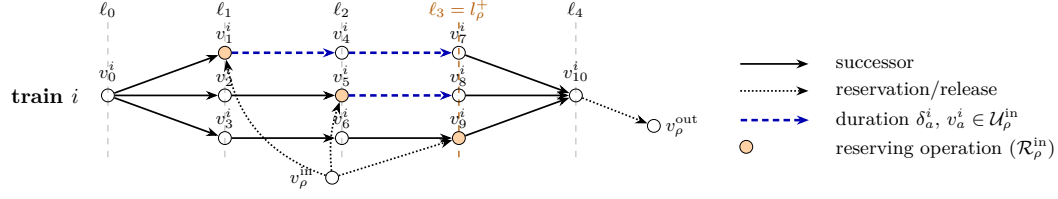
In this section, we discuss several performance improvements. Furthermore, for the DISPLIB instances we require some so-called no-swap constraints. Finally, we detail the implemented heuristic configurations to support faster convergence of the iterative algorithm.

Level-aggregated resource timing constraints. Let l_ρ^- and l_ρ^+ be the levels of the earliest releasing and latest reserving operation of usage $\rho \in \mathcal{R}$, which we call the *anchor levels* of ρ . The release time is bounded below by the start of its earliest releasing level l_ρ^- plus the durations of the selected operations traversed up to the actual release, plus the release time of the selected releasing operation. The reservation time is bounded above by the start of its latest reserving level l_ρ^+ plus the reservation times of the selected reserving operations, minus the durations of the intermediate operations selected before l_ρ^+ . Here $\mathcal{U}_\rho^{\text{in}} \subseteq V^i$ are the intermediate operations on the path strictly before l_ρ^+ , and $\mathcal{U}_\rho^{\text{out}} \subseteq V^i$ are the intermediate operations from l_ρ^- until (and excluding) the actual release. They lie on a path between the anchor level of ρ and its actual reservation or release, respectively. Figure 5 illustrates the constraint for a train with three alternative paths reserving ρ on different levels.

$$t_\rho^{\text{out}} \geq t_{l_\rho^-}^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{out}}} \delta_a^{\text{rl}, i} x_a^i + \sum_{v_a^i \in \mathcal{U}_\rho^{\text{out}}} \delta_a^i x_a^i \quad \rho \in \mathcal{R}, \quad (19)$$

$$t_\rho^{\text{in}} \leq t_{l_\rho^+}^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{in}}} \delta_a^{\text{rv}, i} x_a^i - \sum_{v_a^i \in \mathcal{U}_\rho^{\text{in}}} \delta_a^i x_a^i \quad \rho \in \mathcal{R}, \quad (20)$$

where $i = i(\rho)$.



■ **Figure 5** Level-aggregated resource constraint for a resource usage ρ with three alternative paths that reserve ρ at different levels. The latest reserving level is $\ell_\rho^+ = \ell_3$, with the associated start-time variable $t_{\ell_3}^i$. The dashed arcs are the durations δ_a^i of the intermediate operations $\mathcal{U}_\rho^{\text{in}} = \{v_1^i, v_4^i, v_5^i\}$, which conditionally push the reservation time back from the anchor level to the actual reservation.

If the chosen path reserves ρ at an operation before its latest reserving level ℓ_ρ^+ and some operations between that reservation and ℓ_ρ^+ permit waiting, then we can satisfy (20) by waiting just before ℓ_ρ^+ , even though the reservation has already occurred. The symmetric situation arises for the release in (19). On both sides the constraint is therefore a relaxation of the true timing: t_ρ^{in} may be set later, and t_ρ^{out} earlier, than the selected operations actually allow. Whenever the solver returns a solution that violates the reservation or release timing, we repair it by adding a constraint anchored at each reserving or releasing level, with (11) and (12). We recognize such a violation when processing a solution, where we propagate the time-constraint bounds in our disjunctive graph and compare them against the solution time variable of each reservation or release node. Ideally, most reservations and releases of a given resource usage ρ are anchored at a common level, keeping the potential number of such repair constraints small.

Implication constraints between conflicts. We generalize the implication rules of [4] to our generalized disjunctive graph. Similar ideas about linking conflicts have been shown by [5]. To this end we define the concept of *overlapping* resource usage of a train between ρ and σ . Let train i visit ρ before σ ; then ρ and σ are overlapping iff for every path between $\mathcal{R}_\rho^{\text{in}}$ and $\mathcal{R}_\sigma^{\text{in}}$, σ is reserved before ρ is released. Furthermore, we say ρ forward implies σ iff ρ and σ are overlapping and every path starting at $\mathcal{R}_\rho^{\text{in}}$ unavoidably visits a node of $\mathcal{R}_\sigma^{\text{in}}$. Conversely, we say σ backward implies ρ iff every path ending at $\mathcal{R}_\sigma^{\text{in}}$ unavoidably visits a node of $\mathcal{R}_\rho^{\text{in}}$. If both directions are true, we say ρ and σ are strictly overlapping.

Now let $C' = \{\rho_i, \rho_j\}$ and $C'' = \{\sigma_i, \sigma_j\}$ be a pair of conflicts between the same two trains (i and j). If both trains have strictly overlapping resource uses, then the decisions for C' and C'' have to be the same. Thus, we can reuse the z -variables for the second conflict, which corresponds to grouping the disjunctive constraints and applying them jointly.

If not both trains have strictly overlapping resources for the conflict pair, but train i forward- or backward-implies from ρ_i to σ_i and likewise train j from ρ_j to σ_j , we can no longer share the z -variables. Still, no overtaking is possible once both trains use both resources of the conflict pair. Therefore we add order agreement cuts, which require that the two orientation decisions coincide whenever both conflicts are active:

$$z_{\rho_i, \rho_j} + z_{\sigma_j, \sigma_i} \leq 1, \quad z_{\rho_j, \rho_i} + z_{\sigma_i, \sigma_j} \leq 1. \quad (21)$$

The first inequality forbids train i from preceding train j at C' while train j precedes train i at C'' , and the second forbids the symmetric mismatch. Together they force the two conflicts to select the same train order on these two resources. For an inactive conflict both its orientation variables are forced to 0 (13), so the cuts stay valid even when a usage is optional and the corresponding resource indicator is not selected.

No-swap constraints. If we want to model operations and resource usages of zero duration, or to model the problem assuming a train length of zero, no-swap constraints become relevant. Swapping appears if all current resources of a train are released at the start of an operation exactly when new resources are reserved; if swapping were allowed, then the train would not hold any resource at this precise moment of the start of the operation, and another train that wants to use these resources in reverse order could match the swap point. The train dispatching problem is usually considered with no-swap constraints, whereas for classic job-shop scheduling, swapping may be reasonable.

A feasible solution is a topological ordering without backward arcs considering the state of the disjunctive graph, which is only possible if there are no cycles. The zero-cost cycles are exactly those that show swapping possibilities. Therefore, for every new conflict, we collect all zero-cost cycles and add corresponding cuts on the disjunctive variables. The number of zero-cost cycles can be quite high. We observed up to 2800 zero-cost cycles in `nor5` instances. Alternatively, we could consider adding only the cycles that incrementally appear in the disjunctive graph for any infeasible solution observed while iterating. Note that we also need to add a special case for the previously explained domination among newly discovered conflicts: a resource usage with an observed zero duration interval cannot be dominated since it may be a swapping pivot.

In Section 6, we will show the performance of our approach on a TDP-NW dataset and the DISPLIB instances. To find optimal and feasible solutions for the DISPLIB instances, exactly these no-swap constraints are needed, as swapping scenarios are common there. In the TDP-NW instances, swapping is impossible since the resource usage of operations is modeled considering the train from head to tail, and no trains of zero length are considered.

Heuristic configurations. We implemented several parameters to configure more heuristic approaches. First, we work with a timeout per iteration. Second, after a given time threshold, we begin to fix the decision variables of conflicts. We only fix confident decisions, i.e., decisions that have remained unchanged for at least the last k iterations, where a longer streak means a more confident decision. Among these candidates, we fix a configurable percentage per iteration. In the experiments in Section 6, we always used a minimum streak of $k = 4$ and a percentage of 5%.

6 Computational Results

First, we present computational results on a challenging curated TDP-NW dataset, created from DB InfraGO instances [18] for three different dispatching regions, and time horizons of 20, 40, 60, and 90 minutes. Second, we show that our algorithm is also competitive for the DISPLIB instances [9], where we find new best known solutions within a time limit of 10 minutes, using the commercial MIP solver Gurobi 13.0.1 [7] and a computer with 10 cores.

Benchmark dataset for TDP-NW. Together with this paper, we publish a dataset of challenging problem instances from dispatching data of a DB InfraGO traffic management system [17]. We selected instances for three dispatching regions of Frankfurt, Munich and Stuttgart, covering morning and afternoon traffic. We select instances with many conflicts and a large connected component in the conflict graph. The threshold values for the objective components are raised to the shortest-path lower bound on the arrival time whenever that lower bound exceeds the threshold, to remove the unavoidable delay from the objective. As in DISPLIB, we keep the instance format simple and general. This first TDP-NW benchmark

is created from graphs of velocity profiles for each train with route reservations and releases given relative to a profile, see [18]. The route exclusions are transformed into exclusive resources, as described in Section 3, and the velocity profiles into operations that reserve and release these resources. In Appendix A, Table 4 reports our results on the curated TDP-NW dataset.

DISPLIB instances. The DISPLIB problem does not explicitly model the kind of microscopic feasibility we address with our work on TDP-NW. Distinguishing features are the overlap resource usages and the no-wait property of operations. For the latter, however, one could subsume paths of no-wait operations into a single operation. The DISPLIB instances come from various sources [9] considering dispatching horizons between 6 and 24 hours. The number of trains per instance ranges from 4 to 505, with a maximum of $\approx 50\,000$ operations for the **nor5** instances. Our approach can prove optimality within 170 seconds for all DISPLIB instances, except **wab** and most **nor5** instances. Table 2 reports the average runtime per instance group for the groups we solve to proven optimality. That includes three new optimality proofs for the **smi_headway** instances compared to [5], and in considerably less running time. For detailed results see Table 5 in Appendix A. In Table 6 we show seven new best known solutions given a time limit of 10 minutes for this benchmark. The solutions themselves are published in the repository with the TDP-NW Benchmark [17]. For the **wab_large** and **wab_small** instances our approach is unable to close the gaps within 10 minutes and we need a heuristic setting to find good solutions. Therefore we run our solver with a timeout of 20 seconds per iteration and start fixing 5% of unfixed conflicts after 50 seconds. Table 7 shows that we find 20 new best known solutions out of 32 instances.

■ **Table 2** DISPLIB instance groups solved to proven optimality, with the average runtime over all instances of each group. The average of **swi** instances is mostly driven by **swi_9**.

Instance group	Count	Avg. time [s]
nor1_critical	10	0.12
nor1_full	10	1.31
nor2	5	1.00
nor3	5	1.62
nor4_small	5	17.22
nor4_large	5	46.18
smi_close	9	0.22
smi_headway	12	0.65
swi	9	22.71

Comparison of TDP-NW and DISPLIB instances. The TDP-NW instances differ structurally from the DISPLIB instances, as Table 3 illustrates. The train graphs observed in the DISPLIB instances reach higher depths but have few operations per level, whereas the TDP-NW instances are shallower but have many more operations per level. This is expected, since the TDP-NW instances cover much shorter dispatching horizons. Moreover, they contain many alternative operations for the same physical track section, differing in start and end velocities and the used route. The ratio of optional levels in the TDP-NW instances is similar to the one in the **wab** instances, which results from rerouting alternatives of different length.

■ **Table 3** Structural comparison of selected instances: average number of levels per train, average number of operations per level, and the ratio of optional levels.

Instance	Avg. levels	Avg. level size	Ratio opt. levels
fbzm_large_dh_40	18.19	42.94	0.24
mbzs_dh_20	14.42	34.36	0.19
rbzs_dh_20	15.23	13.14	0.17
wab_large	83.61	1.51	0.20
nor5_large	50.42	1.99	0.00

7 Conclusion and Outlook

The related work on microscopic train scheduling often claims physical feasibility but seldom refers to an explicit enumeration of velocity profiles with precomputed reservation and release times. Moreover, the details of infrastructure exclusions are usually approximated. In this work we have presented a real-life train dispatching dataset which models the physical feasibility to this level. We have established a method to compute an equivalent resource-based representation from route-based infrastructure constraints, including partial route exclusions and overlap route exclusions. This bridges the gap between real-life feasibility and the extensive related work on timetabling and train dispatching using disjunctive graphs. Naturally, no-wait constraints arise in such instances. Moreover, the number of alternative operations becomes very large. While our newly contributed TDP-NW benchmark dataset only covers dispatching horizons of up to 90 minutes and at most 148 trains, its instances are already large in the number of operations. To address that, we developed modeling techniques on disjunctive graphs with dedicated reserving and releasing nodes to express resource conflicts across alternative operations. Combined with big- M -free reservation and release constraints anchored at a single level-time variable for the vast majority of resources, this enables new performance gains. The techniques developed, in particular how to group conflicts by resource usage pair and how conflicts then relate to each other, are MIP-independent contributions that may equally benefit heuristics and custom branch-and-bound methods for train dispatching with rerouting. Table 4 shows that the relative optimality gaps are large for most instances. The absolute gaps, however, stay below 3600 delay seconds for all but four instances, distributed over up to 60 trains of the largest subproblem. This provides some degree of confidence in the quality of the solutions.

While our results are the best known overall for the DISPLIB instances and, for the TDP-NW dataset, meet the standards of our industrial partner DB InfraGO, the high number of operations per level on these instances and the fact that most optimality gaps are far from closed demonstrate the need for different or complementary procedures. Column generation comes to mind, as most operations might be dominated by others given a set of considered conflicts. This has been explored by [16] for the same type of instances, but modeled directly on routes. The transformation and simplification of the instances might enable gains for column generation for TDP-NW. For the same reason data reduction rules could be considered as a preprocessing step, or in an incremental framework depending on the growing set of conflicts. The work of [5] demonstrates data reductions for the DISPLIB instances, contracting sequences of operations. Another stream would be to implement custom branch-and-bound frameworks, where no-wait constraints and fixed operation costs make the longest path subproblem potentially harder. We hope that our TDP-NW dataset will stimulate research for practically efficient and scalable algorithms for railway dispatching.

References

- 1 Maurizio Boccia, Veronica Dal Sasso, Leonardo Lamorgese, Carlo Mannino, and Paolo Ventura. Optimizing train dispatching for the Union Pacific Railroad, March 2025. doi:10.48550/arXiv.2503.19535.
- 2 Valentina Cacchiani, Dennis Huisman, Martin Kidd, Leo Kroon, Paolo Toth, Lucas Veelenturf, and Joris Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, May 2014. doi:10.1016/j.trb.2014.01.009.
- 3 Veronica Dal Sasso, Leonardo Lamorgese, Carlo Mannino, Andrea Onofri, and Paolo Ventura. A 0,1 Linear Programming Approach to Deadlock Detection and Management in Railways. *Transportation Science*, 59(1):187–205, January 2025. doi:10.1287/trsc.2024.0521.
- 4 Andrea D’Ariano, Dario Pacciarelli, and Marco Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, December 2007. doi:10.1016/j.ejor.2006.10.034.
- 5 Florian Fuchs, Thomas Dubach, Jan Lordieck, Francesco Corman, and Bernardo Martin-Iradi. Handling conflicts in microscopic railway timetabling at scale: A parallelized approach. *Journal of Rail Transport Planning & Management*, 39:100600, September 2026. doi:10.1016/j.jrtpm.2026.100600.
- 6 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics*, 13:2:2.2–2:2.15, February 2009. doi:10.1145/1412228.1412236.
- 7 Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2026.
- 8 Oddvar Kloster, Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. An Optimization-Based Decision Support Tool for Incremental Train Timetabling. *Operations Research Forum*, 4(3):65, August 2023. doi:10.1007/s43069-023-00243-2.
- 9 Oddvar Kloster, Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. DISPLIB: A library of train dispatching problems, July 2026. doi:10.2139/ssrn.7069663.
- 10 Leonardo Lamorgese and Carlo Mannino. A Noncompact Formulation for Job-Shop Scheduling Problems in Traffic Management. *Operations Research*, 67(6):1586–1609, November 2019. doi:10.1287/opre.2018.1837.
- 11 Florin Leutwiler, Guillem Bonet Filella, and Francesco Corman. Accelerating logic-based Benders decomposition for railway rescheduling by exploiting similarities in delays. *Computers & Operations Research*, 150:106075, February 2023. doi:10.1016/j.cor.2022.106075.
- 12 Florin Leutwiler and Francesco Corman. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research*, 303(2):525–540, December 2022. doi:10.1016/j.ejor.2022.02.043.
- 13 Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. Exploiting Incrementality in Branch and Bound for Train Dispatching, May 2025. doi:10.2139/ssrn.5260286.
- 14 Alessandro Mascis and Dario Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, December 2002. doi:10.1016/S0377-2217(01)00338-1.
- 15 Jörn Pachl. *Railway Signalling Principles: Edition 2.0*. Universitätsbibliothek Braunschweig, October 2021. doi:10.24355/dbbs.084-202110181429-0.
- 16 Maik Schällicke and Karl Nachtigall. Solving the Real-Time Train Dispatching Problem by Column Generation. *Transportation Science*, 59(3):587–602, May 2025. doi:10.1287/trsc.2023.0215.
- 17 Luka Stärk. TDP-NW: Benchmark Instances for the Train Dispatching Problem with No-Wait Constraints, August 2026. doi:10.5281/zenodo.21775303.
- 18 Stephan Zieger and Hannah Richta. DB InfraGO’s Automated Dispatching Assistant ADA-PMB, May 2025. doi:10.48550/arXiv.2505.10085.

A

 Appendix

■ **Table 4** Our results on the curated TDP-NW dataset, using 10 seconds per iteration and, after 50 seconds, fixing at most 5% of the most confident decisions. Columns: conflicts $|C|$, trains $|I|$, operations $|V|$, largest subset max $|S|$, iterations, objective (UB), lower bound (LB), and time.

Instance	$ C $	$ I $	$ V $	max $ S $	Iter.	UB	LB	Time [s]
fbzm_dh_20_0	117	40	19314	16	5	2172.21	1836.21	58.41
fbzm_dh_20_1	127	48	16003	18	13	2311.17	2311.17	39.69
fbzm_dh_20_2	418	44	18396	29	14	3245.90	2344.65	141.27
fbzm_dh_40_0	80	48	24717	22	10	1177.18	1177.18	34.28
fbzm_dh_40_1	135	49	36616	20	7	1572.11	1327.39	71.27
fbzm_dh_60_0	397	83	52231	45	12	5000.44	3792.61	124.75
fbzm_dh_60_1	365	85	46917	44	14	2602.12	1802.90	152.18
fbzm_dh_60_2	180	77	40889	32	7	1739.01	1572.72	70.32
fbzm_large_dh_20_0	271	56	18710	29	10	2329.97	1863.78	100.56
fbzm_large_dh_20_1	335	56	21945	27	9	3764.98	2486.41	92.54
fbzm_large_dh_20_2	357	49	27794	31	20	2672.22	2450.80	195.20
fbzm_large_dh_20_3	201	48	32261	9	5	2152.21	2152.21	15.62
fbzm_large_dh_20_4	158	39	11057	10	6	1233.10	1170.69	50.17
fbzm_large_dh_40_0	910	86	63760	46	16	3448.25	1877.44	176.60
fbzm_large_dh_40_1	1668	86	70543	55	20	23981.02	2661.29	245.34
fbzm_large_dh_60_0	1695	113	80503	62	33	13442.58	3394.11	369.67
fbzm_large_dh_60_1	1092	107	67923	55	25	6732.99	4482.21	266.61
fbzm_large_dh_90_0	2212	134	118681	83	23	8577.28	4396.40	394.20
mbzs_dh_20_0	135	47	15769	19	5	1346.30	1346.30	4.51
mbzs_dh_20_1	255	41	25314	30	10	3123.87	2960.89	94.69
mbzs_dh_20_2	111	38	10463	8	5	1228.02	1228.02	7.50
mbzs_dh_20_3	274	45	39916	22	10	2334.40	2177.63	103.17
mbzs_dh_20_4	317	46	23593	34	11	2665.67	1773.07	136.36
mbzs_large_dh_40_0	400	63	5778	18	10	5211.96	5209.46	37.90
mbzs_large_dh_60_0	141	46	4718	15	5	2662.33	2662.33	4.59
rbzs_dh_20_0	246	64	12567	33	5	2056.27	2056.27	16.86
rbzs_dh_20_1	378	58	10940	30	7	2724.21	1912.85	74.05
rbzs_dh_20_2	359	60	15629	46	11	3060.58	2419.52	119.58
rbzs_dh_20_3	466	64	12498	40	11	5012.52	2912.90	129.59
rbzs_dh_20_4	348	66	10817	44	9	5340.66	4056.77	87.17
rbzs_dh_60_0	338	120	50457	60	7	3133.21	2818.75	71.42
rbzs_dh_90_0	1208	148	70779	107	17	9290.57	5337.03	197.84

■ **Table 5** Detailed per-instance performance on the DISPLIB instances solved to proven optimality. Columns as in Table 4, with the linking conflicts $|C_{\text{link}}|$ and the order agreement cuts $|C_{\text{agree}}|$.

Instance	$ \mathcal{C} $	$ \mathcal{C}_{\text{link}} $	$ \mathcal{C}_{\text{agree}} $	$ I $	$\max S $	Iter.	UB	LB	Time [s]
nor1_critical_0	50	8	27	12	10	9	4133	4133	0.09
nor1_critical_1	45	8	17	8	8	8	2416	2416	0.05
nor1_critical_2	35	10	18	9	7	9	3775	3775	0.05
nor1_critical_3	132	19	70	16	16	15	8016	8016	0.48
nor1_critical_4	9	0	1	4	3	4	1506	1506	0.01
nor1_critical_5	13	4	1	6	5	6	2677	2677	0.02
nor1_critical_6	63	12	32	12	10	14	4491	4491	0.12
nor1_critical_7	41	5	19	10	9	10	4137	4137	0.06
nor1_critical_8	76	16	41	10	9	16	3836	3836	0.17
nor1_critical_9	78	13	41	12	12	13	5488	5488	0.12
nor1_full_0	447	7	67	119	88	17	3299	3299	0.88
nor1_full_1	520	3	62	148	111	22	2486	2486	1.27
nor1_full_2	207	5	84	40	34	16	6046	6046	0.47

continued on next page

■ Table 5 (continued from previous page)

Instance	$ C $	$ C_{\text{link}} $	$ C_{\text{agree}} $	$ I $	$\max S $	Iter.	UB	LB	Time [s]
nor1_full_3	336	21	93	56	47	32	2658	2658	2.55
nor1_full_4	407	4	86	89	73	22	5358	5358	1.16
nor1_full_5	490	3	86	129	103	18	3732	3732	1.06
nor1_full_6	515	4	87	147	106	21	5262	5262	1.05
nor1_full_7	499	0	88	146	108	19	1012	1012	0.99
nor1_full_8	674	21	155	155	121	28	5370	5370	1.88
nor1_full_9	587	8	86	151	121	26	6076	6076	1.79
nor2_1	193	29	105	23	23	15	4937	4937	0.95
nor2_2	230	34	125	23	22	15	4619	4619	1.10
nor2_3	172	25	93	23	19	11	5500	5500	0.85
nor2_4	202	32	102	23	20	17	6186	6186	1.40
nor2_5	170	24	89	23	20	11	5416	5416	0.69
nor3_1	317	45	178	21	20	28	3667	3667	3.21
nor3_2	237	33	139	22	21	22	5740	5740	2.21
nor3_3	193	18	100	22	21	13	5562	5562	0.79
nor3_4	187	25	102	21	21	11	4605	4605	0.51
nor3_5	205	27	119	22	19	21	2923	2923	1.37
nor4_small_1	1587	9	227	165	146	161	8972	8972	23.42
nor4_small_2	1426	23	247	168	146	107	11217	11217	18.21
nor4_small_3	1213	18	178	161	134	107	7924	7924	13.78
nor4_small_4	1149	7	174	157	126	97	8189	8189	12.07
nor4_small_5	1366	18	199	167	148	128	12144	12144	18.64
nor4_large_1	2895	26	255	365	296	148	9551	9551	43.64
nor4_large_2	3085	27	277	347	288	226	8439	8439	61.37
nor4_large_3	1277	24	229	171	138	95	11985	11985	33.53
nor4_large_4	2433	38	347	257	219	140	13260	13260	71.51
nor4_large_5	1669	20	219	204	174	134	8043	8043	20.83
smi_close_0	7	2	2	6	3	1	679	679	0.01
smi_close_1	65	20	24	6	4	4	4316	4316	0.37
smi_close_2	57	12	29	5	5	5	1331	1331	0.81
smi_close_3	7	0	0	13	4	1	1860	1860	0.04
smi_close_4	4	1	0	5	3	2	24225	24225	0.01
smi_close_5	9	1	1	9	7	3	314	314	0.14
smi_close_6	44	8	35	7	7	4	21034	21034	0.15
smi_close_7	14	0	9	6	6	6	582	582	0.30
smi_close_8	40	4	19	9	4	6	3413	3413	0.18
smi_headway_0	14	3	4	6	3	1	1483	1483	0.01
smi_headway_1	66	22	19	6	6	2	5040	5040	0.39
smi_headway_2	124	28	87	5	5	5	2888	2888	1.59
smi_headway_3	24	2	9	13	7	3	3552	3552	0.14
smi_headway_4	4	1	0	5	3	2	24797	24797	0.01
smi_headway_5	24	0	7	9	7	6	869	869	0.96
smi_headway_6	50	11	38	7	7	5	22236	22236	0.19
smi_headway_7	27	3	14	6	6	8	1150	1150	0.61
smi_headway_8	68	11	37	9	6	10	5159	5159	0.56
smi_headway_9	82	20	34	6	6	5	16006	16006	0.52
smi_headway_10	91	15	45	8	7	8	9007	9007	2.10
smi_headway_11	94	19	37	8	7	5	5579	5579	0.73
swi_1	0	0	0	4	1	0	0	0	0.01
swi_2	207	175	11	58	12	8	0	0	0.06
swi_3	839	697	103	143	56	17	0	0	0.81
swi_4	1081	832	200	148	74	23	0	0	1.72
swi_5	1085	835	200	149	74	23	0	0	1.75
swi_6	2270	1809	403	365	86	8	0	0	1.13
swi_7	5108	4161	1018	467	360	36	0	0	23.68
swi_8	2103	1512	872	133	119	32	0	0	5.00
swi_9	14563	8063	11577	287	285	46	0	0	170.20

■ **Table 6** Our results on the **nor5** instances and **nor4_large_4**, obtained with a timeout of five minutes, after which we limit each solve to 20 seconds and start fixing disjunctive variables. Objective values in bold are new best known solutions. Improvement is the gain (0 if we match) over the previous best published on the DISPLIB page and reported in [5] for a time limit of 10 minutes.

Instance	UB	LB	Previous best	Improvement	Time [s]
nor4_large_4	13260	13260	13261	1	72.0
nor5_small_1	57443	57404	57554	111	422.7
nor5_small_2	66751	66348	66751	0	404.5
nor5_small_3	57446	57446	57446	0	333.5
nor5_small_4	57361	57361	58484	1123	339.7
nor5_small_5	62744	62699	62744	0	359.9
nor5_large_1	72713	72248	72799	86	433.5
nor5_large_2	76410	75189	77118	708	510.9
nor5_large_3	69910	69331	70026	116	414.8
nor5_large_4	69519	69051	69519	0	552.3
nor5_large_5	67277	66874	67312	35	431.9

■ **Table 7** Our results on the **wab_large** and **wab_small** instances, using a heuristic configuration: after 50 seconds we iteratively fix at most 5% of the most confident decisions, with 20 seconds per iteration. Objective values in bold are new best known solutions. Improvement is the gain over the previous best published on the DISPLIB page (negative if we do not match it). We report no lower bounds, as they are not competitive in this setting: once we start fixing decisions they no longer apply to the original instance.

Instance	UB	Previous best	Improvement	Time [s]
wab_large_1	141607	151279	9672	480.6
wab_large_2	145791	144017	-1774	475.8
wab_large_3	121948	152642	30694	492.9
wab_large_4	138726	133630	-5096	508.2
wab_large_5	129493	143886	14393	481.7
wab_large_6	127624	134923	7299	516.1
wab_large_7	125161	146501	21340	484.3
wab_large_8	100352	117051	16699	483.9
wab_large_9	113191	123528	10337	477.7
wab_large_10	130166	127116	-3050	481.3
wab_large_11	140696	127862	-12834	507.9
wab_large_12	133456	130225	-3231	495.3
wab_large_13	91603	117875	26272	473.8
wab_large_14	122546	123753	1207	257.3
wab_large_15	124887	129949	5062	501.3
wab_large_16	107823	128082	20259	489.7
wab_small_1	18003	17055	-948	256.4
wab_small_2	16156	16478	322	254.3
wab_small_3	19179	18767	-412	294.1
wab_small_4	21103	20165	-938	301.1
wab_small_5	25298	25255	-43	374.1
wab_small_6	22540	23318	778	296.8
wab_small_7	21274	26486	5212	293.5
wab_small_8	27450	20138	-7312	373.5
wab_small_9	20601	24116	3515	343.6
wab_small_10	26768	28750	1982	357.3
wab_small_11	23676	24189	513	324.1
wab_small_12	24214	22571	-1643	329.4
wab_small_13	19324	19064	-260	315.7
wab_small_14	19626	19938	312	344.4
wab_small_15	20035	21392	1357	286.2
wab_small_16	15742	19015	3273	265.8

Adaptive Metaheuristics for Multi-Objective Berth Allocation and Scheduling

Konstantinos Karathanasis ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
PIKEI New Technologies, Patras, Greece

Spyros Kontogiannis ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
Computer Technology Institute and Press “Diophantus”, Patras, Greece

Asterios Pegos ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
PIKEI New Technologies, Patras, Greece

Vasileios Sofianos ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece

Christos Zaroliagis ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
Computer Technology Institute and Press “Diophantus”, Patras, Greece

Abstract

The *Berth Allocation and Scheduling Problem* (BASP) is a challenging combinatorial optimisation problem arising in container terminal operations, where decisions must balance conflicting goals including operational efficiency, economic cost, environmental impact, and infrastructure utilisation. This paper investigates the multi-objective discrete dynamic BASP with time windows and introduces a four-objective mixed-integer linear programming formulation that simultaneously captures vessel service efficiency, CO₂ emissions, operating cost, and berth workload balance.

To solve this computationally challenging problem, we develop and systematically evaluate adaptive multi-objective metaheuristic approaches. First, we adapt four evolutionary multi-objective algorithms through problem-specific variation operators and an adaptive operator selection mechanism. Second, we extend *Adaptive Large Neighbourhood Search* (ALNS) to the multi-objective BASP setting and propose two variants: SMOALNS, based on an external Pareto archive, and NSALNS, which integrates ALNS with NSGAII environmental selection.

Extensive computational experiments demonstrate that adaptive operator selection consistently improves evolutionary baselines and that the proposed multi-objective ALNS approaches achieve the best overall performance across benchmark instances. The results indicate that adaptive destroy–repair search mechanisms are particularly effective for highly constrained berth scheduling problems, providing high-quality approximations of the Pareto front for realistic port management scenarios.

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains; Applied computing → Multi-criterion optimization and decision-making

Keywords and phrases Berth Allocation and Scheduling, Multi-objective Optimization, Metaheuristics, Evolutionary Computation

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.7

Funding This work is partially supported by the EU I3 Instrument under GA No 101115116 (project AMBITIOUS) and by the University of Patras under GA No 83770 (programme “MEDICUS”).

1 Introduction

The growth of global maritime trade has placed increasing pressure on marine container terminals, making the efficient utilisation of berthing resources a critical operational challenge. Inefficient berth planning can cause vessel congestion, long waiting times, and increased



© Konstantinos Karathanasis, Spyros Kontogiannis, Asterios Pegos, Vasileios Sofianos, and Christos Zaroliagis;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 7; pp. 7:1–7:22



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

operational and environmental costs. This challenge, commonly modelled as the *Berth Allocation and Scheduling Problem* (BASP), involves assigning arriving vessels to berthing positions and service times while satisfying spatial and temporal constraints. BASP variants may differ w.r.t. the quay layout: (i) *discrete* if the quay is divided into fixed berths, each serving at most one vessel at any time; (ii) *continuous*, if vessels can be assigned to any feasible position along the quay; and (iii) *hybrid*, if a vessel may occupy multiple adjacent berths, while each berth may simultaneously serve multiple vessels. They may also differ w.r.t. vessel arrival information: (i) *static*, if all vessels are already present awaiting to be served; (ii) *dynamic*, if vessels arrive over time at a priori known arrival times; and (iii) *uncertain*, if vessel arrival times are subject to variability. We denote these variants as XYBASP, where $X \in \{D, C, H\}$ represents the quay layout and $Y \in \{S, D, U\}$ the arrival-time setting.

Many practical BASP formulations also include *time-window constraints*, requiring vessels to be serviced within predefined intervals dictated by contractual, tidal, or operational considerations, which significantly increases problem complexity. Moreover, real-world berth planning involves multiple conflicting objectives, such as minimising vessel turnaround times, operational costs and environmental impact, while ensuring balanced berth utilisation.

Motivated by these challenges, we study the *multi-objective Discrete Dynamic Berth Allocation and Scheduling Problem with time windows* (moDDBASPTw). Our model simultaneously minimises total vessel service time in the system, CO₂ emissions and terminal operating costs, while promoting workload balance across berths. Given the intractability of the problem, we propose and experimentally evaluate several metaheuristic approaches that efficiently generate diverse high-quality approximations of the Pareto front, supporting informed decision-making in realistic port management environments.

Related Work. BASP has received considerable attention due to its impact on terminal efficiency and operational costs. The single-objective BASP is NP-hard [15], rendering exact approaches computationally prohibitive for large-scale instances and motivating the use of heuristic and metaheuristic methods. Several studies have focused on single-objective BASP variants. In [13], (single objective) DDBASPTw is formulated as a MILP inspired by vehicle routing models, and is approximately solved using constructive and improvement heuristics, as well as metaheuristics including ALNS, genetic algorithms, and cuckoo search.

The inherent limitations of single-objective formulations have led to increasing interest in multi-objective BASP, which not only inherits NP-hardness [2] but introduces additional complexity due to conflicting objectives. In [10], daytime preference is incorporated as an additional objective, and a bi-objective genetic algorithm is proposed to balance service efficiency and daytime workload. A Pareto-based evolutionary framework is introduced in [4] that jointly minimises makespan, waiting time, and deviation from priority schedules, incorporating problem-specific local search mechanisms. A multi-objective vessel scheduling framework accounting for cost efficiency, service reliability, and environmental impact is studied in [21]. Environmental aspects are explicitly addressed in [22] where shore-side electricity and emission reduction are integrated into berth allocation models, and in [17] where service efficiency and carbon emissions are jointly optimised using the enhanced multi-objective artificial bee colony (EMOABC) algorithm.

Despite these advances, none of the existing works has considered all the involved optimisation directions: (i) operational efficiency (e.g., expressed by total vessel waiting and service times), (ii) aggregate service cost, (iii) environmental footprint (e.g., expressed by CO₂ emissions) and (iv) balanced exploitation of the port infrastructure (e.g., as indicated by the workload distribution). Moreover, systematic comparisons of established multi-objective

metaheuristics within a common BASP setting are scarce, and algorithms with proven effectiveness in single-objective BASP, such as ALNS, have not yet been extensively explored in the multi-objective context. Recent studies have also extended the ALNS paradigm to multi-objective optimisation. Hybrid MOALNS frameworks, combining ALNS with population-based approaches such as NSGAII [6], SPEA2 [24], and MOEA/D [23], have been proposed for applications including transportation planning [18], green machine scheduling [5], disassembly line balancing [8], and vehicle routing [14]. These methods exploit adaptive destroy–repair operators, demonstrating the potential of MOALNS to effectively approximate Pareto fronts in complex combinatorial optimisation problems. Nevertheless, experimental evaluations are often restricted to small instances and a limited set of performance metrics.

Our Contribution. This paper makes the following contributions to moDDBASPTw.

1. We introduce, to the best of our knowledge, the first formulation that simultaneously considers objective functions in all the previously mentioned optimisation directions, namely, operational efficiency, total cost, environmental footprint, and balanced exploitation of the infrastructure.
2. We investigate the application of general-purpose multi-objective evolutionary algorithms (MOEAs) to moDDBASPTw by adapting four well-established approaches, namely NSGAII [6], NSGAIII [7], SPEA2 [24], and MOEA/D [23], through problem-specific variation operators. In particular, we design and implement five crossover operators and two mutation operators tailored to the structure of berth allocation schedules, and perform a systematic experimental analysis of their impact on MOEA performance. To the best of our knowledge, this constitutes the first systematic evaluation of crossover operators in the context of multi-objective BASP. We also implement the EMOABC [17] approach verbatim and include it in our experimental evaluation.
3. We adjust a well-established metaheuristic from the VRP literature, Adaptive Large Neighbourhood Search (ALNS) [19], to the moDDBASPTw setting. We propose two variants of ALNS: a simple multi-objective variant (SMOALNS), and a more sophisticated approach (NSALNS) that integrates ALNS within the selection and diversity-preservation mechanisms (non-dominated sorting) of NSGAII. Both these variants outperform all other approaches for solving moDDBASPTw, consistently achieving superior performance across nearly all evaluation metrics, thus demonstrating the effectiveness of combining large neighbourhood search with evolutionary multi-objective techniques.
4. We extend the adaptive operator selection mechanism originally introduced in ALNS to the considered MOEA frameworks. The resulting adaptive evolutionary variants dynamically select among multiple crossover and mutation operators during the search process. Extensive experimental evaluation demonstrates that adaptive operator selection improves upon deterministic operator configurations.

2 Problem Description & Mathematical Formulation

In this section, we introduce a *multi-objective* mathematical formulation of DDBASPTw, hereafter referred to as **moDDBASPTw**. The proposed formulation extends the *Mixed Integer Linear Programming* (MILP) formulation introduced in [13] for the (single-objective) DDBASPTw, inspired by its connection with the *Heterogeneous VRP*.

In particular, it considers four objective functions (see Figure 2 in the Appendix): (i) the minimisation of the *total time* each vessel spends within the port, reflecting operational efficiency; (ii) the minimisation of *CO₂ emissions* generated during a vessel’s stay in the

port, addressing environmental sustainability; (iii) the minimisation of the *overall berthing cost*, representing economic performance; and (iv) the minimisation of *load variation across berthing positions*, promoting a balanced and sustainable utilisation of port infrastructure. To discourage poor-quality schedules that simply deny service to vessels, the formulation explicitly accounts for penalties associated with vessels that are not assigned to any berth within their feasible time windows, thereby incentivising solutions that maximize service coverage while balancing the competing objectives. Additionally, our formulation explicitly defines appropriate constraints for the CO₂ emissions associated with each vessel-to-berth assignment. All other constraints remain the same as in [13]. Rather than seeking a single optimal solution, the objective is to identify a diverse set of approximate solutions that efficiently cover as much as possible the Pareto front, while satisfying all problem constraints.

Each vessel $i \in S$ is characterized by an estimated time of arrival T_i^{ea} and a requested time of departure T_i^{rd} (perceived as a soft deadline). The terminal enforces a delay policy T^{dp} , which specifies the maximum allowable delay beyond a vessel's requested departure time. This policy induces a hard time window $[T_i^{ea}, T_i^{rd} + T^{dp}]$, within which vessel i must be assigned to a berth and complete its service; otherwise, the vessel is denied service. Each vessel $i \in S$ carries a cargo quantity Q_i and has a desirable number of cranes that could ideally be allocated for its service. Furthermore, vessel i is associated with four penalty parameters P_i^t , P_i^e , P_i^c , and P_i^l , which represent the loss incurred by the terminal operator per objective function, in the event that the vessel is denied service.

Regarding the port infrastructure, the wharf is assumed to be partitioned into a set B of heterogeneous berths. Each berth $j \in B$ is characterized by its length L^j , draft D^j , per-hour operational cost C^j , and number N^j of operational homogeneous cranes (e.g., requiring the same amount of time K to load / unload one unit of cargo). Similarly, each vessel $i \in S$ is characterized by its length L_i and draft D_i , which must be compatible with the corresponding attributes of the berth to which it is assigned. Assigning vessel i to berth $j \in B$ results in different waiting times, handling times, departure delays and CO₂ emissions. The *waiting time* of i is defined as the elapsed time between its arrival and the commencement of its mooring at berth j . The *handling time* of i depends on the specific vessel–berth pairing and reflects the duration required by berth j to service vessel i , based on its cargo quantity and the operational characteristics of the berth. A *departure delay* for i is defined as the extra time (if any) from T_i^{rd} until the completion of its service at berth j . The *CO₂ emissions* of i depend on the specific vessel–berth pairing.

To formalize the problem, the following assumptions are adopted: (i) all the spatio-temporal details of the vessels arriving within the planning horizon are known in advance (discrete-dynamic scenario); (ii) each vessel requires a single, uninterrupted time interval for handling (no pre-emption of currently handled vessels); and (iii) all berths are empty at the beginning of the planning horizon. The proposed MILP formulation employs decision binary variables $x_{i,k}^j$, defined for all triples $(i, k, j) \in S^2 \times B$, which indicate whether vessel k is serviced immediately after vessel i in the servicing sequence of berth j . It further includes continuous temporal variables defined for pairs $(i, j) \in S \times B$, representing the mooring time t_i^j of vessel i at berth j (equal to T_i^{ea} if vessel i is not assigned to berth j), the handling time h_i of vessel i , the CO₂ emissions e_i of vessel i and the departure delay t_i^{ld} , defined as the time by which the completion of service exceeds the requested departure time T_i^{rd} . For more details about the formulation see Section A in the Appendix.

3 Solution Approaches

In multi-objective optimisation problems, the aim is to determine a maximal set of solutions that reflect different trade-offs among the competing objectives, so that every feasible solution is not superior across all objectives to (i.e., does not dominate) any solution within this set. Such a set is commonly referred to as *Pareto set*, or *Pareto front*. In this work, we approximately solve moDDBASPTw by employing custom-tailored variants of well-known multi-objective evolutionary algorithms (MOEAs) and two novel multi-objective variants of ALNS to construct large solution sets, which are not necessarily subsets of the Pareto front, but they nevertheless cover a large range of trade-offs among the competing objectives and none of the solutions is dominated any other solution within this set.

3.1 Baseline MOEA Approaches

The majority of MOEAs share a common algorithmic framework. They maintain a population of feasible solutions, initialized either randomly or heuristically. This population evolves over successive generations until a predefined termination criterion is fulfilled. In each generation, *mate selection* is used to choose individuals for reproduction, followed by a *variation (crossover/mutation) operator* to generate an offspring. Each offspring is then evaluated by computing all objective values, and *environmental selection* determines which individuals advance to the next generation. MOEAs are largely problem-independent, with each algorithm defining its own mate and environmental selection mechanisms. To adapt a MOEA to a specific problem, one needs only to specify how the population is initialized and how variation operators are applied to generate new solutions. In this study, we use four well-known MOEAs as our baseline algorithms: NSGAII [6], NSGAIII [7], SPEA2 [24], and MOEA/D [23]. All these baseline algorithms use the same population initialization procedure (cf. Section 3.2) and apply identical variation operators (cf. Section 3.3). They differ in the *mate selection* and *environmental selection* mechanisms, as well as in whether they return the final population or maintain an external archive of non-dominated solutions.

3.2 Population Initialization

The initial population is always generated using the constructive heuristic *aggregate pre-determined order Insertion* (agg-po-INS), which is a simple variant of the *predetermined order insertion* (po-INS) heuristic proposed in [13]. This heuristic incrementally constructs berth schedules by inserting one unassigned vessel at a time into partially built schedules. Based on a predetermined vessel ordering, at each step, all feasible insertion positions for the current vessel within each eligible berth are evaluated, and the position minimising the marginal cost increase is chosen, while preserving the relative order of previously assigned vessels. Mooring times of already scheduled vessels may be adjusted as long as temporal and operational constraints are satisfied. In our multi-objective setting, the marginal cost increase is computed as the increase in a weighted aggregation of the four objectives. For this operation to be meaningful, all objective values are normalized; this is consistently applied in this work whenever a weighted aggregation of the objectives is used.

To promote diversity of the initial population and to avoid bias towards any single objective, five predefined weight vectors are used: one assigning equal importance to all objectives (a weight of 0.25 each), and the other four emphasizing a particular objective (assigning to it a weight of 0.4) and equally splitting the remaining weight to the others (0.2 each). For a population of size N , approximately $N/5$ solutions are generated per weight vector. Diversity within each weight vector is achieved by varying the vessel insertion order, producing structurally distinct schedules and ensuring broad coverage of the objective space.

3.3 Variation Operators

Crossover Operators. A crossover operator generates an offspring solution by combining structural information from two parent solutions. In this work, we designed and evaluated five crossover operators, which are described below.

RandomParent (RP) processes vessels independently. For each vessel $i \in S$, if it was assigned to some berth by only one parent, the offspring inherits that assignment (berth and mooring time). If it was assigned to berths by both parents, one assignment is selected with equal probability. If the inherited assignment is infeasible w.r.t. assignments already copied into the offspring (e.g., temporal overlap on the same berth), it is discarded and vessel i is marked as unassigned. After all vessels are processed, unassigned vessels are reinserted using an **agg-po-INS**-based repair mechanism. The repair follows the **agg-po-INS** procedure described in Section 3.2, using equal objective weights (0.25 each), and operates on partially constructed berth schedules formed by the inherited assignments.

RandomBerth (RB) acts at berth level, rather than at vessel level. A berth $j \in B$ is selected uniformly at random, and its entire schedule in the first parent is replaced by its schedule in the second parent. As a consequence, vessels that were originally assigned to berth j in the first parent but are not present in the second parent's schedule of j become unassigned. Moreover, vessels copied from the second parent may have been assigned to other berths in the first parent; in such cases, their original assignments are removed to preserve uniqueness. All vessels that are left unassigned, are reinserted using the same **agg-po-INS**-based repair mechanism employed by the **randomParent** operator.

HalfBerths (HB) partitions the set of berths B in two disjoint subsets of equal size. The offspring inherits the schedules of berths in the first subset from the first parent and those of the remaining berths from the second parent, ignoring assignments that would duplicate vessels already inherited. Any unassigned vessels resulting from this process are reinserted using the **agg-po-INS**-based repair mechanism.

HalfTime (HT) performs recombination along the time dimension. Assignments from both parents are combined and ordered by the midpoint of their service intervals, $t_i^j + h_i/2$, and the temporal split $\bar{t}$ is defined as the median midpoint. The offspring inherits assignments from the first parent whose midpoints occur before $\bar{t}$ and from the second parent whose midpoints occur after $\bar{t}$, provided the vessel has not already been assigned. Assignments causing temporal conflicts, as well as vessels whose midpoints lie on the non-selected side of $\bar{t}$ in both parents, are marked as unassigned. All unassigned vessels are subsequently reinserted using the **agg-po-INS**-based repair mechanism.

Quarter (Q) combines **HalfBerths** and **HalfTime** by partitioning both the time horizon and the berth set. The temporal split $\bar{t}$ is defined as in **HalfTime**, while B is partitioned into two equal subsets B_1 and B_2 . For assignments with midpoints before $\bar{t}$, vessels assigned to berths in B_1 are inherited from the first parent and those in B_2 from the second parent; for assignments after $\bar{t}$, the inheritance is reversed. Duplicate assignments are ignored, and assignments causing temporal conflicts or omitted due to the combined partitioning are marked as unassigned. All unassigned vessels are finally reinserted using the **agg-po-INS**-based repair mechanism, yielding a feasible offspring.

Mutation Operators. The role of a mutation operator is to introduce controlled perturbations to existing solutions in order to enhance exploration while preserving feasibility. In this work, we design two mutation operators which are based on two well-known improvement heuristics from VRP literature, namely *ReInsert* and *Swap* [16]. In each case, before executing a mutation operator, the **agg-po-INS** repair mechanism is first applied to reinsert vessels that are not serviced in the current solution.

ReInsert is applied for a predefined number of iterations. In each iteration, a vessel is selected uniformly at random and removed from its assigned berth, allowing subsequently scheduled vessels to advance their mooring times if permitted by temporal constraints. The removed vessel is then reinserted using the **agg-po-INS** repair mechanism. The resulting solution is evaluated, and the changes are accepted only if the number of unserved vessels does not increase; otherwise, the solution is reverted. In the computational experiments, the number of iterations is set to 0.4 times the number of vessels.

Swap explores the neighbourhood through pairwise exchanges of vessel assignments between berths. A predefined number of vessel pairs is selected uniformly at random. For each pair, both vessels are first removed from their own berths and consequently they are reinserted into each other's original berths; if either vessel is incompatible with the new berth, the pair is skipped. All feasible insertion positions are evaluated, and the one minimising the marginal cost increase is selected according to the **agg-po-INS** principles, using equal objective weights. While reinsertion may shift the mooring times of other vessels, their relative order is preserved. Each swap is accepted only if it does not increase the number of unserved vessels; otherwise, the solution is reverted. In the computational experiments, the number of evaluated vessel pairs is set to 0.6 times the number of vessels.

3.4 Multi-Objective Adaptive Large Neighbourhood Search (MOALNS)

Adaptive Large Neighbourhood Search (**ALNS**) is a metaheuristic for single-objective optimisation problems that was introduced in [19] as an extension of Large Neighbourhood Search (**LNS**). It iteratively evolves the current solution by applying to it a pair of *destroy* and *repair* operators (exactly as **LNS**) but the selection process is now adaptive, choosing among multiple operators with probabilities proportional to their past performances. A custom-tailored variant of **ALNS** for **DDBASPtW** was proposed in [13], where the experimental evaluation demonstrated its clear superiority over baseline (single-objective) evolutionary algorithms. Motivated by this observation, we propose two novel adaptations for **moDDBASPtW**: the Simple Multi-Objective **ALNS** (**SMOALNS**) and the Non-dominated Sorting **ALNS** (**NSALNS**).

First, we introduce a generic **MOALNS** framework that defines the common structure shared by all proposed variants. At the beginning, an initial population of solutions is generated, as described in Section 3.2. Consequently, we initialise the weights of all the destroy and repair operators to equal values, implying uniformity in selection probabilities. Any additional data structures required by a specific variant are also initialized at this point. **MOALNS** proceeds iteratively until a predefined termination criterion is reached. In each generation, for each solution x in the current population $\mathcal{P}$, a destroy operator and a repair operator are selected according to their adaptive selection probabilities and applied consecutively, resulting in a new candidate solution x' . An acceptance function $\text{accept}(x', x, T)$ is then invoked, which is identical for all variants: If x' dominates x , then $\text{accept}(x', x, T)$ returns 1. If x dominates x' , it returns -1 . Otherwise (i.e., when neither x nor x' dominates the other), a simulated annealing-based criterion is applied, to allow the probabilistic acceptance of the newly generated solution even though it does not dominate the current solution, thereby promoting exploration. In particular, given the temperature parameter T , the value $e^{-diff/T}$ is computed, where $diff$ denotes the absolute difference between the values of x and x' w.r.t. the weighted sum of all the objectives. A random number r is drawn then uniformly from $[0, 1]$; if $r < e^{-diff/T}$, then $\text{accept}(x', x, T)$ returns 0 (to indicate the fact that a non-dominating solution was promoted this time), otherwise it returns -1 . After the acceptance decision is made, each variant applies its own procedure to handle the result. The scores of the selected operators are then updated based on the outcome of $\text{accept}(x', x, T)$.

These scores are subsequently used to update the operator weights. If $\text{accept}(x', x, T)$ returns 1, the selected operators receive the highest reward $rw_1 \in (0, 1)$; if it returns 0, they receive a smaller reward $rw_2 \in (0, rw_1)$; otherwise, no reward is granted to them. Once all solutions in the population have been processed, each variant performs its respective post generation operations.

A sequence of *updatePeriod* consecutive generations constitutes an *epoch*. At the end of the k -th epoch, the aggregate scores and weights (indicating new selection probabilities) for all the destroy and repair operators within the subsequent epoch $k + 1$ are calculated as follows: For each operator i ,

$$w_i^{k+1} := (1 - \rho) w_i^k + \rho \cdot \begin{cases} \frac{s_i^k}{u_i^k}, & \text{if } u_i^k > 0, \\ 0, & \text{otherwise,} \end{cases}, \quad p_i^{k+1} := \frac{w_i^{k+1}}{\sum_{j: \text{type}(j) = \text{type}(i)} w_j^{k+1}}, \quad (1)$$

where the type of each operator is from $\{D(\text{estroy}), R(\text{epair})\}$, w_i^k is the weight of operator i , s_i^k is its aggregate score, and u_i^k is the number of times it was selected, within epoch k . The new selection probability of i for epoch $k + 1$ is p_i^{k+1} , whereas $\rho \in (0, 1)$ determines the learning rate. This procedure allows MOALNS to adaptively favor operators that have been more successful in improving solutions, over time. At the same time, the temperature parameter is reduced from one epoch to the next, according to a geometric cooling schedule, i.e., $T^{k+1} := a \cdot T^k$, where $a \in (0, 1)$ is the cooling parameter.

Simple multi-objective ALNS (SMOALNS) and Non-Dominated Sorting ALNS (NSALNS).

SMOALNS maintains an external archive A that stores all mutually non-dominated solutions identified during the entire search. Starting with a population initialization step (cf. Section 3.2), for each new generation, if the acceptance function returns a value greater than or equal to zero (i.e., the current solution x is replaced by the new solution x'), then A is updated only if x' is not dominated by any solution already in A , and (in that case) all existing solutions in A which are dominated by x' are removed. SMOALNS does not apply any additional post generation population processing. Upon termination, the algorithm returns the external archive A obtained during the search, as an approximation of the Pareto front.

NSALNS incorporates the environmental selection principles of NSGAII. Instead of maintaining an external archive, NSALNS relies exclusively on population-based selection. During each generation, an offspring population $\mathcal{O}$ is constructed by appending each newly generated solution x' to $\mathcal{O}$, regardless of the outcome of $\text{accept}(x, x', T)$. After all solutions in $\mathcal{P}$ have been processed, non-dominated sorting (as in NSGAII) is performed on the combined population $\mathcal{P} \cup \mathcal{O}$, followed by crowding distance computation. The lexicographically best $|\mathcal{P}|$ solutions from $\mathcal{P} \cup \mathcal{O}$, w.r.t (smaller) dominance ranks and (larger) crowding distances, are selected to form the population for the next generation. Upon termination, NSALNS returns the final population $\mathcal{P}$ as the approximation of the Pareto front. Among the examined environmental selection mechanisms, the NSGAII strategy was retained due to its superior hypervolume and spread performance over all other baseline approaches in our experimental evaluation (cf. Section 4.1).

MOALNS Operators. Both variants employ the same set of destroy and repair operators.

Destroy Five destroy operators were implemented to selectively dismantle parts of the current solution: (i) random removals of assigned vessels, (ii) clearance of randomly selected berths, (iii) removal of vessels that significantly contribute to the total objective aggregated cost, (iv) removal of vessels that are spatially similar to a selected seed vessel

(i.e., vessels with similar length and draft) and (v) removal of vessels that are temporally similar to a selected seed vessel (i.e., vessels with close estimated time of arrival). The degree of destruction (*deg*) controls the proportion of the solution to be removed. A sufficiently high value of *deg* increases the likelihood of escaping from local optima.

Repair Six repair operators were implemented to reconstruct solutions following the destruction phase. Three of these operators are based on the **agg-po-INS** heuristic and differ only in the order in which the removed vessels are processed. Specifically, the vessel processing order is determined by (ascending) estimated-arrival times, or by (ascending) requested-departure times, or by (ascending) compatible-berths numbers. For all these repair operators, equal objective weights (0.25 per objective) are used in the aggregate cost computation. The fourth operator also relies on the **agg-po-INS** heuristic, processing vessels according to the order in which they were removed during the destruction phase, using again equal objective weights in the computation of the aggregate cost. The fifth operator is almost identical to the fourth, but employs randomly generated (instead of equal) objective weights. Finally, the sixth operator is based on a *k*-regret insertion strategy, which iteratively selects and inserts the vessel with the largest regret value from the set of removed and currently unassigned vessels. The *regret value* of a vessel is defined as the difference between the aggregate cost for its best feasible insertion position and the aggregate cost of its *k*-th best feasible insertion, using equal objective weights.

3.5 NSCSA with Decomposition and Adaptive Operator Selection

We developed also NSACSA/D, an enhanced version of NSCSA, a recently proposed metaheuristic [12] that combines the selection mechanisms of NSGAI with the exploration capabilities of the *Cuckoo Search Algorithm* (CSA) [9]. NSACSA/D adopts a decomposition-based framework and incorporates the adaptive operator selection mechanism of ALNS. The experimental results (cf. Section 4.1) revealed only a marginal performance improvement over the baseline methods. Therefore, we do not further analyze its behavior in detail.

3.6 Adaptive Operator Selection in Evolutionary and Genetic Algorithms

The performance of evolutionary and genetic algorithms is highly sensitive to the choice of crossover and mutation operators, and no single operator is expected to be universally optimal. In our case, the *a priori* confinement of the baseline algorithms to fixed combinations of variation operators may bias the search. On the other hand, it could be beneficial to equip each baseline MOEA with an adaptive operator selection mechanism to autonomously identify and exploit the most effective operators, while preserving exploration through probabilistic selection, similarly to the adaptive operator selection mechanism of ALNS.

To this end, adaptive variants of these algorithms are also introduced, denoted by adding an “A” to each original algorithm’s name: NSAGAI, NSAGAI, SPAEA2, and MOAEA/D. In particular, the adaptive mechanism for the destroy and repair operators in MOALNS (cf. Section 3.4) is adopted also in MOAEAs for selecting crossover and mutation operators. Each operator is associated with a weight, initialized uniformly and updated online based on observed performance. During reproduction, operators are chosen according to their adaptive selection probabilities (which are proportional to operator weights). After offspring evaluation, rewards are assigned independently for each parent: rw_1 if the offspring dominates the parent, rw_2 if they are mutually nondominated, and 0 if the parent dominates the offspring. For example, an offspring dominating both parents receives reward $rw_1 + rw_1$, while one dominating one parent and neither dominating nor being dominated by the other parent receives reward $rw_1 + rw_2$. Operator weights and the corresponding selection probabilities are updated periodically via exponential smoothing, similarly to MOALNS (cf. Equations (1)).

4 Experimental Evaluation

All experiments were conducted on a workstation running Ubuntu 22.04, equipped with an AMD Ryzen Threadripper 3960X processor (24 physical cores, 48 threads, base frequency of 3.80 GHz, and 128 MB L3 cache) and 256 GB of RAM. All algorithms were implemented in C++ and compiled using GCC with the `-O3` optimisation flag. The algorithm parameters were determined through a preliminary parameter tuning procedure, conducted on a representative subset of the instances used in the complete experimental evaluation, and were subsequently fixed for all experiments. For algorithms incorporating adaptive operator selection, the tuning process considered four parameters: the initial temperature T , cooling rate a , update period, and weight update parameter ρ . The candidate values investigated were: $T \in \{1, 2, 3, 4, 5, 6, 7\}$, $a \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $updatePeriod \in \{10, 15, 20\}$, and $\rho \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Parameter combinations were evaluated, and the configurations providing the best performance during the tuning phase were selected. The final parameter settings were: $T^{\text{MOALNS}} = T^{\text{MOEA}} = 3$, $a^{\text{MOALNS}} = a^{\text{MOEA}} = 0.3$, $\rho^{\text{MOALNS}} = 0.6$, $\rho^{\text{MOEA}} = 0.8$, and $updatePeriod = 10$ iterations for all methods. The operator rewards were set to $rw_1 = 1$ and $rw_2 = 0.4$. Finally, all MOALNS variants used a fixed destruction degree of $deg = 0.4$ in all experiments.

Performance Indicators. In our MILP formulation for 4-objective DDBASPTw, computing the (exact) Pareto front is computationally inefficient for all but toy instances. Even constructing a reliable approximation of the Pareto front through exact methods would be prohibitively expensive for meaningful problem sizes. Nevertheless, several performance indicators require a reference set that somehow approximates the Pareto front. Therefore, we adopt the following approach for the construction of a meaningful reference set, which is common practice in the literature [3, 20, 12, 17]: We first construct the union of the solution sets produced by all the algorithms under comparison across the experimental runs. From this set, only the non-dominated solutions are retained. The resulting set, called the *reference Pareto front*, is used as a surrogate for the unknown Pareto front when evaluating performance metrics that depend on it.

The experimental evaluation of each algorithm is done using the following set of commonly adopted performance indicators that capture computational efficiency, convergence, and diversity w.r.t. the reference Pareto front. **(i) Runtime (T):** It measures the total wall-clock time required by an algorithm to complete an optimisation run. It reflects computational efficiency and is particularly relevant for large-scale or time-sensitive applications. **(ii) Hypervolume (HV):** It is a unary performance indicator that simultaneously captures convergence and diversity. Given a solution set $S \subset \mathbb{R}^d$ and a single reference point $r \in \mathbb{R}^d$, the hypervolume of S is defined as the value of the Lebesgue measure $\Lambda(S)$ of the region that is weakly dominated by S and bounded above by r : $H(S) = \Lambda(\{q \in \mathbb{R}^d \mid \exists p \in S : p \leq q \leq r\})$. **(iii) Spread (Δ):** It evaluates the diversity and uniformity of the distribution of solutions returned by each algorithm. It measures how evenly the solutions cover the objective space and whether extreme regions of the front are adequately represented. A *lower* spread value indicates a more uniformly distributed solution set, whereas higher values suggest clustering or gaps along the front. This indicator is particularly useful when assessing an algorithm's ability to maintain solution diversity. **(iv) Inverted Generational Distance (IGD):** It assesses the quality of an obtained solution set S by measuring its average distance to a given reference Pareto front P^* . IGD computes the mean of distances of points in P^* from their nearest neighbour in S . Lower IGD values indicate better convergence and coverage of

the reference Pareto front, as they imply that the obtained solutions closely approximate the reference front across the entire objective space. In this work, the performance indicators are considered in lexicographic order of importance: $HV \prec \Delta \prec IGD \prec T$. We mainly focus on the quality of the obtained solution sets rather than execution time, as all algorithms have comparable running times (up to 15 minutes and differing by at most $3.6\times$).

4.1 Experimental Results

For the experimental evaluation, the time unit is set to 1 hour; for example, if $h_i = 2$, vessel i requires 2 hours to be handled by its assigned berth. The time for a single crane to load or discharge a unit of cargo is $K = 5$ minutes (1/12 hours), and the delay policy is $T^{dp} = 10$ hours. Vessel arrival times are considered within a single week (i.e., $\forall i \in S, T_i^{ea} \in [0, 168]$). Since no benchmark sets are available for the specific problem addressed in this study, all instances were generated using the data generator described in [13]. This generator exploits publicly available real-world data on vessel arrivals and berthing activity from Piraeus Container Terminal [1] (which can service up to eleven vessels simultaneously) for the creation of synthetic-data instances which reflect varying operational scales and at the same time are statistically comparable to real-world instances. Each benchmark data set is denoted as X-Y (X vessels, Y berths) and contains ten randomly generated instances of the given size; all tables report the arithmetic means across these random instances per data set. Columns include: “Algorithm” (method executed), “T” (average runtime in seconds), “HV” (hypervolume), “ Δ ” (spread), and “IGD” (inverted generational distance). The performance metrics are computed as follows. For methods returning populations, only non-dominated solutions are retained. The *reference Pareto front* is then constructed from the solution sets of all methods, and all points returned by the algorithms are normalized using the nadir and zenith points of this reference Pareto front. Metrics are subsequently calculated on the normalized solutions. For the HV metric, again following common practice, we consider the single reference point $r = (1.1, 1.1, 1.1, 1.1)$.

[EXP-I] Crossover Operator Comparison. Evolutionary algorithms are often sensitive to the choice of crossover. With multiple operators available in our framework, it is not immediately clear which provides a fair baseline configuration. To investigate this, each baseline algorithm was executed with all five crossover operators while keeping other parameters fixed. The **ReInsert** operator was used as the mutation operator in all cases, as preliminary experiments consistently indicated slightly superior performance over **Swap**. All MOEAs were run with a fixed population size of 100 solutions, and for 200 generations. For each crossover and instance, results were averaged across all baselines (i.e., the mean performance of a given crossover over all algorithms was reported). HV and Δ were used as performance metrics.

Table 1 reports the resulting average values for each problem instance. The results indicate that **RandomParent** (RP) consistently provides strong performance across instances, achieving the largest HV in all but two instances and the smallest Δ in all cases. Based on this observation, **RandomParent** is selected as the fixed crossover operator for all non-adaptive baseline algorithms in subsequent experiments. This choice ensures a fair and well-justified comparison with the adaptive variants, which are free to dynamically select among all available crossover operators.

[EXP-II] Baseline vs. Adaptive Variants. The comparison between baseline MOEAs and their adaptive counterparts are reported in Table 5 of Section B. Overall, the adaptive variants demonstrate slightly better solution quality and substantially faster execution times across most instances, reflecting the benefits of the **ALNS**-inspired adaptive operator selection.

■ **Table 1** Crossover averages (HV and Δ) across problem instances.

C/o	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$
	40 - 10	45 - 10	60 - 20	70 - 20	80 - 20	90 - 20
RP	1.022 0.733	1.119 0.701	1.058 0.781	0.996 0.713	1.103 0.771	1.12 0.684
RB	1.005 0.761	1.073 0.768	1.046 0.823	0.977 0.802	1.098 0.831	1.097 0.792
HB	0.981 0.831	1.035 0.812	1.026 0.849	0.97 0.799	1.084 0.833	1.073 0.795
HT	0.997 0.836	1.058 0.815	1.056 0.875	0.977 0.8	1.092 0.817	1.067 0.836
Q	0.995 0.801	1.069 0.806	1.049 0.851	0.967 0.785	1.094 0.82	1.068 0.783
C/o	100 - 25	120 - 30	135 - 30	140 - 35	150 - 35	
RP	1.055 0.733	1.094 0.725	1.058 0.696	1.091 0.722	1.087 0.748	
RB	1.05 0.831	1.09 0.806	1.068 0.784	1.065 0.829	1.072 0.871	
HB	1.023 0.806	1.072 0.812	1.034 0.754	1.063 0.788	1.101 0.845	
HT	1.035 0.798	1.082 0.784	1.049 0.751	1.077 0.798	1.099 0.835	
Q	1.032 0.792	1.073 0.759	1.036 0.743	1.062 0.775	1.057 0.797	

[EXP-III] Comparison with MOALNS Variants, NSACSA/D, and EMOABC. This experiment evaluates the performance of the proposed adaptive multi-objective evolutionary algorithms against NSACSA/D and an existing approach, namely EMOABC, as well as two MOALNS variants, namely SMOALNS and NSALNS. The EMOABC algorithm was selected as a baseline due to its recent application to multi-objective berth allocation problems [17]. It extends the conventional Artificial Bee Colony framework through several enhancement mechanisms, including a cooperative crossover operator, multiple neighbourhood search strategies, and archive-based local refinement. The implementation of EMOABC follows the procedure described in [17] as closely as possible to ensure a fair comparison. The original parameter settings were adopted, except for the parameter M , which was set to 10 to account for the higher number of objectives considered in this study. EMOABC, MOAEA/D, NSACSA/D, and SMOALNS return an external archive, whereas the remaining algorithms return a fixed-size set of solutions. We preserve this behavior to avoid altering the original design of the algorithms. All algorithms were evaluated using a population size of 100 individuals under two complementary experimental protocols. First, in the *Same Generations Experiment*, each algorithm was executed for 200 generations, enabling a direct comparison of solution quality under an identical number of evolutionary iterations. Second, in the *Time Limit Experiment*, all algorithms were compared under identical computational budgets to assess their performance w.r.t. computational efficiency.

[EXP-III.a] Same Generations Experiment. Table 2 summarizes the results obtained when all algorithms are executed for the same number of generations. Overall, the results indicate that the proposed MOALNS variants consistently outperform the competing approaches across all problem instances and performance indicators. Although the MOALNS variants generally require higher computational times than the adaptive baseline MOEAs, particularly for larger instances, they consistently achieve superior solution quality. This additional computational cost is mainly attributed to the higher complexity of the MOALNS repair operators compared to the crossover and mutation operators used in evolutionary algorithms. In particular, some repair mechanisms require repeated evaluation of candidate assignments within the solution reconstruction process, which increases the overall computational overhead.

Among the two variants, NSALNS demonstrates the most robust performance, achieving the highest HV values and the lowest Δ values across all tested instances. In contrast, SMOALNS obtains the lowest IGD values across all tested instances. Regarding NSACSA/D, the algorithm achieves higher HV values than all adaptive baseline variants and EMOABC in every instance. However, this advantage does not consistently hold also for the remaining indicators. Moreover, NSACSA/D exhibits substantially higher execution times than all other algorithms. Based on the adopted lexicographic ranking of performance indicators, NSALNS is

■ **Table 2** Same Generations Experiment.

Algorithm	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓
40-10					45-10				60-20			
NSAGAI	13.4	0.800	0.700	0.090	18.0	0.801	0.726	0.089	50.0	0.670	0.776	0.120
NSAGAIH	13.5	0.790	0.836	0.093	18.4	0.786	0.841	0.089	50.9	0.659	0.955	0.122
SPAEA2	13.8	0.782	0.734	0.122	18.6	0.793	0.723	0.108	50.1	0.662	0.832	0.136
MOAEA/D	14.1	0.799	0.764	0.085	19.3	0.791	0.745	0.080	50.5	0.644	0.676	0.126
EMOABC	7.5	0.710	0.683	0.137	10.7	0.636	0.792	0.166	43.6	0.616	0.762	0.155
NSACSA/D	27.1	0.816	0.783	0.085	35.5	0.815	0.821	0.084	101.9	0.709	0.781	0.108
SMOALNS	12.2	1.056	0.573	0.038	16.9	1.075	0.586	0.043	50.2	1.050	0.552	0.035
NSALNS	11.3	1.099	0.539	0.062	15.8	1.156	0.571	0.060	45.9	1.119	0.505	0.051
70-20					80-20				90-20			
NSAGAI	65.4	0.650	0.765	0.115	84.0	0.747	0.839	0.108	105.8	0.766	0.717	0.081
NSAGAIH	66.3	0.651	0.886	0.112	85.6	0.742	0.926	0.107	104.8	0.765	0.903	0.084
SPAEA2	66.0	0.646	0.842	0.125	85.5	0.745	0.894	0.118	106.5	0.762	0.863	0.089
MOAEA/D	66.0	0.629	0.693	0.112	85.3	0.735	0.692	0.129	101.4	0.738	0.784	0.085
EMOABC	69.9	0.578	0.703	0.154	95.0	0.708	0.777	0.132	136.7	0.631	0.718	0.151
NSACSA/D	135.5	0.670	0.785	0.102	178.9	0.779	0.845	0.093	214.2	0.769	0.828	0.076
SMOALNS	71.4	1.046	0.576	0.035	104.9	1.094	0.606	0.031	141.7	1.084	0.596	0.042
NSALNS	66.6	1.108	0.506	0.059	97.7	1.142	0.561	0.050	133.4	1.171	0.495	0.056
100-25					120-30				135-30			
NSAGAI	162.8	0.682	0.786	0.088	249.4	0.607	0.798	0.123	316.3	0.654	0.750	0.096
NSAGAIH	163.8	0.669	0.912	0.088	251.3	0.604	0.893	0.122	321.1	0.647	0.896	0.099
SPAEA2	161.8	0.679	0.897	0.098	248.6	0.598	0.905	0.133	314.7	0.654	0.845	0.109
MOAEA/D	158.9	0.667	0.729	0.083	247.2	0.585	0.723	0.120	305.3	0.619	0.721	0.098
EMOABC	186.8	0.630	0.768	0.121	299.6	0.548	0.740	0.157	394.9	0.598	0.680	0.143
NSACSA/D	333.9	0.691	0.788	0.081	523.1	0.614	0.789	0.111	656.9	0.684	0.861	0.091
SMOALNS	223.0	1.046	0.570	0.031	385.5	1.016	0.546	0.039	562.2	0.998	0.534	0.041
NSALNS	208.7	1.093	0.543	0.058	356.8	1.066	0.485	0.062	514.7	1.095	0.457	0.065
140-35					150-35							
NSAGAI	391.0	0.583	0.781	0.148	464.0	0.626	0.819	0.112				
NSAGAIH	397.7	0.571	0.932	0.155	467.0	0.620	0.945	0.115				
SPAEA2	390.5	0.580	0.871	0.157	464.8	0.624	0.918	0.117				
MOAEA/D	388.8	0.546	0.714	0.153	455.5	0.598	0.756	0.124				
EMOABC	473.7	0.558	0.776	0.172	547.5	0.580	0.727	0.151				
NSACSA/D	848.1	0.589	0.798	0.143	950.6	0.635	0.868	0.104				
SMOALNS	680.1	1.046	0.617	0.032	813.2	1.014	0.578	0.038				
NSALNS	636.0	1.083	0.567	0.055	740.4	1.062	0.526	0.062				

identified as the best-performing algorithm when considering the same number of generations.

[EXP-III.b] *Time Limit Experiment.* Since NSALNS requires higher computational effort than the baseline methods and EMOABC, this experiment assumes a fixed runtime budget. Specifically, the termination criterion is defined separately for each benchmark set according to the average computational time required by NSALNS to complete that benchmark set. It is then assessed whether the other algorithms, with lower computational requirements, could achieve competitive performance when provided with an equivalent computational budget. The results, reported in Table 3, confirm the trends observed in the previous experiments. The MOALNS variants continue to deliver the strongest overall performance across all test cases and evaluation metrics. Furthermore, NSACSA/D consistently outperforms the adaptive baseline MOEAs and EMOABC in terms of HV, although its performance w.r.t. the remaining indicators exhibits greater variability. Overall, these results further highlight the robustness and effectiveness of the MOALNS framework.

Two additional observations emerge from the experimental results. First, NSAGAI achieves higher HV values than NSAGAI in 21 out of 22 experiments presented in Tables 2 and 3. This is consistent with similar observations in the literature (e.g., [11]) indicating that, although NSAGAI is regarded as more suitable for problems with more than 3 objectives due to its environmental selection mechanism, it does not necessarily outperform NSAGAI. Second, the comparison between NSALNS and NSAGAI highlights the importance of the variation operators employed. Since the two algorithms differ only in their underlying

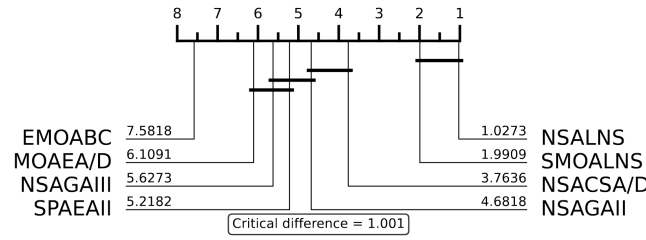
■ **Table 3** Time Limit Experiment.

Algorithm	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓
40-10					45-10				60-20			
NSAGAI	12	0.776	0.662	0.081	16	0.765	0.767	0.081	45	0.645	0.790	0.117
NSAGAIH		0.766	0.838	0.083		0.754	0.847	0.082		0.630	0.955	0.119
SPAEA2		0.755	0.736	0.098		0.754	0.714	0.106		0.638	0.837	0.132
MOAEA/D		0.756	0.748	0.084		0.753	0.706	0.071		0.614	0.680	0.122
EMOABC		0.690	0.671	0.124		0.635	0.810	0.142		0.593	0.771	0.153
NSACSA/D		0.781	0.794	0.077		0.776	0.815	0.076		0.676	0.761	0.101
SMOALNS		1.012	0.586	0.056		0.990	0.602	0.065		1.006	0.589	0.055
NSALNS		1.089	0.524	0.065		1.134	0.591	0.059		1.108	0.497	0.056
70-20					80-20				90-20			
NSAGAI	67	0.687	0.766	0.100	98	0.751	0.804	0.088	134	0.795	0.737	0.077
NSAGAIH		0.675	0.873	0.098		0.748	0.968	0.086		0.790	0.965	0.082
SPAEA2		0.676	0.825	0.110		0.749	0.908	0.102		0.785	0.863	0.092
MOAEA/D		0.660	0.691	0.094		0.735	0.702	0.099		0.760	0.736	0.081
EMOABC		0.599	0.714	0.142		0.700	0.774	0.117		0.649	0.771	0.145
NSACSA/D		0.708	0.779	0.086		0.772	0.859	0.082		0.798	0.835	0.079
SMOALNS		0.988	0.624	0.057		1.063	0.629	0.041		1.050	0.586	0.062
NSALNS		1.081	0.490	0.065		1.143	0.539	0.048		1.171	0.517	0.063
100-25					120-30				135-30			
NSAGAI	209	0.678	0.732	0.073	357	0.637	0.784	0.093	515	0.682	0.739	0.066
NSAGAIH		0.673	0.882	0.073		0.625	0.880	0.095		0.662	0.882	0.068
SPAEA2		0.672	0.888	0.084		0.626	0.866	0.103		0.677	0.872	0.073
MOAEA/D		0.668	0.726	0.068		0.623	0.731	0.085		0.650	0.744	0.061
EMOABC		0.623	0.730	0.109		0.582	0.719	0.131		0.608	0.691	0.114
NSACSA/D		0.694	0.790	0.067		0.647	0.781	0.084		0.703	0.894	0.062
SMOALNS		0.972	0.641	0.044		0.989	0.606	0.052		0.965	0.600	0.056
NSALNS		1.083	0.557	0.057		1.080	0.528	0.074		1.089	0.476	0.070
140-35					150-35							
NSAGAI	636	0.644	0.775	0.087	741	0.656	0.782	0.076				
NSAGAIH		0.630	0.958	0.090		0.652	0.958	0.077				
SPAEA2		0.641	0.917	0.095		0.657	0.919	0.082				
MOAEA/D		0.623	0.727	0.084		0.635	0.724	0.076				
EMOABC		0.605	0.783	0.115		0.612	0.732	0.117				
NSACSA/D		0.656	0.806	0.084		0.677	0.875	0.074				
SMOALNS		1.031	0.669	0.039		0.970	0.648	0.048				
NSALNS		1.114	0.545	0.061		1.074	0.479	0.060				

search operators, the observed performance gap can be largely attributed to this component. The superior performance of NSALNS suggests that the destroy–repair paradigm is better suited to the moDDBASPTw than conventional crossover and mutation operators. A likely explanation is the highly constrained nature of berth allocation schedules, where problem-specific neighbourhood modifications can exploit structural information more effectively and preserve feasibility more reliably than recombination-based search. Appendix C presents the fraction of solutions contributed by each algorithm to the reference Pareto front.

[EXP-IV] Statistical Comparison. To assess whether the observed performance differences among the algorithms are statistically significant, a non-parametric Friedman test was conducted over the instances reported in Table 2, using HV as the performance metric. All algorithms were executed under the same number of generations, as in [EXP-III.a]. For each instance, algorithms were first ranked according to their HV values, and the Friedman test was then applied to these ranks to evaluate the null hypothesis that all methods perform equivalently. The Friedman test ($n = 110, k = 8$) rejected the null hypothesis ($\chi^2 = 601.4, p < 0.05$), indicating statistically significant differences among the compared algorithms.

To further identify which methods differ, we performed a post-hoc Nemenyi test and visualized the results using a critical difference (CD) diagram: the algorithms are ordered according to their average ranks across all instances. Methods connected by a horizontal line do not exhibit statistically significant differences at the 95% confidence level; methods not



■ **Figure 1** Critical difference diagram based on the Friedman test and Nemenyi post-hoc analysis for hypervolume performance across all benchmark instances.

connected by such a line are considered significantly different. Figure 1 shows that **NSALNS** achieves the best average rank (1.0273), followed closely by **SMOALNS** (1.9909); these two methods are not significantly different from each other, but both significantly outperform all remaining algorithms. **NSACSA/D** (3.7636) and **NSAGAI** (4.6818) form a second pairwise non-significant group. **NSAGAI** also overlaps with a third group comprising **NSAGAI** (5.6273) and **SPAEA2** (5.2182), and these two in turn overlap with a fourth group that includes **MOAEA/D** (6.1091), reflecting a chain of pairwise non-significant differences across the mid-ranked methods rather than a single homogeneous cluster. **EMOABC**, with the worst average rank (7.5818), is not connected to any other method and is therefore significantly outperformed by all other algorithms in the comparison. Overall, the CD analysis confirms that **NSALNS** and **SMOALNS** are the most consistently effective methods, with no statistically significant difference between them, while **NSACSA/D**, **NSAGAI**, **NSAGAI**, **SPAEA2**, and **MOAEA/D** form an overlapping mid-tier with no sharp boundaries between adjacent methods, and **EMOABC** performs significantly worse than every other algorithm tested. The same analysis for Δ and IGD confirmed the significant superiority of **MOALNS** variants over all other algorithms.

[EXP-V] Component ablation study for NSALNS. To investigate the contribution of individual destroy and repair operators within **NSALNS**, we conducted an ablation study in which the complete algorithm was compared against variants obtained by removing one operator at a time. For destroy operators, the results indicate that all operators contribute positively to the search process, with relatively small differences among them. The removal of the random removal operator resulted in the smallest performance degradation. This observation is expected, as random removal does not exploit (by design) any specific structural information from the current solution, unlike the more problem-oriented destroy operators. For repair operators, the impact was considerably more heterogeneous. While most repair operators resulted in moderate performance variations when removed, eliminating the k-regret insertion operator caused a substantial deterioration in solution quality. Furthermore, k-regret insertion is the most computationally expensive repair mechanism due to its intensive search procedure during solution reconstruction (cf. Section 3.4). To further analyze the role of k-regret insertion, we compared **NSALNS** with a variant using only this repair operator. Although k-regret insertion was identified as the most influential individual repair mechanism, the resulting algorithm was unable to outperform the complete **NSALNS**. This indicates that the effectiveness of **NSALNS** arises from the complementary interaction among multiple repair strategies rather than from a single dominant operator. Finally, we evaluated reduced variants containing all destroy operators and different subsets of repair operators, specifically all possible pairs of repair operators that include the k-regret insertion operator. The results showed that combining k-regret insertion with the random-weight agg-po-INS-based repair operator achieves performance close to the full **NSALNS**. Therefore,

the results suggest that k-regret insertion and the random-weight agg-po-INS-based repair operator provide particularly valuable contributions to the search process; however, the ablation results also demonstrate that the overall effectiveness of the algorithm relies on the adaptive combination of multiple operators rather than on any individual repair strategy.

5 Conclusions

In this work, we proposed a new multi-objective formulation for moDDBASPTw, consisting of four objectives that capture realistic aspects and operational variables of container terminals. To solve this optimisation model, we designed multiple variation operators and employed well-established MOEAs, which we enhanced with adaptive operator selection mechanisms. Additionally, we introduced an enhanced version of NSCSA, named NSACSA/D, and we adapted ALNS to the multi-objective setting, providing SMOALNS and NSALNS. Our experimental results highlighted the advantage of incorporating adaptive mechanisms into evolutionary algorithms and demonstrated the superior performance of SMOALNS and NSALNS, which consistently outperformed all other algorithms. These findings provide an interesting insight: in complex multi-objective problems, such as moDDBASPTw, generating new solutions via destroy-and-repair operators can lead to higher-quality solutions and faster convergence than relying solely on classical variation mechanisms. Future work will investigate different adaptivity mechanisms for operator selection and will explore their impact on convergence speed, solution quality, and stability in dynamic and large-scale problem instances.

References

- 1 Piraeus Container Terminal S. A. Piraeus container terminal. URL: www.pct.com.gr.
- 2 Tobias Buer and Herbert Kopfer. A pareto-metaheuristic for a bi-objective winner determination problem in a combinatorial reverse auction. *Computers & Operations Research*, 41:208–220, 2014. doi:10.1016/j.cor.2013.04.004.
- 3 Kamal Chakkarapani, Thyagarajan Thangavelu, Kalpana Dharmalingam, and Pragadheeshwaran Thandavarayan. Multiobjective design optimization and analysis of magnetic flux distribution for slotless permanent magnet brushless dc motor using evolutionary algorithms. *Journal of Magnetism and Magnetic Materials*, 476:524–537, 2019. doi:10.1016/j.jmmm.2019.01.029.
- 4 C. Y. Cheong, K. C. Tan, D. K. Liu, and C. J. Lin. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research*, 180(1):63–103, November 2010. doi:10.1007/S10479-008-0493-0.
- 5 Luciano P. Cota, Frederico G. Guimarães, Roberto G. Ribeiro, Ivan R. Meneghini, Fernando B. de Oliveira, Marcone J.F. Souza, and Patrick Siarry. An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem. *Swarm and Evolutionary Computation*, 51:100601, 2019. doi:10.1016/j.swevo.2019.100601.
- 6 Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6(2):182–197, 2002. doi:10.1109/4235.996017.
- 7 Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evol. Comput.*, 18(4):577–601, 2014. doi:10.1109/TEVC.2013.2281535.
- 8 Amir M. Fathollahi-Fard, Peng Wu, Guangdong Tian, Dexin Yu, Tongzhu Zhang, Jianwei Yang, and Kuan Yew Wong. An efficient multi-objective adaptive large neighborhood search algorithm for solving a disassembly line balancing model considering idle rate, smoothness, labor cost, and energy consumption. *Expert Systems with Applications*, 250:123908, 2024. doi:10.1016/j.eswa.2024.123908.

- 9 Amir Hossein Gandomi, Xin-She Yang, and Amir Hossein Alavi. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Eng. Comput.*, 29(1):17–35, 2013. doi:10.1007/S00366-011-0241-Y.
- 10 Zhi-Hua Hu. Multi-objective genetic algorithm for berth allocation problem considering daytime preference. *Computers & Industrial Engineering*, 89:2–14, 2015. Maritime logistics and transportation intelligence. doi:10.1016/j.cie.2015.04.035.
- 11 Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052. IEEE, 2016. doi:10.1109/CEC.2016.7744174.
- 12 Konstantinos Karathanasis, Spyros Kontogiannis, and Christos Zaroliagis. Task orchestration in the cloud continuum via multi-objective evolutionary algorithms. In *International Symposium on Algorithmic Aspects of Cloud Computing*, pages 73–93. Springer, 2025. doi:10.1007/978-3-032-13744-9_6.
- 13 Konstantinos Karathanasis, Spyros C. Kontogiannis, Asterios Pegos, Vasileios Sofianos, and Christos D. Zaroliagis. Vrp-inspired techniques for discrete dynamic berth allocation and scheduling. In *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 137 of *OASICS*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/OASICS.ATMOS.2025.6.
- 14 Abdelfettah LABDIAD. Empirical study of mopso, nsga ii, aco and alns comparison in multiobjective vehicle routing problem with flexible time windows. *IAENG International Journal of Computer Science*, 52(10), 2025.
- 15 Andrew Lim. The berth planning problem. *Operations Research Letters*, 22(2):105–110, 1998. doi:10.1016/S0167-6377(98)00010-8.
- 16 Fei Liu, Chengyu Lu, Lin Gui, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Heuristics for vehicle routing problem: A survey and recent advances. *CoRR*, abs/2303.04147, 2023. doi:10.48550/arXiv.2303.04147.
- 17 Xiaomeng Ma and Xujin Pu. Low-carbon berth allocation: An analysis of the effectiveness of an enhanced multi-objective artificial bee colony algorithm based on a case study. *Ocean & Coastal Management*, 261:107529, 2025. doi:10.1016/j.ocecoaman.2024.107529.
- 18 Rui Qi, Jun qing Li, and Xiao feng Liu. A knowledge-driven multiobjective optimization algorithm for the transportation of assembled prefabricated components with multi-frequency visits. *Automation in Construction*, 152:104944, 2023. doi:10.1016/j.autcon.2023.104944.
- 19 Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. doi:10.1287/trsc.1050.0135.
- 20 Zhongshi Shao, Dechang Pi, and Weishi Shao. A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. *Knowledge-Based Systems*, 165:110–131, 2019. doi:10.1016/j.knosys.2018.11.021.
- 21 Xin Wen, Qiong Chen, Yu-Qi Yin, Yui-yip Lau, and Maxim A. Dulebenets. Multi-objective optimization for ship scheduling with port congestion and environmental considerations. *Journal of Marine Science and Engineering*, 12(1), 2024. doi:10.3390/jmse12010114.
- 22 Mingyuan Yue, Yubing Wang, Siqing Guo, Lei Dai, and Hao Hu. A multi-objective optimization study of berth scheduling considering shore side electricity supply. *Ocean & Coastal Management*, 261:107500, 2025. doi:10.1016/j.ocecoaman.2024.107500.
- 23 Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.*, 11(6):712–731, 2007. doi:10.1109/TEVC.2007.892759.
- 24 Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103, 2001.

A

 Notation & Formulation

This section presents the mathematical formulations of the objectives and briefly describes the constraints. For further details, see the single-objective formulation in [13]. The first objective function (2) aims to minimise the total waiting time, port stay duration, and departure delay of each vessel within the port. The second objective function (3) seeks to minimise both the fuel consumption and the CO₂ emissions generated both during a vessel's entry to and exit from the port and throughout its stay within the port. The third objective function (4) focuses on minimising the total operational cost incurred by the port for the berthing and servicing of a vessel. The fourth objective function (5)¹ aims to minimise load variation across berthing positions, that is, to distribute the workload evenly among berths, thereby reducing potential equipment wear and, consequently, limiting additional maintenance costs incurred by the port.

■ **Table 4** Notation. **Boldfaced** symbols indicate decision variables.

Symbol	Description
$i, k \in S, j \in B$	Indices as subscript for vessels from S and as superscript for berths from B .
$x_{i,k}^j$	1 if $i, k \in S$ are both assigned to berth $j \in B$ for their service, with k being served right after i , 0.0 otherwise.
t_i^j	Mooring time of $i \in S$ at berth $j \in B$ to start being served.
t_i^{ld}	Late departure time for $i \in S$ from its assigned berth.
e_i	CO ₂ Emission Factor for $i \in S$ at its assigned berth.
T_i^{ea}, T_i^{rd}	Estimated Time of Arrival and Requested Time of Departure for $i \in S$.
H_i^j, H_i^-	Handling time for $i \in S$ at berth $j \in B$, Worst Handling time for $i \in S$ across all berths.
$F_i^{wt}, F_i^{ht}, F_i^{SC}$	Fuel Consumption rate while waiting to moore and Fuel Consumption rate while vessel being serviced for $i \in S$. Specific Fuel Consumption for auxiliary engines (tn/kWh).
N_i, N^j	Number of cranes requested by vessel $i \in S$. Number of cranes operating at berth $j \in B$.
K	Time of a single crane to load/discharge a unit quantity.
Q_i, Q_i^f	Cargo Quantity of vessel $i \in S$, average load factor of the containers on vessel $i \in S$.
$P_i^t, P_i^e, P_i^c, P_i^l$	Penalties for not servicing $i \in S$ for each objective respectively.
L_i, D_i	Length and draft of vessel $i \in S$, respectively.
L^j, D^j	Length and draft of berth $j \in B$, respectively.
C^j, C^-	Operational cost for $j \in B$ and worst operational cost across B .
T^{dp}, A	Delay Policy (time units), Fuel Consumption Coefficient
p^{CO_2}, h^{CO_2}	CO ₂ /hour cost generated by vessel operation in port and CO ₂ cost for serving one TEU.
Θ_b, Θ_c	Berth reservation fee(€/m/hour) and crane operation fee (€/hour).

The model is subject to a standard set of spatio-temporal constraints. Each vessel may be assigned to at most one berth. Spatial feasibility is enforced through berth length and draft compatibility. The handling time of vessel i depends on its cargo quantity Q_i , the time K required for a single crane to handle one unit of cargo, the number N^j of operational cranes available at berth j , and the maximum number of cranes that vessel i can effectively utilise. The actual number of cranes employed during service is given by the minimum of the berth's available cranes and the vessel's crane requirement. Accordingly, the handling time of vessel i at berth j is proportional to its cargo quantity divided by this effective number of cranes, scaled by K . This value is precomputed for each vessel–berth pair (i, j) and stored as the parameter H_i^j , obtained by applying a ceiling operation to preserve data integrity.

¹ Although 5 is presented as nonlinear formulation, it can be linearized using McCormick envelopes.

$$\min \sum_{i \in S} \left[\sum_{j \in B} \left(t_i^j - T_i^{ea} \right) + (t_i^{ld} + h_i) + P_i^t \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (2)$$

$$\min \sum_{i \in S} \left[\sum_{j \in B} \left(F_i^{wt} \cdot (t_i^j - T_i^{ea}) \right) + (F_i^{ht} \cdot h_i + e_i) + P_i^e \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (3)$$

$$\min \sum_{i \in S} \left[\sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} (x_{i,k}^j \cdot h_i \cdot C^j) + P_i^c \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (4)$$

$$\min \sum_{j \in B} \left[\sum_{i \in S} \sum_{k \in S \cup \{d\}: k \neq i} \left(H_i^j \cdot x_{i,k}^j - \frac{1}{|B|} \sum_{j' \in B} H_i^{j'} \cdot x_{i,k}^{j'} \right) \right]^2 + \sum_{i \in S} \left[P^l \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (5)$$

■ **Figure 2** Objective functions for moDDBASptw. Parameters in **bold** indicate decision, or temporal variables. Parameters in capital letters indicate constants, which are provided as input.

The CO₂ emissions of vessel i , depend on its handling time H_i^j , its cargo quantity Q_i , the hourly CO₂ cost p^{CO_2} generated by vessel operation across the whole port and CO₂ cost h^{CO_2} for servicing. These values are again precomputed and stored. A hard time-window restriction is imposed, ensuring that any vessel i that is granted service is handled within the interval $[T_i^{ea}, T_i^{rd} + T^{dp}]$. The simultaneous servicing of two different vessels at the same berth is prohibited. Finally, the departure delay t_i^{ld} is defined as the maximum between zero and the difference between the actual departure time and T_i^{rd} .

The penalties for all objectives are designed so that the algorithm is never incentivized to leave a vessel unassigned. The time-related penalty $P_i^t = (T_i^{rd} + T^{dp} - T_i^{ea}) + T^{dp} + \max_j H_i^j = T_i^{rd} + 2T^{dp} - T_i^{ea} + H_i^-$, provides a conservative upper bound on the total time vessel i may spend in the port if it cannot be serviced. It is obtained by summing the maximum possible waiting time, the maximum allowable departure delay, and the maximum handling time. Here, $H_i^- = \max_{j \in B} H_i^j$ denotes the longest handling time of vessel i across all berths. Similarly, the energy-related penalty $P_i^e = (T_i^{rd} + T^{dp} - T_i^{ea}) F_i^{wt} + H_i^- F_i^{ht} + E_i^-$, defines a conservative upper bound on the maximum CO₂ emissions that vessel i could generate if left unassigned. This penalty consists of three components. The first term captures emissions during waiting, computed as the product of the maximum possible waiting time and the waiting fuel consumption rate F_i^{wt} . The second term accounts for emissions during service, given by the maximum handling time H_i^- multiplied by the service fuel consumption rate F_i^{ht} . The third term (E_i^-) represents the maximum emissions associated with crane operations.

The fuel consumption rates are computed as follows: $F_i^{wt} = \epsilon \cdot F^{SC} \cdot Q_i \cdot Q_i^f$, $F_i^{ht} = 0.3 \cdot F_i^{wt}$. Here, ϵ is a scaling parameter, F^{SC} denotes the specific fuel consumption of the auxiliary engines, and Q_i is the total cargo quantity of vessel i , expressed in Twenty-foot Equivalent Units (TEUs). The parameter Q_i^f represents the average container load factor, i.e., the average fraction of capacity utilised by the containers on board. For instance, $Q_i^f = 0.8$ indicates that the containers of vessel i are, on average, filled to 80% of their capacity. The fuel consumption rate during service is conservatively defined as 30% of the consumption rate during waiting. The operational cost penalty is defined as the product of the maximum handling time and the maximum berthing cost: $P_i^c = \max_j H_i^j \cdot \max_j C^j = H_i^- \cdot C^-$ where the berthing cost at berth j is given by $C^j = \Theta_b \cdot L^j + \Theta_c \cdot N^j$. This cost consists of the berth reservation fee multiplied by the berth length and the crane operation fee multiplied by the number of cranes assigned to that berth.

Finally, the load-related penalty P^l provides a conservative upper bound on workload imbalance across the terminal. It is defined as the variance of berth loads under a worst-case assignment in which all vessels are hypothetically allocated to the slowest berth j_0 . This extreme scenario induces uniformly large handling times and maximizes load imbalance across berths, yielding an upper-bound reference for load dispersion. Let H_i^- denote the handling time of vessel i when processed at berth j_0 . Under this assumption, the load penalty is computed as $P^l = \frac{|B|-1}{|B|} \left(\sum_{i \in S} H_i^- \right)^2$.

B Baseline vs. Adaptive Variants

Table 5 Baseline vs. Adaptive Variants Experiment. Red boldface indicates the best value of the metric for the benchmark set.

Algorithm	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓
40-10				45-10				60-20				
NSGAII	17.7	1.085	0.699	0.045	23.3	1.098	0.733	0.058	59.7	1.063	0.818	0.046
NSAGAI	13.9	1.098	0.693	0.039	18.6	1.130	0.701	0.048	50.3	1.069	0.779	0.038
NSGAIII	17.5	1.065	0.836	0.049	23.3	1.082	0.753	0.052	59.3	1.051	0.812	0.043
NSAGAI	14.1	1.084	0.832	0.039	18.8	1.117	0.818	0.040	51.0	1.052	0.942	0.043
SPEA2	18.2	1.063	0.758	0.080	24.0	1.081	0.763	0.087	60.1	1.054	0.836	0.057
SPAEA2	14.2	1.087	0.751	0.079	19.2	1.104	0.750	0.099	50.5	1.057	0.828	0.073
MOEA/D	18.2	1.076	0.743	0.043	24.4	1.041	0.753	0.063	57.0	1.064	0.650	0.038
MOEA/D	15.0	1.091	0.755	0.027	20.1	1.120	0.697	0.034	51.7	1.065	0.641	0.036
70-20				80-20				90-20				
NSGAII	82.5	1.094	0.707	0.043	108.7	1.166	0.779	0.031	133.2	1.136	0.694	0.047
NSAGAI	66.1	1.107	0.705	0.043	85.5	1.183	0.781	0.028	107.3	1.158	0.713	0.040
NSGAIII	82.1	1.072	0.855	0.040	108.8	1.160	0.950	0.031	133.2	1.132	0.713	0.035
NSAGAI	67.6	1.102	0.757	0.035	86.6	1.164	0.789	0.029	106.7	1.139	0.766	0.040
SPEA2	82.3	1.084	0.900	0.066	108.7	1.175	0.871	0.041	133.4	1.125	0.804	0.067
SPAEA2	67.1	1.099	0.789	0.057	86.3	1.178	0.828	0.039	107.5	1.147	0.788	0.046
MOEA/D	79.1	1.052	0.693	0.043	102.6	1.152	0.688	0.059	120.2	1.046	0.714	0.060
MOEA/D	67.6	1.085	0.666	0.026	88.9	1.157	0.700	0.040	103.9	1.109	0.745	0.046
100-25				120-30				135-30				
NSGAII	205.4	1.084	0.698	0.037	316.7	1.088	0.698	0.044	407.0	1.045	0.700	0.053
NSAGAI	164.2	1.100	0.736	0.036	248.9	1.105	0.749	0.040	317.0	1.057	0.694	0.050
NSGAIII	204.9	1.076	0.721	0.032	316.9	1.069	0.710	0.040	403.5	1.048	0.668	0.044
NSAGAI	166.7	1.082	0.904	0.034	251.9	1.089	0.877	0.041	322.3	1.064	0.801	0.042
SPEA2	204.1	1.080	0.748	0.059	316.9	1.088	0.842	0.064	405.5	1.046	0.791	0.067
SPAEA2	164.6	1.083	0.762	0.057	250.4	1.090	0.857	0.062	319.4	1.052	0.749	0.059
MOEA/D	188.0	1.074	0.716	0.028	282.0	1.063	0.727	0.028	364.5	1.020	0.707	0.036
MOEA/D	162.0	1.089	0.696	0.024	246.3	1.090	0.696	0.025	311.3	1.048	0.674	0.032
140-35				150-35								
NSGAII	503.6	1.064	0.745	0.054	590.5	1.058	0.742	0.046				
NSAGAI	394.9	1.083	0.687	0.045	455.9	1.069	0.693	0.038				
NSGAIII	503.1	1.057	0.739	0.055	586.1	1.044	0.829	0.043				
NSAGAI	399.5	1.062	0.761	0.059	455.4	1.052	0.795	0.037				
SPEA2	504.8	1.063	0.782	0.0457	585.2	1.045	0.800	0.063				
SPAEA2	394.4	1.080	0.731	0.056	449.3	1.062	0.875	0.052				
MOEA/D	454.6	1.022	0.723	0.049	506.1	1.028	0.715	0.052				
MOEA/D	387.2	1.044	0.694	0.038	439.7	1.036	0.749	0.031				

Having identified **RandomParent** as the most effective crossover operator on average, we evaluate the impact of the **ALNS**-inspired adaptive operator selection mechanism on classical MOEAs. We compare each baseline algorithm, configured with the **RandomParent** crossover operator and the **ReInsert** mutation operator, against its corresponding adaptive variant, which dynamically selects among all available crossover and mutation operators.

Table 5 summarizes the comparative results, with all algorithms executed using a population size of 100 for 200 generations. Overall, the adaptive variants demonstrate stronger performance across most problem instances and evaluation metrics. A notable improvement appears in computational time: in all instances, the adaptive variants are substantially faster

than their non-adaptive counterparts. This reduction can be attributed to the adaptive operator selection mechanism, which enables the use of multiple crossover and mutation operators, including operators that are faster than **RandomParent** and **ReInsert**. In terms of solution quality, the adaptive variants achieve in most cases, higher HV, lower Δ and IGD.

C Non-Dominated Solutions

In this section, we present the results of our experimental evaluation on the same benchmark instances and under the same termination criteria described in Section 4, using the **Recall** metric. Recall measures the fraction of solutions in the reference Pareto front that are contributed by a given algorithm. We consider this metric because it is particularly sensitive to the number of non-dominated solutions returned by each algorithm.

Four algorithms, namely **MOAEA/D**, **NSACSA/D**, **EMOABC**, and **SMOALNS**, maintain an external archive of non-dominated solutions. Consequently, the number of solutions they return is not fixed, unlike the remaining algorithms, which always return a population of predetermined size. We therefore investigate how this characteristic affects the Recall metric.

As shown in Table 6, **MOAEA/D** consistently achieves the highest Recall values. In the same generations experiment, **SMOALNS** ranks second in 10 out of the 11 cases, with **NSACSA/D** outperforming it only for case 90–20. **NSALNS** consistently ranks behind these three algorithms, except in the 45–10, 120–30, and 150–35 cases, where it outperforms **NSACSA/D**, while it consistently outperforms all remaining methods. Among the evolutionary algorithms with a fixed population size, **NSGAI**, **NSGAI**, and **SPEA2** achieve the lowest Recall values. Specifically, **NSGAI** outperforms **NSGAI** in 10 of the 11 cases, whereas **NSGAI** is superior only once and outperforms **SPEA2** in all but the 140–35 and 150–35 cases. These results are expected, as the archive-based design of **MOAEA/D**, **SMOALNS**, and **NSACSA/D** enables them to contribute substantially more non-dominated solutions to the reference Pareto front than algorithms that return a fixed-size population. In contrast, although **EMOABC** also maintains an external archive, it consistently achieves the lowest Recall values because the quality of its generated solutions is considerably lower than that of the competing methods, resulting in only a limited contribution to the reference Pareto front.

The results obtained under the time-limit criterion, reported in Table 7, exhibit a different ranking among algorithms. Although **MOAEA/D** again achieves the highest Recall values, **NSALNS** now outperforms **NSACSA/D** in all but four cases and performs comparably to **SMOALNS**, outperforming it in five cases while ranking behind it in six. Furthermore, **NSACSA/D** achieves the second-highest Recall value only in three cases. Among the fixed-population evolutionary algorithms, **NSGAI** outperforms **NSGAI** in all but the 45–10 instance and even surpasses **NSACSA/D** for cases 40–10, 80–20 and 90–20, as well as **SMOALNS** for cases 90–20, 135–30, and 150–35. **SPEA2** consistently achieves lower Recall values than both **NSGAI** and **NSGAI**, while **EMOABC** again obtains the lowest Recall values overall. The different ranking observed under the time-limit criterion can be explained by the computational cost of the algorithms. As discussed in Section 4, the time limit is set equal to the average execution time of **NSALNS** for each instance, as it is the best-performing algorithm in HV and Δ . **NSACSA/D** and **SMOALNS** require substantially more computation per generation than **NSALNS**. Consequently, under a fixed computational budget they complete fewer generations, reducing the number of non-dominated solutions they contribute to the reference Pareto front and, therefore, their Recall values.

An interesting observation is that **NSALNS**, despite achieving the best overall performance in terms of HV and Δ , does not return the largest number of non-dominated solutions. Nevertheless, it produces the second-largest solution sets in several instances of the Time

■ **Table 6** Same Generations Experiment Recall.

Algorithm	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R
	40-10		45-10		60-20		70-20		80-20		90-20	
NSAGAI	13.4	0.051	18.0	0.047	50.0	0.048	65.4	0.046	84.0	0.055	105.8	0.046
NSAGAI	13.5	0.052	18.4	0.039	50.9	0.058	66.3	0.058	85.6	0.059	104.8	0.074
SPAEA2	13.8	0.034	18.6	0.033	50.1	0.038	66.0	0.036	85.5	0.031	106.5	0.032
MOAEA/D	14.1	0.491	19.3	0.569	50.5	0.447	66.0	0.471	85.3	0.447	101.4	0.545
EMOABC	7.5	0.001	10.7	0.001	43.6	0.001	69.9	0.003	95.0	0.002	136.7	0.001
NSACSA/D	27.1	0.083	35.5	0.079	101.9	0.118	135.5	0.122	178.9	0.121	214.2	0.112
SMOALNS	12.2	0.211	16.9	0.152	50.2	0.188	71.4	0.182	104.9	0.190	141.7	0.111
NSALNS	11.3	0.077	15.8	0.081	45.9	0.103	66.6	0.082	97.7	0.095	133.4	0.080
Algorithm	100-25		120-30		135-30		140-35		150-35			
NSAGAI	162.8	0.031	249.4	0.038	316.3	0.043	391.0	0.042	464.0	0.047		
NSAGAI	163.8	0.051	251.3	0.073	321.1	0.078	397.7	0.076	467.0	0.084		
SPAEA2	161.8	0.024	248.6	0.027	314.7	0.027	390.5	0.043	464.8	0.054		
MOAEA/D	158.9	0.528	247.2	0.539	305.3	0.534	388.8	0.414	455.5	0.452		
EMOABC	186.8	0.001	299.6	0.001	394.9	0.002	473.7	0.004	547.5	0.003		
NSACSA/D	333.9	0.143	523.1	0.082	656.9	0.098	848.1	0.096	950.6	0.092		
SMOALNS	223.0	0.150	385.5	0.145	562.2	0.131	680.1	0.232	813.2	0.173		
NSALNS	208.7	0.071	356.8	0.095	514.7	0.085	636.0	0.093	740.4	0.093		

Limit Experiment and, among all algorithms with a fixed output size, consistently attains the highest Recall values across both experimental settings. These results further demonstrate that the quality and distribution of the generated solutions are more influential than their quantity with respect to the HV and Δ indicators.

By contrast, IGD is inherently more sensitive to the density of solutions approximating the reference Pareto front. Consequently, algorithms that contribute a larger number of high-quality non-dominated solutions tend to achieve better IGD values. This explains why SMOALNS outperforms NSALNS in terms of IGD in the experimental evaluation presented in Section 4, despite the superior HV and Δ performance of NSALNS.

■ **Table 7** Time Limit Experiment Recall.

Algorithm	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R
	40-10		45-10		60-20		70-20		80-20		90-20	
NSAGAI		0.052		0.049		0.056		0.051		0.049		0.057
NSAGAI		0.063		0.044		0.070		0.060		0.059		0.088
SPAEA2		0.042		0.030		0.043		0.034		0.030		0.053
MOAEA/D	12	0.560	16	0.650	45	0.519	67	0.564	98	0.593	134	0.599
EMOABC		0.001		0.001		0.004		0.003		0.001		0.001
NSACSA/D		0.058		0.053		0.071		0.081		0.057		0.039
SMOALNS		0.117		0.076		0.113		0.106		0.112		0.065
NSALNS		0.105		0.097		0.125		0.101		0.099		0.099
Algorithm	100-25		120-30		135-30		140-35		150-35			
NSAGAI		0.022		0.041		0.039		0.032		0.041		
NSAGAI		0.041		0.060		0.054		0.058		0.078		
SPAEA2		0.023		0.028		0.033		0.034		0.042		
MOAEA/D	209	0.633	357	0.643	515	0.619	636	0.594	741	0.566		
EMOABC		0.001		0.002		0.001		0.003		0.003		
NSACSA/D		0.128		0.062		0.132		0.096		0.113		
SMOALNS		0.083		0.088		0.048		0.110		0.073		
NSALNS		0.069		0.076		0.074		0.072		0.084		

Finding Maximum-Success Disjoint Paths

Aaron Neugebauer 

Department of Computer Science, University of Würzburg, Germany

Image Processing Department, Fraunhofer ITWM-Institute for Industrial Mathematics,
Kaiserslautern, Germany

Marie Schmidt 

Department of Computer Science, University of Würzburg, Germany

Abstract

Motivated by challenges occurring in disaster relief operations, we study the problem to find two edge-disjoint paths in a graph whose arcs are labeled with traversability probabilities that maximize the probability that (at least) one of the paths is traversable.

We show that this problem is NP-hard. To solve the problem, we propose to model it as a bi-objective problem, taking traversability probabilities of the two individual paths as objective functions. We show that an optimal solution to our problem is an extreme-supported solution of the so defined bi-objective problem and exploit this property to find optimal solutions. Furthermore, we present two approximation approaches, and compare the approaches experimentally with respect to runtime and solution quality.

2012 ACM Subject Classification Applied computing → Operations research; Mathematics of computing → Graph theory; Mathematics of computing → Graph algorithms; Mathematics of computing → Quasiconvex programming and unimodality; Mathematics of computing → Integer programming; Theory of computation → Shortest paths; Mathematics of computing → Approximation algorithms

Keywords and phrases Route finding under uncertainty, Edge-disjoint paths, Optimization, Exact algorithms, Approximation algorithms, Vectorization

Digital Object Identifier 10.4230/OASIcs.ATMOS.2026.8

Supplementary Material *Software (Source Code)*: <https://github.com/PantomInach/uncertain-road-network-traversal-strategies.git> [29]

archived at `swb:1:dir:08084beb228c3c09cb4a6456e342fc5f853b3874`

1 Introduction

Research on humanitarian logistics, and particularly on the distribution of emergency supplies following disasters is gaining increasing attention in recent years, see, e.g., [1, 27, 3]. Providing disaster relief is often complicated by road damage. To get an overview on the current state of the road infrastructure, air footage can be gathered by drones, planes and helicopters. However, the analysis of this material by human experts is time-consuming and constitutes a bottleneck. Image recognition can help to automate this process, but is prone to uncertainties, i.e. the respective applications cannot provide definite answers but only traversability *probabilities* for the roads in question. [37, 21]

Motivated by the sketched situation, in this paper we investigate the following optimization problem: given a graph with a dedicated source and a dedicated destination and edge labels that denote the probability that an edge is traversable, find two edge-disjoint source-destination paths that maximize the probability to reach the destination on either path. We describe the underlying road network by an (directed) undirected graph $G = (V, E)$ with vertices V and edges E . If the road network contains roads that cannot be traversed from both sides, we model the road network using directed edges, otherwise, the edges are undirected. To incorporate the traversability of the road segments, each edge $e \in E$ also gets a probability $p_e \in (0, 1]$, which are independent from each other. A probability of 1 denotes



© Aaron Neugebauer and Marie Schmidt;

licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 8; pp. 8:1–8:18



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

that the corresponding road segment is always traversable. A probability of 0 would indicate that the road segment is impassable, but we exclude such edges from our problem. In total the tuple (V, E, p) gives an *uncertain graph*.

We call $P = ((v_1, v_2), (v_2, v_3), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k))$ a *path* in an undirected graph $G = (V, E^{\leftrightarrow})$ if $\{v_i, v_{i+1}\} \in E^{\leftrightarrow}$ for $i = 1, \dots, k-1$ or in a directed graph $G = (V, E^{\rightarrow})$ if $(v_i, v_{i+1}) \in E^{\rightarrow}$ for $i = 1, \dots, k-1$. An s - t path starts with the vertex s and ends in the vertex t . The *probability* $\pi(P)$ of a path P is given by $\pi(P) = \prod_{(u,v) \in P} p_{\{u,v\}}$ in an undirected graph. For a directed graph, let $\pi(P) = \prod_{e \in P} p_e$. If not further restrictions are given, the edge set E of a graph can be either directed or undirected.

► **Definition 1** (MAXIMUMSUCCESSDISJOINTPATHSPROBLEM (MSDPP)). *Given (G, p, s, t) with $G = (V, E)$, where (V, E, p) is an (directed) uncertain graph and $s, t \in V$, find two (directed) edge-disjoint s - t paths P_1, P_2 in G that maximize the success probability*

$$\tau(P_1, P_2) = \pi(P_1) + (1 - \pi(P_1))\pi(P_2) = \pi(P_1) + \pi(P_2) - \pi(P_1) \cdot \pi(P_2)$$

of traversing from s to t over the main path P_1 and else the alternative path P_2 .

We show NP-hardness of MSDPP for directed and undirected graphs in Section 3.

Our solution approaches for the MSDPP are based on a key observation: any optimal solution to the MSDPP is an extreme-supported efficient solution to an appropriately defined bi-objective optimization problem. We prove this property (for a more general problem class) in Section 4.1. We propose two exact solution approaches: Sections 4.2 and 4.3. The algorithm proposed in Section 4.2 iterates the most-certain paths and computes subsequently a sufficient subset of efficient solutions, while the second approach, presented in Section 4.3, enumerates extreme supported solutions by ILP-based dichotomic search.

Furthermore, we propose an approximation algorithm and a heuristic that compute specific extreme supported solutions. The approximation algorithm presented in Section 4.4 has a $1/2$ approximation ratio, however, the algorithm has exponential time complexity. The heuristic presented in Section 4.5 is based on Suurballe's algorithm to compute edge-disjoint paths, but no approximation guarantee in the input can be given. We can, however, bound the quality of the solution found by properties of an optimal solution. We compare the runtime of the proposed algorithms and the performance in terms of solution quality for the approximation algorithms in Section 5.

2 Related Work

Shortest paths under uncertainty

There are various uncertain variants of the shortest path problem, spanning stochastic, robust, and online optimization perspectives. While many robust and stochastic optimization approaches concentrate on finding a *single* path that is good under uncertain edge lengths (see, e.g., [28, 32, 36]), authors have also proposed two-stage approaches, which allow a single adjustment of the initially planned path (see, e.g., [8, 18]), and online approaches that gain knowledge while crossing the network and subsequently update the proposed path “on the go” (see, e.g., [4, 42, 35]). Particularly related to our problem is the problem to find a single *most-certain* path in an (undirected or directed) uncertain graph (V, E, p) . As explained by Christofides [10], this problem can be solved by applying Dijkstra's algorithm [11] on the graph where edge probabilities p_e are replaced by edge weights $-\ln p_e$.

Multi-objective optimization and vectorization

Multi-objective optimization aims to find solutions to optimization problems with multiple objective functions. Instead of finding a single solution that optimizes one objective function, in multi-objective optimization one aims to find a representative set of all solutions whose performance (with respect to all considered objective functions) is not *dominated* by any other solution. Concretely, a solution x is called *efficient* if there is no solution x' such that x' is at least as good as x with respect to each objective function and better with respect to at least one. Often, multi-objective optimization problems are solved by applying *scalarization* techniques that transform a multi-objective problem to a single-objective one. One example of such a technique is *dichotomic search* which varies the weights in a *weighted sum* of the different objective functions in a way that allows one to find a subset of efficient solutions, namely the ones characterized as belonging to the efficient solutions on extreme points of the convex hull of solution vectors in $\mathbb{R}^m$. See Ehrgott [12] for an introduction to multi-objective optimization and the concepts and method described here.

Vectorization relates to the approach to transform an originally single-objective optimization problem into a multi-objective problem such that an optimal solution to the single-objective problem can be found by solving the multi-objective problem. This relationship can be exploited in robust optimization with a finite scenario set $\mathcal{U}$, where it is easy to show that there is always an optimal solution that is efficient with respect to the $|\mathcal{U}|$ objective functions obtained from evaluating the solution in specific scenarios, though not every robust optimal solution is efficient in this sense; see Inacu and Trichakis [23]. It has also been exploited for several robust and stochastic variants of the shortest path problem; see [36, 9, 33, 34]. However, we are not aware of a general description of this approach in the literature, hence we provide one and prove its correctness in Section 4.1 and then apply it to our problem.

Edge-disjoint paths

The problem to decide whether an undirected graph contains disjoint paths between k vertex-tuples (s_i, t_i) is known as the *k-disjoint paths problem* (*k-DPP*) [19]. In directed graphs, it is called *k-directed disjoint paths problem* (*k-dDPP*) [19]. With the further restriction that the respective s_i-t_i paths should be shortest s_i-t_i paths in a graph with edge lengths $\ell: E \rightarrow \mathbb{R}$, this gives the disjoint shortest path problem (*k-DSPP* and *k-dDSPP*). The computational complexity of these problems depends on the assumptions made on edge lengths (being harder in some cases if edge lengths are allowed to be 0), the number of paths k that need to be found, and whether the underlying graph is directed or not. In Table 1 we restate an overview table on the complexity results for these decision problems from Gottschau et al. [19]. For symmetric directed graphs – for each directed edge exists an opposite edge – *k-dDPP* is solvable in polynomial time [24]. Note that similar results hold for the problems to find vertex-disjoint paths, since the respective decision problems can be transformed into each other.

A further problem variant, in which two edge-disjoint s_1-t_1 and s_2-t_2 paths in an undirected graph should be computed, such that the total length is minimum, can be solved in polynomial time [7]. For the case $s_1 = s_2$ and $t_1 = t_2$ Suurballe's algorithm [39, 6] can be used. Eilam-Tzoref [13] prove that another problem variant, the *two disjoint one shortest path problem* (2D1SPP), is NP-hard both in directed and in undirected graphs. Here, the question is to decide whether there exist two edge-disjoint paths P_1 from s_1 to t_1 and P_2 from s_2 to t_2 , where P_1 should be a shortest s_1-t_1 -path. It is also NP-hard if paths should be vertex-disjoint instead of edge-disjoint.

■ **Table 1** Results on the complexity of the disjoint-path problem from [19].

k	No edge weights $\ell \equiv 0$		Non-negative edge weights ℓ	
	k -DPP	k -dDPP	k -DSPP	k -dDSPP
arbitrary	NP-hard [14]	NP-hard [14]	NP-hard [13]	NP-hard [13]
fixed	P [38]	NP-hard [17]	Open for $\ell > 0$	Open for $\ell > 0$
			Open for $\ell \geq 0$	NP-hard for $\ell \geq 0$ [17]
2	P [38, 41]	NP-hard [17]	P for $\ell > 0$ [13]	P for $\ell > 0$ [5]
			P for $\ell \geq 0$ [19, 26]	NP-hard for $\ell \geq 0$ [17]

3 NP-Hardness

► **Theorem 2.** *MSDPP is NP-hard in both directed and undirected uncertain graphs.*

Proof. We show this statement by reduction to the decision version of our problem from the 2D1SPP which is NP-hard (both in directed and undirected graphs) according to [13]. Since the NP-hardness reduction for 2D1SPP from 3SAT ([25]) in [13] only utilizes integer edge lengths and distinct vertex tuples $(s_1, t_1), (s_2, t_2)$, the 2D1SPP is NP-hard for graphs with integer edge lengths. For the construction we use undirected graph notation, but the construction in a directed graph is analogous.

Consider an instance of the 2D1SPP on an undirected graph $G = (V, E)$ with integer edge lengths ℓ_e and two distinct vertex tuples (s_1, t_1) and (s_2, t_2) . We denote by $\ell(P)$ the length of a path P in G . Denote by L the length of a shortest s_1 - t_1 path in G .

We construct an instance of MSDPP as follows: Let $V' = V \cup \{s, t\}$ and $E' = E \cup \{\{s, s_1\}, \{s, s_2\}, \{t_1, t\}, \{t_2, t\}\}$. Consider the uncertain graph $G' = (V', E', p)$ where, for $e \in E$, $p_e = e^{-\ell_e}$, $p_{\{s, s_1\}} = p_{\{t_1, t\}} = 1$ and $p_{\{s, s_2\}} = p_{\{t_2, t\}} = \varepsilon$ with $0 < \varepsilon < \min\{(\prod_{e \in E} p_e) / 2, \sqrt{(e-1)/(e^{L+1}-1)}\}$. Decide whether there are two edge-disjoint s - t paths with a success probability $\geq e^{-L}$.

Note that for any path P in G , in particular for any s_i - t_i path, P has probability $\pi(P) = \prod_{e \in P} p_e = \prod_{e \in P} e^{-\ell_e} = e^{-\sum_{e \in P} \ell_e} = e^{-\ell(P)}$ in G' .

Assume that there are two edge-disjoint s_1 - t_1 and s_2 - t_2 paths P_1 and P_2 in G such that P_1 is a shortest s_1 - t_1 path ($\ell(P_1) = L$). Define s - t paths P'_1 and P'_2 in G' by appending the edges $\{s, s_i\}$ and $\{t_i, t\}$ to P_i for $i = 1, 2$, respectively. Then

$$\tau(P'_1, P'_2) = e^{-L} + (1 - e^{-L}) \cdot \varepsilon^2 e^{-\ell(P_2)} \geq e^{-L}.$$

On the other hand, consider two edge-disjoint s - t paths in G' with $\tau(P''_1, P''_2) \geq e^{-L}$. Without loss of generality, let P''_1 be the path that goes via s_1 and P''_2 be the path that goes via s_2 . Note that then P''_1 also goes via t_1 , because if it contained $\{t_2, t\}$, we would have

$$\tau(P_1, P_2) \leq \pi(P_1) + \pi(P_2) \leq 2\varepsilon < \prod_{e \in E} p_e \leq e^{-L}.$$

We conclude that P''_i contains edges $\{t_i, t\}$ for $i = 1, 2$.

We now show that the path P'_1 that we obtain from P''_1 by removing edges $\{s, s_1\}$ and $\{t_1, t\}$ corresponds to a path of length L (i.e., of minimum length) in G . Assume, for contradiction, that P'_1 was longer, i.e., $\ell(P'_1) \geq L + 1$. Then, we obtain

$$\begin{aligned}
\tau(P_1'', P_2'') &= \pi(P_2'') + (1 - \pi(P_2''))\pi(P_1'') = \varepsilon^2\pi(P_2') + (1 - \varepsilon^2\pi(P_2'))\pi(P_1') \\
&\leq \varepsilon^2\pi(P_2') + (1 - \varepsilon^2\pi(P_2'))e^{-(L+1)} \\
&= (\varepsilon^2\pi(P_2') \cdot e^{L+1} + 1 - \varepsilon^2\pi(P_2'))e^{-(L+1)} \\
&= (\varepsilon^2\pi(P_2') \cdot (e^{L+1} - 1) + 1)e^{-(L+1)} \\
&\leq (\varepsilon^2 \cdot (e^{L+1} - 1) + 1)e^{-(L+1)} \\
&< \left(\frac{e - 1}{e^{L+1} - 1} \cdot (e^{L+1} - 1) + 1 \right) e^{-1}e^{-L} \\
&\leq ee^{-1}e^{-L} = e^{-L}.
\end{aligned}$$

We can thus conclude that P_1 has length L , and by construction, P_2 (the path corresponding to P_2'') does not share an edge with P_1 . $\blacktriangleleft$

Note that the proof extends to the case where vertex-disjointness of the paths is required since in this case 2D1SPP is also NP-hard according to [13], and to uncertain graphs with edge-probabilities $p: E \rightarrow [0, 1]$.

4 Solution Approaches

In this section, we develop solution approaches for the MSDPP. These are based on an important insight: optimal solutions to the MSDPP are extreme-supported efficient solutions to a suitably defined multi-objective optimization problem. This perspective on the problem, explained in Section 4.1 inspires two exact, an approximation algorithm, and a heuristic approach, which we describe in Sections 4.2–4.5.

4.1 A vectorization approach to the MSDPP

We first explain our vectorization approach to a more general class of problems and prove its correctness in Theorem 3, before we apply it on the MSDPP in Theorem 4.

Consider an optimization problem

$$\begin{aligned}
&\text{maximize} && h(x) \\
&\text{subject to} && x \in X
\end{aligned} \tag{1}$$

where $X \subset \mathbb{R}^n$ is compact and h can be written as $f \circ g$ with f and g having the following properties: 1.) $g: X \rightarrow \mathbb{R}^p$ is continuous. 2.) There exists a compact and convex superset $Y' \supseteq Y = g(X)$, in which f is continuous, monotonously increasing in each argument, and quasi-convex, i.e.,

$$f((1 - \lambda)y + \lambda y') \leq \max\{f(y), f(y')\} \quad \forall y, y' \in Y' \quad \forall \lambda \in [0, 1]. \tag{2}$$

For this function g , we now consider the multi-objective optimization problem

$$\begin{aligned}
&\text{maximize} && g(x) = \begin{pmatrix} g_1(x) \\ \vdots \\ g_p(x) \end{pmatrix} \\
&\text{subject to} && x \in X.
\end{aligned} \tag{3}$$

In multi-objective optimization, a solution for x of Problem (3) is called *efficient* if there is no $x' \in X$ with $g_i(x) \leq g_i(x')$, for all $i \in \{1, \dots, p\}$, and $g_j(x) < g_j(x')$ for some $j \in \{1, \dots, p\}$. If, for an efficient solution x , there exists a $\lambda \in \mathbb{R}_{>0}^p = \{\lambda \in \mathbb{R}^p \mid \forall i \in \{1, \dots, p\}: \lambda_i > 0\}$ such that $x \in \operatorname{argmax}_{x' \in X} \sum_{i=1}^p \lambda_i g_i(x')$, then x is *supported efficient* for Problem (3). A supported efficient solution x of Problem (3) is *extreme-supported efficient* if there are no $\mu \in (0, 1)$ and no supported efficient solutions x_1, x_2 with $x \neq x_1, x \neq x_2$ such that $x = \mu x_1 + (1 - \mu)x_2$.

► **Theorem 3.** *Let x^* be an optimal solution for Problem (1) and denote by X_S the set of extreme-supported efficient solutions to the multi-objective optimization Problem (3). Then there exists a solution $x^S \in X_S$ to the Problem (3) with $h(x^*) = h(x^S)$. If f is strictly quasi-convex on Y' , i.e., if the inequality in (2) is strict, every optimal solution to (1) is extreme-supported efficient for Problem (3).*

Proof. Since X is non-empty and compact and g is continuous, for any $\lambda \in \mathbb{R}_{>0}^p$, the set $\operatorname{argmax}_{x' \in X} \sum_{i=1}^p \lambda_i g_i(x')$ is non-empty and compact and thus the set of extreme-supported non-dominated points, $Y_S := g(X_S)$, is also compact. Then $\operatorname{conv}(Y_S) - \mathbb{R}_{\geq}^p \supset Y$, because all supported non-dominated points $\operatorname{conv}(Y_S)$ are in the convex hull of the extreme-supported non-dominated points. That is, there is a finite set $\{y_1, \dots, y_q\} \subseteq Y_S$, non-negative numbers λ_i , where $i = 1, \dots, q$, with $\sum_{i=1}^q \lambda_i = 1$ and non-negative numbers μ_j , where $j = 1, \dots, q$, such that $g(x^*) = \sum_{i=1}^q \lambda_i y_i - \sum_{j=1}^p \mu_j e_j$ with e_j being the j -th unit vector in $\mathbb{R}^p$. We find that

$$\begin{aligned} h(x^*) &= f(g(x^*)) = f\left(\sum_{i=1}^q \lambda_i y_i - \sum_{j=1}^p \mu_j e_j\right) \leq f\left(\sum_{i=1}^q \lambda_i y_i\right) \leq \max_{i=1, \dots, q} f(y_i) \leq \max_{x \in X_S} f(g(x)) \\ &= \max_{x \in X_S} h(x) \end{aligned}$$

where the first inequality holds due to monotonicity of f on Y' (and $y \in Y \subset Y'$) and the second one due to quasi-convexity on Y' .

If f is strictly quasi-convex, then the second inequality is strict, unless x^* is extreme-supported efficient. ◀

► **Theorem 4.** *Denote by $P_{s,t}(G)$ the set of pairs of edge-disjoint s - t paths in G . Any optimal solution (P_1, P_2) to an instance (G, d, p, s, t) of MSDPP is an extreme-supported solution for the multi-objective optimization problem*

$$\begin{aligned} &\text{maximize} && \begin{pmatrix} \pi(P_1) \\ \pi(P_2) \end{pmatrix} \\ &\text{subject to} && (P_1, P_2)^T \in P_{s,t}(G) \end{aligned} \tag{4}$$

as well as for the multi-objective problem

$$\begin{aligned} &\text{maximize} && \begin{pmatrix} \sum_{e \in P_1} \ln p_e \\ \sum_{e \in P_2} \ln p_e \end{pmatrix} \\ &\text{subject to} && (P_1, P_2)^T \in P_{s,t}(G). \end{aligned} \tag{5}$$

Proof. Note that $P_{s,t}(G)$, the feasible set of both Problem (4) and MSDPP, is finite (and thus compact). To show the first statement, we write the objective function of MSDPP as $h = f \circ g$ with $g: P_{s,t}(G) \rightarrow [0, 1]^2$ with $g(P_1, P_2) = (\pi(P_1), \pi(P_2))^T$ and $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ with $f(a, b) = a + b - a \cdot b$. Since $P_{s,t}(G)$ is finite, the continuity of g trivially follows. For the compact and convex set $Y' = [0, 1]^2 \supseteq g(P_{s,t}(G))$ we now show that f is monotonically increasing in each argument and quasi-convex on Y' .

For $d \in [0, 1]$ with $d > b$ it follows that

$$f(a, b) = a + (1 - a)b < a + (1 - a)d = f(a, d),$$

thus f is strictly monotonically increasing in the second argument. Since f is symmetric, it follows that f is strictly monotonically increasing in both arguments.

To show the quasi-convexity of f , we prove that the level sets $L_\alpha = \{(a, b)^T \in [0, 1]^2 \mid f(a, b) \leq \alpha\}$ are convex for any $\alpha \in [0, 1]$. This definition of quasi-convexity is equivalent to the one given above, see, e.g., [16]. Note that the sets L_α are convex if and only if the sets $L_\alpha - (1, 1)^T$ are convex. These can be rewritten as

$$L_\alpha - (1, 1)^T = \{(a-1, b-1)^T \in [0, 1]^2 \mid (1-a)(1-b) \geq 1-\alpha\} = \{(a, b)^T \in [0, 1]^2 \mid ab \geq 1-\alpha\}$$

We now show convexity of these sets: Consider $\lambda \in [0, 1]$ and $(a, b)^T, (c, d)^T \in [0, 1]^T$ with $ab \geq 1 - \alpha$ and $cd \geq 1 - \alpha$. Due to the inequality of arithmetic and geometric means, $(ad + cb)/2 \geq \sqrt{abcd}$ for non-negative real numbers a, b, c, d (see, e.g., Heuser [22]), it follows that

$$\begin{aligned} (\lambda a + (1 - \lambda)c)(\lambda b + (1 - \lambda)d) &= \lambda^2 \underbrace{ab}_{\geq 1-\alpha} + \lambda(1 - \lambda) \underbrace{(ad + cb)}_{\geq 2\sqrt{abcd} \geq 2(1-\alpha)} + (1 - \lambda)^2 \underbrace{cd}_{\geq 1-\alpha} \\ &\geq (1 - \alpha)(\lambda^2 + 2\lambda(1 - \lambda) + (1 - \lambda)^2) = (1 - \alpha)(\lambda + (1 - \lambda))^2 = 1 - \alpha. \end{aligned}$$

This proves that the sets $L_\alpha - (1, 1)^T$ and thus L_α are convex. Thus, all conditions of Theorem 3 are fulfilled and there exists an extreme-supported solution of Problem (4) that is an optimal solution of MSDPP.

For the second statement, we write the objective function of MSDPP as $h = (f \circ \psi) \circ (\phi \circ g)$ with f, g as defined above, $\psi: [\ln \pi_{\min}, 0]^2 \rightarrow [0, 1]^2$ with $\psi(a, b) \mapsto (e^a, e^b)^T$, and $\phi: [0, 1]^2 \rightarrow [-\infty, 0]^2$ with $\phi(a, b) \mapsto (\ln a, \ln b)^T$ where $\pi_{\min} > 0$ is the probability of the most-uncertain s - t path in G . Thus, $Y' = [\ln \pi_{\min}, 0]^2 \supset (\phi \circ g)(P_{s,t}(G))$ is a convex and compact superset. Since ψ_1, ψ_2 are strictly convex and monotonically increasing functions in Y' and f is strictly quasi-convex in $[0, 1]^2 \supseteq Y'$ and monotonically increasing in each argument, $f \circ \psi$ is monotonically increasing in each argument and strictly quasi-convex in Y' , compare Agrawal and Boyd [2] where this is shown for (not necessarily strict) quasi-convexity, the proof for strict quasi-convexity is analogous. Applying Theorem 3, it now follows that an extreme-supported solution of Problem (5) is an optimal solution to MSDPP. ◀

4.2 Path Iteration

In this section, we propose an algorithm that iterates through the simple s - t paths of a graph $G = (V, E)$ with given $s, t \in V$ in order of decreasing path probability. For each path P_1 , the algorithm computes a second most-certain s - t path P_2 that is edge-disjoint from P_1 , evaluating each solution found with respect to the objective function of MSDPP and keeping track of the best MSDPP solution found. It stops when the probability of the second path found exceeds the probability of the first. We show in Lemma 5 that in this way, the solutions (P_1, P_2) considered by the algorithm constitute a representative set of the efficient solutions from Problem (4) with $\pi(P_1) \geq \pi(P_2)$. Correctness then follows from Theorem 4 since the set of efficient solutions contains the extreme-supported efficient solutions.

The iterator for the simple s - t paths that are ordered by the probability of the paths used in Algorithm 1 is Yen's algorithm [43]. The second path can be found by computing the most-certain path in the graph $(V, E \setminus E(P))$ where P is the path of the current iteration. If

these two paths provide a solution to MSDPP with a higher success probability than the current best solution, the two paths are set as the new current best solution. The pseudocode for our approach is given in Algorithm 1. To compute the second path, we use Fanding's shortest paths faster algorithm [15] (SPFA) with an expected runtime of $\mathcal{O}(|E|)$.

■ **Algorithm 1** Path Iteration Algorithm.

Data: (Directed or undirected) uncertain graph $G = (V, E, p)$ start $s \in V$ and destination $t \in V$

Result: Two edge-disjoint s - t paths that maximize the success probability

Let $P_1^{\text{opt}}, P_2^{\text{opt}}$ be empty paths;

$\tau_{\text{opt}} = 0$;

foreach simple s - t -path P_1 in G ordered by $\pi(P_1)$ **do**

Compute most-certain s - t -path P_2 in $(V, E \setminus E(P_1))$ with SPFA;

if P_2 does not exist **then**

continue;

end

if $\pi(P_1) < \pi(P_2)$ **then**

return $(P_1^{\text{opt}}, P_2^{\text{opt}})$;

end

if $\tau(P_1, P_2) > \tau_{\text{opt}}$ **then**

$\tau_{\text{opt}} \leftarrow \tau(P_1, P_2)$;

$P_1^{\text{opt}}, P_2^{\text{opt}} \leftarrow P_1, P_2$;

end

end

Return $(P_1^{\text{opt}}, P_2^{\text{opt}})$;

► **Lemma 5.** Let (P_1, P_2) be an efficient solution for Problem (4) with $\pi(P_1) \geq \pi(P_2)$. Algorithm 1 considers a solution (P'_1, P'_2) with $\tau(P_1, P_2) = \tau(P'_1, P'_2)$.

Proof. Assume that the algorithm terminates on a solution $(\bar{P}_1, \bar{P}_2)$ before considering the solution (P_1, P_2) . Then $\pi(\bar{P}_2) > \pi(\bar{P}_1) \geq \pi(P_1) \geq \pi(P_2)$. Thus $(\bar{P}_1, \bar{P}_2)$ dominates (P_1, P_2) , which contradicts the assumption that (P_1, P_2) is efficient. ◀

4.3 Integer-Linear-Program-based Dichotomic Search

In our second solution approach, we solve MSDPP by finding a representative set of supported efficient solutions of Problem (5) and comparing their objective values. To this end, we write Problem (5) as a biobjective integer linear program and apply dichotomic search, where in each step we solve a weighted-sum-scalarization of both objective functions. Correctness of this approach is ensured by Theorem 4.

For a directed uncertain graph $G = (V, E^{\rightarrow})$, we use the integer linear programming formulation Problem (6) (with $\lambda_1 \geq \lambda_2 > 0$) for the weighted sum scalarization. Note that since the problem is symmetric, weights with $\lambda_2 > \lambda_1$ do not need to be considered. We denote by $\delta^+(v) = \{(u, w) \in E^{\rightarrow} \mid u = v\}$ the set of edges originating from $v \in V$ and by $\delta^-(v) = \{(u, w) \in E^{\rightarrow} \mid w = v\}$ the set of edges terminating in $v \in V$.

$$\underset{x}{\text{maximize}} \quad \lambda_1 \sum_{e \in E^{\rightarrow}} \ln p_e x_e^1 + \lambda_2 \sum_{e \in E^{\rightarrow}} \ln p_e x_e^2 \quad (6a)$$

subject to

$$\sum_{e \in \delta^+(v)} x_e^i - \sum_{e \in \delta^-(v)} x_e^i = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{else} \end{cases} \quad \forall v \in V \forall i \in \{1, 2\}, \quad (6b)$$

$$\begin{aligned} x_e^1 + x_e^2 &\leq 1 & \forall e \in E^{\rightarrow}, \\ x_e^i &\in \{0, 1\} & \forall e \in E^{\rightarrow} \forall i \in \{1, 2\} \end{aligned} \quad (6c)$$

Each path is encoded by the binary variables x_e^i , where an edge $e \in E^{\rightarrow}$ belongs to path $i \in \{1, 2\}$ if $x_e^i = 1$. The flow constraints (6b) ensure that the paths originate from s , terminate in t , and are actual paths. The constraints (6c) enforce that an edge in $E^{\rightarrow}$ is only taken once over both paths and in either direction. So, the constraints of Problem (6) encode $P_{s,t}(G)$. A solution x of Problem (6) then induces two edge-disjoint s - t paths P_1, P_2 in G with

$$\pi(P_1) = \prod_{e \in E^{\rightarrow}} p_e^{x_e^1} \text{ and } \pi(P_2) = \prod_{e \in E^{\rightarrow}} p_e^{x_e^2}.$$

If the graph G is undirected, exchange $E^{\rightarrow}$ with $E^{\leftrightarrow} = \{(u, v), (v, u) \mid \{u, v\} \in E^{\rightarrow}\}$ - replace the edges in G with two directed edges of opposite direction - and swap (6c) with the constraint

$$x_{(u,v)}^1 + x_{(v,u)}^1 + x_{(u,v)}^2 + x_{(v,u)}^2 \leq 1 \quad \forall \{u, v\} \in E^{\leftrightarrow}.$$

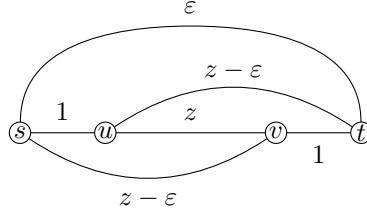
The dichotomic search starts with a lexicographic solution x_1^* and a solution x_2^* of Problem (6) with $\lambda_1 = \lambda_2$. From the paths P_1^1, P_1^2 and P_2^1, P_2^2 induced by x_1^* and x_2^* , compute an optimal solution x^* to Problem (6) with respect to $\lambda_1 = \pi(P_1^2) - \pi(P_2^2)$, $\lambda_2 = \pi(P_2^1) - \pi(P_1^1)$, which describes the slope between $(\pi(P_1^1), \pi(P_2^1))$ and $(\pi(P_1^2), \pi(P_2^2))$. If the objective value of x_1^*, x_2^* for Problem (6) with the new λ_1, λ_2 is smaller than the objective value of x^* for the same problem, then x^* is an extreme supported solution and we repeat this step for (x_1^*, x^*) and (x^*, x_2^*) .

Note that using a similar construction as in the proof of Theorem 2 we can show that solving Problem (6) is NP-hard for weights $\lambda_1 > \lambda_2$ by choosing a sufficiently small $\varepsilon < \min\{\prod_{e \in E} p_e, \sqrt{(1 - \lambda_1 e^{-1})e^{-L}/\lambda_2}\}$, which implies NP-hardness of Problem (6), though the special case $\lambda_1 = \lambda_2$ can be solved by Suurballe's algorithm [39] in polynomial time.

4.4 Approximation by lexicographically optimal solution

In this section and in the following one, we propose two approximation algorithms for the MSDPP. Instead of enumerating a large set of candidate solutions, namely a representative set of (extreme-supported) efficient solutions for Problem (4) or Problem (5), the underlying idea of our approximation algorithms is to find *a specific* extreme-supported efficient solution.

A *lexicographically optimal* solution to a bi-objective optimization problem is one that optimizes the second objective function among all solutions that maximize the first objective function. (In general, the role of "first" and "second" objective function may also be reversed. However, for our optimization problem, this is irrelevant, because the function f mapping the objective vector of the biobjective problem to the objective value of MSDPP is symmetric.) Note, however, that finding a lexicographically optimal solution to Problem (5) corresponds to solving the (2D1SPP) which is NP-hard [13], and this result can be easily transferred to Problem (4).



■ **Figure 1** Example graph that shows the tightness of the approximation guaranteed of Theorem 6.

Despite this theoretical complexity result, we argue that computing *just one* of the potentially exponentially many extreme-supported solutions of Problem (4) is considerably faster than computing a representative set of *all* extreme-supported efficient or even of all efficient solutions, as we do in the approaches proposed in Sections 4.2 and 4.3, respectively.

To compute a lexicographically optimal solution, we can modify the approach from Section 4.2, stopping as soon as the first feasible solution is found or apply integer programming, i.e., the first step of the dichotomic search from Section 4.3. We also refer to computing the lexicographically optimal solution as the *lexicographic* approach.

► **Theorem 6.** *Computing a lexicographically optimal solution of Problem (4) gives a tight approximation guarantee of $1/2$ for MSDPP.*

Proof. Let $(P_1^{\text{opt}}, P_2^{\text{opt}})$ be an optimal solution to MSDPP with $\pi(P_1^{\text{opt}}) \geq \pi(P_2^{\text{opt}})$ and let (P_1, P_2) be a lexicographically optimal solution to Problem (4). Then $\pi(P_1) \geq \pi(P_1^{\text{opt}})$ and $\pi(P_1) \geq \pi(P_2^{\text{opt}})$. It follows that

$$\begin{aligned} \frac{\tau(P_1, P_2)}{\tau(P_1^{\text{opt}}, P_2^{\text{opt}})} &= \frac{\pi(P_1) + (1 - \pi(P_1))\pi(P_2)}{\pi(P_1^{\text{opt}}) + \pi(P_2^{\text{opt}}) - \pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})} \geq \frac{\pi(P_1)}{\pi(P_1^{\text{opt}}) + \pi(P_2^{\text{opt}})} \\ &\geq \frac{\pi(P_1)}{2\pi(P_1)} = \frac{1}{2}. \end{aligned}$$

For the tightness consider the graph in Figure 1. The optimal solution with a success probability of $2(z - \varepsilon) - (z - \varepsilon)^2$ are the paths $s-v-t$ and $s-u-t$. The lexicographically optimal solution with the paths $s-u-v-t$ and $s-t$ has a success probability of $z + \varepsilon - z\varepsilon$. Taking the limit it follows that

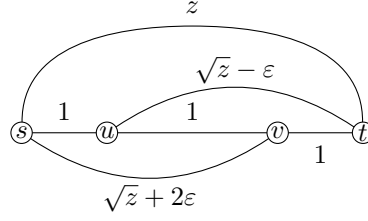
$$\lim_{z \rightarrow 0} \lim_{\varepsilon \rightarrow 0} \frac{z + \varepsilon - z\varepsilon}{2(z - \varepsilon) - (z - \varepsilon)^2} = \frac{1}{2}. \quad \blacktriangleleft$$

4.5 Suurballe's Algorithm based Heuristic

While the approach described in the previous section produces solutions consisting of paths of very different probabilities, we now investigate the worst-case performance of taking the solution (P_1, P_2) that minimizes Problem (6) with equal weights $\lambda_1 = \lambda_2 = 1$. This is *equivalent* to finding two edge-disjoint $s-t$ paths that maximize the product of the paths' probabilities. Analogously to the approach sketched in [10] for the *most-certain path* problem, one can show that this problem is equivalent to finding a solution that minimizes the sum of negative log-probabilities. This problem can be solved in polynomial-time with Suurballe's algorithm [39].

Unlike the approach described in Section 4.4, we cannot give a general bound on the approximation factor of the solution found by this approach. This is illustrated by the instance sketched in Figure 2, where the approximation factor approaches

$$\lim_{z \rightarrow 0} \lim_{\varepsilon \rightarrow 0} 2\sqrt{z} + \varepsilon - z - \varepsilon\sqrt{z} = 0.$$



■ **Figure 2** For $\varepsilon > 0$ and $z \in (0, 1 - 2\varepsilon + \varepsilon^2)$, Suurballe's algorithm gives the paths $s-u-t$ and $s-v-t$ with a success probability of $2\sqrt{z} + \varepsilon - z - \varepsilon\sqrt{z}$ and a multiplied path probability of $z + \varepsilon\sqrt{z}$. The optimal solution with a success probability 1 is given by $s-u-v-t$ and $s-t$ and has a multiplied path probability of z .

Note that in the constructed example, the probabilities of the two paths $(P_1^{\text{opt}}, P_2^{\text{opt}})$ in the optimal solution are very different, $\pi(P_1^{\text{opt}}) = 1$ while $\pi(P_2^{\text{opt}})$ approaches 0. On the other hand, intuitively, Suurballe's algorithm should provide good results for instances where the probabilities of the two paths $(P_1^{\text{opt}}, P_2^{\text{opt}})$ are (almost) equal. In Theorem 7 we see that indeed, the approximation guarantee can be bounded depending on (an upper bound on) the probability of the more certain path and (a lower bound on) the probability of the less certain path.

► **Theorem 7.** *Let $(P_1^{\text{opt}}, P_2^{\text{opt}})$ be an optimal solution to MSDPP with $\pi(P_1^{\text{opt}}) \geq \pi(P_2^{\text{opt}})$ and let $\alpha, \beta \in [0, 1]$ with $\beta \geq \pi(P_1) \geq \pi(P_2) \geq \alpha$. Then, for any solution (P_1, P_2) that is optimal for Problem (5) with $\lambda_1 = \lambda_2 = 1$ we have that*

$$\frac{\tau(P_1, P_2)}{\tau(P_1^{\text{opt}}, P_2^{\text{opt}})} \geq \frac{2\sqrt{\alpha\beta} - \alpha\beta}{\alpha + \beta - \alpha\beta}.$$

Proof. W.l.o.g. assume that $\pi(P_1) \geq \pi(P_2)$ and note that by definition $\pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}}) \leq \pi(P_1)\pi(P_2)$ and $\tau(P_1^{\text{opt}}, P_2^{\text{opt}}) \geq \tau(P_1, P_2)$. This implies that $\pi(P_1^{\text{opt}}) \geq \pi(P_1) \geq \pi(P_2) \geq \pi(P_2^{\text{opt}})$.

Using the fact that the function $f(a, b) = a + b - a \cdot b$ attains its minimum on $\{(a, b)^T \in [0, 1]^2 : ab \geq \pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})\}$ at $a = b = \sqrt{\pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})}$ (see Lemma 8 in the appendix), we conclude that $\tau(P_1, P_2) \geq \sqrt{\pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})} - \pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})$.

$$\begin{aligned} \frac{\tau(P_1, P_2)}{\tau(P_1^{\text{opt}}, P_2^{\text{opt}})} &\geq \frac{2\sqrt{\pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})} - \pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})}{\pi(P_1^{\text{opt}}) + \pi(P_2^{\text{opt}}) - \pi(P_1^{\text{opt}})\pi(P_2^{\text{opt}})} \\ &\stackrel{\text{Lemma 10}}{\geq} \frac{2\sqrt{\beta\pi(P_2^{\text{opt}})} - \beta\pi(P_2^{\text{opt}})}{\beta + \pi(P_2^{\text{opt}}) - \beta\pi(P_2^{\text{opt}})} \stackrel{\text{Lemma 11}}{\geq} \frac{2\sqrt{\alpha\beta} - \alpha\beta}{\alpha + \beta - \alpha\beta} \end{aligned}$$

This gives the approximation factor. ◀

5 Experimental Results

After we describe the benchmark instances and the implementation of the presented algorithms in Section 5.1, we compare the average runtime of the algorithms in Section 5.2. Lastly, we compare the success probabilities of the approximation algorithms against those of the exact algorithms in Section 5.3.

5.1 Benchmark Instances and Implementation Details

We consider four classes of benchmark graphs: GRID, GRIDLOW, GRIDHIGH, and COASTAL. The base class GRID consists of grid-graphs with 5 columns and $2k$ rows with $k \in \mathbb{N}$ and edge probabilities are uniformly sampled from $[0, 1]$. The classes GRIDLOW and GRIDHIGH are identical to GRID but the edge probabilities are uniformly sampled from $[0.1, 0.35]$ and $[0.75, 1]$, respectively. Graphs from the COASTAL class are metric planar graphs designed to resemble networks in coastal regions. A detailed description of this graph class is provided in [31]. Each graph has $5 \cdot 2k$ nodes and up to $3(5 \cdot 2k)$ edges. The probability of an edge in a COASTAL graph is given by $p \cdot (\ell_{\max} - \ell_e) / (\ell_{\max} - \ell_{\min})$ where p is uniformly sampled from $[0, 1]$ for each edge, ℓ_e is the length of the edge e , and $\ell_{\min}, \ell_{\max}$ are the lengths of the shortest and longest edge of the graph instance, respectively. For each graph class and $k \in \{2, 4, \dots, 100\}$, 10 different graphs were generated. Standalone implementations of the graph generation process are available at [30]. Due to the inefficiency of our COASTALgraph generation process, we limit the number of nodes for these instance to 300. For each graph, we choose a vertex with the lowest x -coordinate, and, if there are several with the same x -coordinate, then take the one with the lowest y -coordinate as start vertex s . To set an end vertex t we perform a breadth-first search in the graph starting from s and take the last vertex found. To ensure that there are two edge-disjoint s - t paths, a direct s - t edge with the smallest probability greater than zero that a 64-bit float value can represent is added.

The algorithms described in the paper are implemented in Rust [40] and are provided in [29]. To improve the runtime and the memory usage of Algorithm 1, only the paths whose probability could improve the current best found solution are kept in the internal heap data structure. For the approximation algorithm described in Section 4.4, we solve Problem (6) with $\lambda_1 = 1$ and a sufficiently small λ_2 since this proved to be considerably faster than applying a modified version of the path iteration algorithm, as proposed in Section 4.4. Problem (6) is solved using Gurobi 13.0.2 [20]. For Suurballe’s algorithm, we used Fanding’s SPFA [15] with an expected $\mathcal{O}(|E|)$ runtime instead of Dijkstra’s algorithm.

For each instance and algorithm, we set a time limit of 60 seconds. The computational experiments were executed on 8 threads of an AMD Epic 9654 with 32GB of RAM on Debian 13.

5.2 Runtime Analysis

■ **Table 2** Number of solved instances by the path iteration algorithms from Section 4.2 per graph class.

	GRID	GRIDHIGH	GRIDLOW	COASTAL
Solved	183 / 500	117 / 500	492 / 500	116 / 150

The path iteration could not solve all considered instance within the time limit expect for the GRIDLOW graphs as we can see in Table 2. Here, we can see that GRIDLOW instances can be solved more often than GRID, GRIDLOW and COASTAL instances. We suspect that comparatively low edge probabilities make it easier for the implemented approach to follow high probability paths.

Furthermore, only for the GRIDLOW instances the path iteration algorithm is faster for graphs with less than 600 nodes than the ILP-based dichotomic search; see Figure 3. For the other graph classes, around 180 nodes the path iteration becomes slower and begins to reach

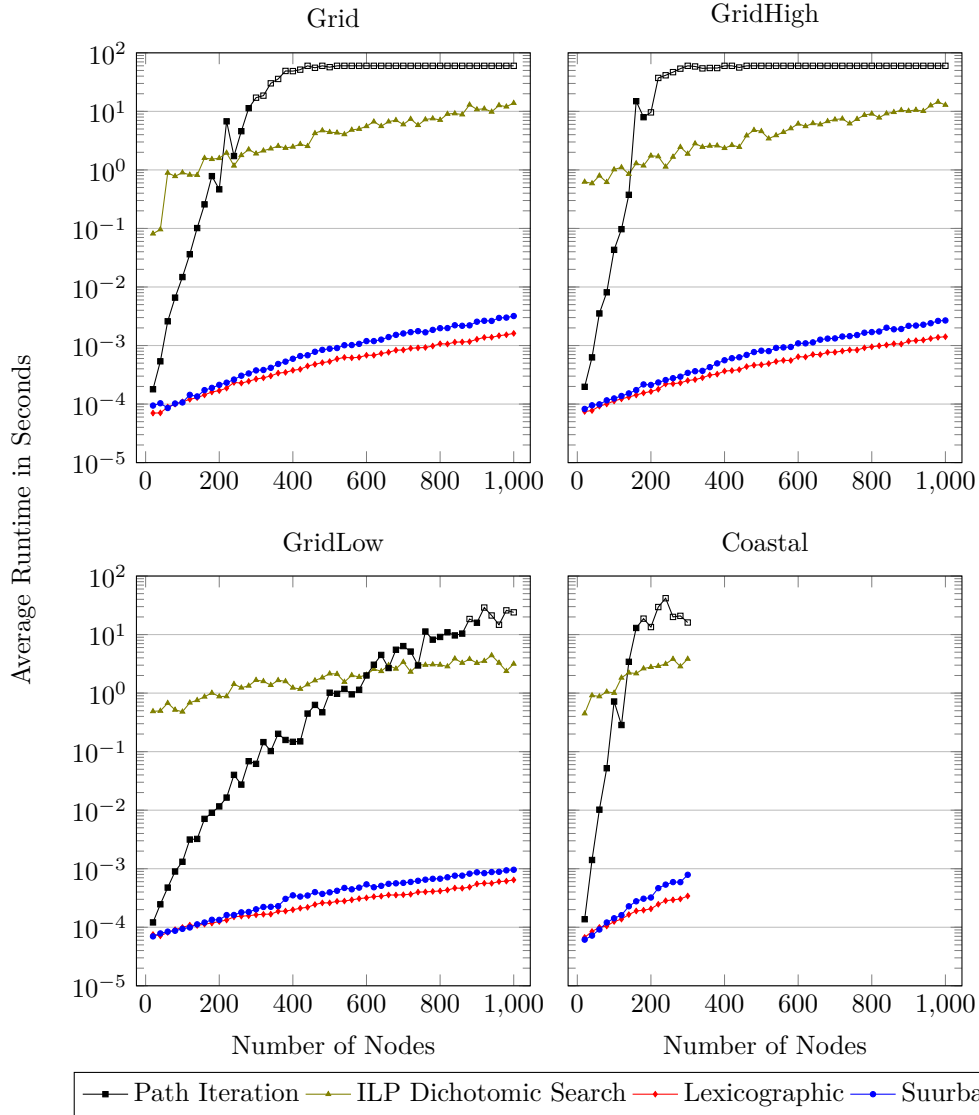


Figure 3 Average runtime of the algorithms over the graph classes. Unfilled markers indicate that not all instances were solved to optimality within the time limit.

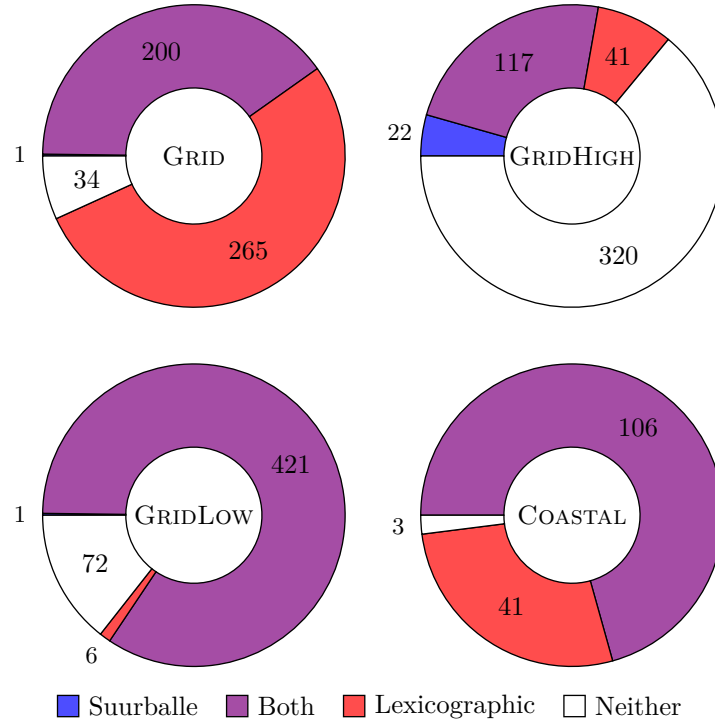
the time limit. Both the approximation algorithms show near identical runtimes. Even if the lexicographic algorithm solves 2D1SPP, which is NP-hard, the low runtime of the path iteration based lexicographic algorithm implies that the first considered most-certain s - t path allows for a second edge-disjoint s - t path. Thus, the lexicographic algorithm solves as many most-certain paths problems as Suurballe's algorithm, which has a higher computational overhead.

5.3 Heuristics' Quality Analysis

For the analysis of the solution quality of the approximation algorithm and the heuristic, consider Table 3 and Figure 4. In the first row of Table 3, we see for how many of the instances the approximation approaches found an optimal solution. In the second to fourth rows we see average, the 5th percentile - the best of the worst 5% -, and worst observed value of the approximation ratio ALG/OPT rounded to two decimal places.

■ **Table 3** Comparison of the solution quality of the Suurballe (Suur.) and lexicographic (Lexico.) approximation algorithms.

	GRID		GRIDHIGH		GRIDLOW		COASTAL	
	Suur.	Lexico.	Suur.	Lexico.	Suur.	Lexico.	Suur.	Lexico.
Found optimum	201	456	139	158	422	427	106	147
Average	0.73	1.00	0.96	0.98	0.98	0.99	0.93	1.00
5th percentile	0.16	1.00	0.84	0.92	0.93	0.93	0.51	1.00
Worst observed	0.02	0.78	0.67	0.87	0.20	0.58	0.06	0.90



■ **Figure 4** Number of optimally solved instances per graph class for the approximation approaches.

The pie charts in Figure 4 show the amount of instances for which each and both approximation approaches found an optimal solution.

For both approaches, we remark that they found the optimal solutions in remarkably many cases. As Figure 4 confirms, for a considerable proportion of the instances, the lexicographic and Suurballe's solution are identical and optimal. This also shows in the average approximation ratios in Table 3, which are almost everywhere close to 1.00.

Overall, the lexicographic algorithm is better than Suurballe's algorithm for all our considered metrics, especially in the 5th percentile and worst observed for GRID and COASTAL.

6 Conclusions and future research

In this paper, we studied the NP-hard problem MSDPP, i.e., the problem to find two edge-disjoint paths between a start and an end node in a graph equipped with edge traversal properties that maximize the success probability. We have shown that the problem is NP-hard and presented four solution algorithms, which are all motivated by the property that optimal

solutions to the MSDPP are extreme supported efficient solutions to a suitably defined bi-objective optimization problem. Regarding exact approaches, we find that the ILP-based dichotomic search approach outperforms the *path iteration* algorithm for larger graphs. When it comes to the approximation algorithm and the heuristic, the lexicographic approximation algorithms proved better than Suurballe's algorithm. It would be interesting to see whether the runtime of the ILP-based dichotomic search approach could be improved by warm-starting it with the result of one of the approximation algorithms. Regarding the hardness results and the proposed algorithms, it would be interesting to investigate to what extent results extend to the case of finding two vertex-disjoint maximum-success paths.

Where this paper focused purely on algorithmic problem aspects, in future research we plan to focus on solving real-world instances, where edge probabilities are derived, e.g. from air surveillance data. It would be interesting to see whether our algorithms perform well on these instances (which are expected to differ considerably from our current test instances) as well, and how the results we obtain compare to the results of other routing algorithms designed for such applications, or, in general routing under uncertainty.

References

- 1 Birce Adsanver, Burcu Balcik, V. Bélanger, and M.-E. Rancourt. Operations research approaches for improving coordination, cooperation, and collaboration in humanitarian relief chains: A framework and literature review. *European Journal of Operational Research*, 319(2):384–398, 2024. doi:10.1016/j.ejor.2023.11.031.
- 2 Akshay Agrawal and Stephen Boyd. Disciplined quasiconvex programming. *Optimization Letters*, 14:1643–1657, 2020. doi:10.1007/s11590-020-01561-8.
- 3 Ana Maria Anaya-Arenas, Jacques Renaud, and Angel Ruiz. Relief distribution networks: a systematic review. *Annals of Operations Research*, 223:53–79, 2014. doi:10.1007/s10479-014-1581-y.
- 4 Amotz Bar-Noy and Baruch Schieber. The Canadian Traveller Problem. In *SODA*, volume 91, pages 261–270, 1991. doi:10.5555/127787.127835.
- 5 Kristóf Bérczi and Yusuke Kobayashi. The Directed Disjoint Shortest Paths Problem. In *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, 2017. doi:10.4230/LIPIcs.ESA.2017.13.
- 6 Ramesh Bhandari. Optimal Physical Diversity Algorithms and Survivable Networks. In *Proceedings of the Second IEEE Symposium on Computers and Communications (ISCC 1997)*, pages 433–441, 1997. doi:10.1109/ISCC.1997.616037.
- 7 Andreas Björklund and Thore Husfeldt. Shortest Two Disjoint Paths in Polynomial Time. In *Automata, Languages, and Programming (ICALP)*, pages 211–222, 2014. doi:10.1007/978-3-662-43948-7_18.
- 8 Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012. doi:10.1002/net.20487.
- 9 André Chassein, Trivikram Dokka, and Marc Goerigk. Algorithms and uncertainty sets for data-driven robust shortest path problems. *European Journal of Operational Research*, 274(2):671–686, 2019. doi:10.1016/j.ejor.2018.10.006.
- 10 Nicos Christofides. *Graph theory: An algorithmic approach*. Academic Press, Inc., 1975.
- 11 Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.
- 12 Matthias Ehrgott. *Multicriteria Optimization*. Springer, 2005. doi:10.1007/3-540-27659-9_9.
- 13 Tali Eilam-Tzoref. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998. doi:10.1016/S0166-218X(97)00121-2.

- 14 Shimon Even, Alon Itai, and Adi Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal on Computing*, 5(4):691–703, 1976. doi:10.1137/0205048.
- 15 Duan Fanding. A faster algorithm for shortest-path-SPFA. *Journal of Southwest Jiaotong University*, 2(9):2, 1994.
- 16 Werner Fenchel. *Convex cones, sets, and functions*. Lecture Notes, Princeton University, Department of Mathematics, 1953.
- 17 Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980. doi:10.1016/0304-3975(80)90009-2.
- 18 Daniel Golovin, Vineet Goyal, Valentin Polishchuk, R. Ravi, and Mikko Sysikaski. Improved approximations for two-stage min-cut and shortest path problems under uncertainty. *Mathematical Programming*, 149(1):167–194, 2015. doi:10.1007/s10107-013-0742-0.
- 19 Marinus Gottschau, Marcus Kaiser, and Clara Waldmann. The undirected two disjoint shortest paths problem. *Operations Research Letters*, 47(1):70–75, 2019. doi:10.1016/J.ORL.2018.11.011.
- 20 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL: <https://www.gurobi.com>.
- 21 Damjan Hatic, Markus Rauhut, Vladyslav Polushko, Marco Codastefano, Sophia Rosa, Patrick McKay, Francesco Stompanato, Antonio Belezza, and Hans Hagen. Disaster impact analysis in Africa: EDDA-UAV building damage dataset of rural and urban Mozambique. In *Earth Resources and Environmental Remote Sensing/GIS Applications XIV*, volume 12734, pages 153–166, 2023. doi:10.1117/12.2683882.
- 22 Harro Heuser. *Lehrbuch der Analysis*. Springer-Verlag, 2013.
- 23 Dan A. Iancu and Nikolaos Trichakis. Pareto Efficiency in Robust Optimization. *Management Science*, 60(1):130–147, 2013. doi:10.1287/mnsc.2013.1753.
- 24 Aubin Jarry and Stéphane Pérennes. Disjoint paths in symmetric digraphs. *Discrete Applied Mathematics*, 157(1):90–97, 2009. doi:10.1016/j.dam.2008.04.024.
- 25 Richard M Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, pages 85–103, 1972. doi:10.1007/978-3-540-68279-0_8.
- 26 Yusuke Kobayashi and Ryo Sako. Two disjoint shortest paths problem with non-negative edge length. *Operations Research Letters*, 47(1):66–69, 2019. doi:10.1016/J.ORL.2018.11.012.
- 27 Tanmoy Kundu, Jiah-Biing Sheu, and Hsin-Tsz Kuo. Emergency logistics management—Review and propositions for future research. *Transportation research part E: logistics and transportation review*, 164:102789, 2022. doi:10.1016/j.tre.2022.102789.
- 28 Roberto Montemanni and Luca Maria Gambardella. The robust shortest path problem with interval data via Benders decomposition. *4OR*, 3(4):315–328, 2005. doi:10.1007/s10288-005-0066-x.
- 29 Aaron Neugebauer. Finding Maximum-Success Disjoint Paths, 2026. Version 1.0.0. URL: <https://github.com/PantomInach/uncertain-road-network-traversal-strategies>.
- 30 Aaron Neugebauer. Gengraph: A Graph Generation Library, 2026. Version 1.2.0. URL: <https://github.com/PantomInach/gengraph>.
- 31 Aaron Neugebauer, Alena Otto, and Marie Schmidt. The relief distribution problem with trucks and drones under incomplete demand information. *arXiv*, 2026. doi:10.48550/arXiv.2603.23135.
- 32 Yu Marco Nie and Xing Wu. Shortest path problem considering on-time arrival probability. *Transportation Research Part B: Methodological*, 43(6):597–613, 2009. doi:10.1016/j.trb.2009.01.008.
- 33 Evdokia Nikolova. High-Performance Heuristics for Optimization in Stochastic Traffic Engineering Problems. In *Large-Scale Scientific Computing*, pages 352–360, 2009. doi:10.1007/978-3-642-12535-5_41.

- 34 Evdokia Nikolova. Approximation Algorithms for Reliable Stochastic Combinatorial Optimization. In *Approximation, Randomization, and Combinatorial Optimization*, pages 338–351. Springer, 2010. doi:10.1007/978-3-642-15369-3_26.
- 35 Evdokia Nikolova and David R. Karger. Route Planning under Uncertainty: The Canadian Traveller Problem. In *AAAI Conferece on Artificial Intelligence*, volume 23, pages 969–974, 2008. doi:10.5555/1620163.1620223.
- 36 Evdokia Nikolova, Jonathan A. Kelner, Matthew Brand, and Michael Mitzenmacher. Stochastic Shortest Paths Via Quasi-convex Maximization. In *Algorithms - ESA*, pages 552–563, 2006. doi:10.1007/11841036_50.
- 37 Vladyslav Polushko, Alexander Jenal, Jens Bongartz, Immanuel Weber, Damjan Hatic, Ronald Rösch, Thomas März, Markus Rauhut, and Andreas Weinmann. Blessemflood21: Advancing Flood Analysis with a High-Resolution Georeferenced Dataset for Humanitarian Aid Support. In *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*, pages 368–373, 2024. doi:10.1109/IGARSS53475.2024.10642128.
- 38 Neil Robertson and Paul D. Seymour. Graph Minors .XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/JCTB.1995.1006.
- 39 John W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974. doi:10.1002/NET.3230040204.
- 40 The Rust Team. The Rust Programming Language, 2020. URL: <https://rust-lang.org>.
- 41 Torsten Tholey. Solving the 2-Disjoint Paths Problem in Nearly Linear Time. *Theory of Computing Systems*, 39(1):51–78, 2006. doi:10.1007/S00224-005-1256-9.
- 42 Stephan Westphal. A note on the k-Canadian Traveller Problem. *Information Processing Letters*, 106(3):87–89, 2008. doi:10.1016/j.ipl.2007.10.004.
- 43 Jin Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 27(4):526–530, 1970. doi:10.1090/qam/253822.

A

 Appendix

► **Lemma 8.** Let $M \in (0, 1]$. The function $f(a, b) = a + b - a \cdot b$ takes its minimum on $\{(a, b)^T \in [0, 1]^2 : ab \geq M\}$ at $a = b = \sqrt{M}$.

Proof. Since $f(a, b)$ is strictly monotonously increasing in a and b , the minimum can only occur if $ab = M$. Furthermore, with $\tilde{f}(b) := f(\frac{M}{b}, b)$ we have

$$\frac{\partial \tilde{f}}{\partial b}(b) = 1 - \frac{M}{b^2}.$$

Thus, $b = \sqrt{M}$ is the only relevant critical point and

$$\frac{\partial^2 \tilde{f}}{\partial b^2}(\sqrt{M}) = \frac{2}{\sqrt{M}} > 0,$$

therefore this is indeed a minimum. ◀

► **Lemma 9.** For $1 \geq a \geq b > 0$, it holds that $a + b\sqrt{ab} \geq ab + b$.

Proof. Let $a = bk$ with $k \geq 1$.

$$a + b\sqrt{ab} = bk + b\sqrt{b^2k} = b(\underbrace{k}_{\geq 1} + bk) + b^2(\underbrace{\sqrt{k} - k}_{\leq 0}) > b(bk + 1) = b^2k + b = ab + b. \quad \blacktriangleleft$$

8:18 Finding Maximum-Success Disjoint Paths

► **Lemma 10.** For $b \in (0, 1)$, the function f defined by

$$f(a) = \frac{2\sqrt{ab} - ab}{a + b - ab}$$

is monotonously decreasing on $[0, b]$.

Proof.

$$\frac{\partial}{\partial a} f(a) = \frac{b(b - a + ab - b\sqrt{ab})}{\sqrt{ab}(a + b - ab)^2}$$

The function is monotonously decreasing if $\frac{\partial}{\partial a} f(a) \leq 0$ which is equivalent to $b - a + ab - b\sqrt{ab} \leq 0$. This inequality is true due to Lemma 9. ◀

► **Lemma 11.** For $a \in (0, 1]$, the function f defined by

$$f(a) = \frac{2\sqrt{ab} - ab}{a + b - ab}$$

is monotonously decreasing on $[a, 1]$.

Proof. Analogous to the proof of Lemma 10. ◀

Scheduling Electric-Vehicle Charging Under Grid Capacity and Minimum Charge Rates

Bruna Cavalcanti Lauro 

Department of Advanced Computing Sciences, Maastricht University, The Netherlands

Matúš Mihalák 

Department of Advanced Computing Sciences, Maastricht University, The Netherlands

Filip Schlembach 

Department of Advanced Computing Sciences, Maastricht University, The Netherlands

Evgueni Smirnov 

Department of Advanced Computing Sciences, Maastricht University, The Netherlands

Abstract

Electric vehicle adoption is growing rapidly, and parking facilities face increasing pressure to expand their charging infrastructure. This paper studies the power allocation scheduling problem that arises when a parking facility with fixed grid-capacity limit must distribute power to connected electric vehicles, under given lower and upper bounds on the charge rate at any given time. We focus on the optimization goal of maximizing the number of vehicles that get charged a given target of power, with the secondary goal of providing as much power as possible. We show that the problem is NP-hard. We formulate the problem as a mixed-integer linear program (MILP) that establishes a theoretical upper bound on performance of any online algorithm, which is the main goal of the real-world problem. We design and compare a few online scheduling strategies against the offline bound, using a real-world instance of vehicles and testing across 6 to 40 chargers. We use MILP as an online strategy, classical earliest-deadline first greedy algorithm, and we also design data-driven approaches using machine learning techniques. Results show that the simplest static load balancing approach cannot compete with the best performing online approach, and that the real-world problem can benefit from dynamic load balancing strategies such as those designed and evaluated in this paper.

2012 ACM Subject Classification Applied computing → Operations research; Computing methodologies → Machine learning

Keywords and phrases Electric-Vehicle Charging, Scheduling, Grid-Capacity Constraints, Charge-Rate Lower-Bounds

Digital Object Identifier 10.4230/OASIcs.ATMOS.2026.9

Acknowledgements We thank the anonymous reviewers for very thorough and inspiring comments, which helped to improve the paper.

1 Introduction

The adoption of electric vehicles (EVs) is growing, bringing new challenges and demands on the existing automotive infrastructure, which was tailored and optimized to be used primarily by combustion-engine vehicles. In particular, the relatively long charging times and the limited availability of electrical power provides several scheduling and coordination problems to be solved.



© Bruna Cavalcanti Lauro, Matúš Mihalák, Filip Schlembach, and Evgueni Smirnov; licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 9; pp. 9:1–9:17



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

One such problem arises in parking facilities, where customers demand more parking spots with charging possibilities of electric vehicles. Yet, most of parking facilities operate under a fixed grid capacity limit that caps the total power available at any moment.¹ As a result, the straightforward charging approach of “split the total available power equally among the connected EVs to the charging facility” is no longer feasible, since the IEC-61851 standard [9] mandates a minimum power threshold of 4.1 kW for full-battery EVs (BEVs) and 1.4 kW for plug-in hybrid EVs (PHEVs): as more vehicles connect simultaneously, the equal share per vehicle decreases; if the equal share falls below a vehicle’s threshold, its internal battery management system stops charging and transitions to an idle state. Consequently, beyond a certain number of concurrent vehicles, the entire facility stalls and delivers zero power until enough vehicles depart. Moreover, this straightforward approach, also called the *static load balancing* (SLB), does not take into account information such as the status of the battery charge, which could be used to further optimize, for example, the total amount of delivered energy or the number of cars charged close to full battery capacity.

In this paper, motivated by our real-world contacts with a parking infrastructure operator, we study the electric-vehicle charging scheduling problem with a power (grid-capacity) constraint and a minimum and maximum charging threshold for vehicles. Our research is guided by the following real-world scenario: a parking garage is currently equipped with 6 parking spots with charging facilities; at any moment, the facility can charge the fleet with a total of at most 50 kW of electric power; the facility uses static load balancing. The main question is whether dynamic load balancing, i.e., a centrally planned scheduling that continuously allocates available power across connected EVs in a, possibly, non-uniform way, can allow to extend the number of parking spots while maintaining a satisfactory level of service. Motivated by these real-world optimization concerns, we study the resulting offline and online scheduling problem defined in the following.

1.1 Terminology, Notation, and Problem Definition

A set of n electric vehicles $[n] := \{1, 2, \dots, n\}$ arrives at the parking facility over a fixed scheduling horizon $[0, T]$ partitioned into one-minute intervals $t \in \{1, 2, \dots, T\} =: [T]$. Each vehicle $i \in [n]$ is characterized by its arrival time $t_{\text{arr},i}$, its departure time $t_{\text{dep},i}$, its total requested energy $E_{\text{req},i}$ (in kW hours), its maximum power limit $P_{\text{max},i}$ (in kW), and its minimum power limit $P_{\text{min},i}$ (in kW). The facility can provide at any time a total power of at most P_{grid} (in kW). All of this information is an input to our problem, including the departure times (which the operator of the parking facility asks from the drivers at the entrance to the parking facility).

A *schedule* is an assignment of power to vehicles over time, given by values $S := \{p_{i,t}\}_{i \in [n]; t \in [T]}$, with the meaning that vehicle i is given power $p_{i,t}$ during time-interval t . A *feasible* schedule is a schedule $S = \{p_{i,t}\}_{i \in [n]; t \in [T]}$ such that (i) for any time interval $t \in T$, the total amount of energy assigned to all vehicles at time t is at most P_{grid} , (ii) no vehicle $i \in [n]$ is assigned a positive power at time $t \notin [t_{\text{arr},i}, t_{\text{dep},i}]$, (iii) every vehicle i is either assigned zero power, or power that is at least its lower-bound $P_{\text{min},i}$ and at most its upper-bound $P_{\text{max},i}$, and (iv) no vehicle is assigned, in total, more than the requested amount of power $E_{\text{req},i}$.

¹ Typical grid capacity of an average parking garage reported by our industrial partner is between 50kW and 150kW.

The goal is to find a feasible schedule that maximizes the following hierarchical objective: first, it maximizes the number of vehicles that get charged at least α -fraction of their requested energy, and second, the total energy delivered to the vehicles. The primary optimization goal is thus the customer satisfaction (number of satisfied customers), and the secondary optimization goal is both the customer satisfaction and the company profit (the total amount of delivered/sold energy). We adopt the jargon of the industry, and call the goal of charging the α -fraction of the requested energy the *service level agreement* target, or, for short, the SLA target. Throughout the paper, we maintain $\alpha = 0.9$ (or, 90%). We chose the value of 90%, because it is the largest value of α for which the charging of a lithium-ion battery is approximately linear in delivered energy [12]. None of the algorithmic approaches presented in this paper depends on the concrete choice of the value of α . The modeling assumes that the charging is approximately linear in delivered energy, and for that, any value of $\alpha \leq 0.9$ is fine.²

We call the interval $[t_{\text{arr},i}, t_{\text{dep},i}]$ the *connection window* of vehicle $i \in [n]$. For a time $t \in [T]$ within the connection window of vehicle $i \in [n]$, i.e., for $t_{\text{arr},i} \leq t \leq t_{\text{dep},i}$, vehicle i is *active* at time t with respect to a feasible schedule S if $p_{i,t} > 0$, and otherwise it is *inactive*.

1.2 Our Contributions

We first consider the offline problem, when the entire input is given to the algorithm at the beginning. We show that the problem is NP-hard, even when every vehicle has the same arrival time equal to zero, and every vehicle has the same (integer) departure time t_{dep} of value $\Theta(n)$ (i.e., linear in n). To address the real-world problem, we formulate the offline scheduling problem as a mixed integer linear program (MILP) and use it to compute optimum schedules to obtain a reference point for comparison to online approaches, and for comparison with the static load balancing approach.

We then consider the online problem, in which the input information about the vehicles is given to the algorithm upon the arrival times of the vehicles, i.e., at any time $t \in [T]$, the algorithm knows the information about all vehicles $i \in [n]$ with arrival time $t_{\text{arr},i} \leq t$. For the online problem, we design and experimentally evaluate several online scheduling algorithms that need to make decisions at every time point $t = 1, 2, \dots, T$ using only the information about vehicles whose arrival time is at or before t . The motivation for knowing the (future) departure time comes from the operation of the parking facility which asks every vehicle, upon arrival, for an (estimated) departure time and also for the requested energy power.

Some of our online strategies use data-driven approaches, in which the online algorithms can use, beyond the current state of the facility, also historical data. We use machine learning techniques to predict future arrival of electric vehicles, their requested power, and their departure times, and evaluate an online algorithm that at any moment computes an offline schedule on the current vehicles plus the predicted vehicles for a short future time window. The motivation for this approach is to find online strategies that possibly approach the offline optimum under realistic operational conditions.

² Albeit, much of the literature takes a more conservative approach and sets the threshold of $\alpha = 0.8$ for approximately-linear charging behavior.

1.3 Related Work

There are several related concepts in the field of machine scheduling. In machine scheduling, in general, there are n jobs having processing times, and possibly release dates, due dates, or weights, and there are m machines that can process the jobs, while, at any time, working on at most one job. A preemptive schedule is a schedule that allows a machine to process a job over several time intervals, whose total length is equal to the job's processing time.

Chen et al. [3] studies an online deadline scheduling problem with an on-arrival commitment requirement on one machine, where a scheduler must immediately accept or reject a job upon arrival. Their optimization goal is to maximize the total load of scheduled processing minus the remaining processing time of uncompleted jobs. Beside the differences in the optimization goal and the number of machines, observe also that in our problem, we cannot accept or reject any vehicles (since vehicles find autonomously any free parking spot with a charging facility). Furthermore, our problem requires allocating a shared continuous power capacity across a simultaneous fleet of connected vehicles at every minute, rather than managing a sequential single-job processor queue.

Alinia et al. [1] study an online scheduling problem for deadline-constrained EV charging, maximizing the total weighted delivered energy. Their problem permits power allocations up to a vehicle-specific maximum and having a facility grid capacity, which mirrors our work. However, their optimization goal is different, and they do not enforce a minimum power threshold as we do.

The PhD thesis of Zaidi [17] provides a classification of *electric-vehicle charging-scheduling problem* with respect to various optimization objectives (from the power-grid operators as well as from the charging-facility point of view), various constraints and considerations related to vehicles (e.g., arrival and departure times, charging limits) or infrastructure (electricity prices or charging capacity), and various solution approaches (exact offline algorithms, heuristics, or online algorithms). Besides the overview, the PhD thesis also provides NP-hardness of several fundamental variants of EV charging scheduling problems: (i) computing a minimum capacity $P_{\max}$ of the grid to satisfy all demands of given electric vehicles; (ii) maximizing the number of fully-charged vehicles where every parking slot can provide a uniform power rate or zero power at any time interval, by seeing the problem as a problem of minimizing the number of late jobs in a setting with parallel machines with release dates and due dates (which is known to be strongly NP-hard due to result of Du and Leung [4]).

Uiterkamp et al. [14] study an EV-charging problem with lower-bound constraints on the provided power (just like our paper does), but their objective is to flatten the combined base-load (given) and charging-load of EVs (to be computed) of the grid.

There are other scheduling papers related to charging of EVs, differing substantially in the constraints or in the objective, see e.g. [8, 16, 6] and references therein for a small subset of such papers.

Existing literature highlights a distinct gap: to the best of our knowledge, no prior work simultaneously incorporates a minimum operating power threshold, passive admission constraints, and a customer-centric hierarchical satisfaction objective.

1.4 Empirical Data and Demand Reconstruction

The experimental evaluation is driven by data collected in a concrete real-world parking facility during Week 51, 2024, one of the highest-demand weeks of the year. Two data sources were combined: (i) gate-access logs recording entry time-stamps, exit time-stamps, and vehicle types, and (ii) charging-session logs recording active charging durations and total energy consumed.

Because historical logs only capture vehicles that successfully accessed a charger, unserved demand had to be reconstructed. Vehicles that arrived when all chargers were occupied were identified by mapping gate arrivals against a minute-by-minute occupancy timeline. Their energy requirements were then generated using Monte Carlo sampling [13] from historical vehicle battery capacities and charging profiles. This reconstruction was developed and validated in previous work [10]. We present it here for completeness as it forms the basis of the dataset used for training and evaluating the scheduling strategies in this paper.

2 The Mixed-Integer Linear Program

The offline problem is given by n vehicles and their arrival times, departure times, requested energy, the lower and upper bounds on admissible power, and by the maximum power capacity of the facility.

To model the scheduling problem as a mixed-integer linear program (MILP), we introduce the variables that define the schedule, i.e., variables $p_{i,t} \in \mathbb{R}_{\geq 0}$, for every $i \in [n]$ and every $t \in [T]$, which express the power (in kW/min) allocated to vehicle i at minute t .

We further define binary indicator variables $s_{i,t} \in \{0, 1\}$, for every $i \in [n]$ and every $t \in [T]$, indicating whether vehicle i is actively charging ($s_{i,t} = 1$) or idle ($s_{i,t} = 0$) at time point t .

To track the number of vehicles that successfully charged 90% of their energy demand, we introduce a binary indicator variable $y_i \in \{0, 1\}$, for every $i \in [n]$, indicating whether vehicle i charged 90% of its requested energy.

Naturally, when solving the resulting MILP, we only instantiate decision variables for the intervals a vehicle is physically present. We define $T_i = \{t_{\text{arr},i}, \dots, t_{\text{dep},i}\}$ as the connection window of vehicle i , and conversely, $N_t = \{i \in [n] \mid t \in T_i\}$ as the set of vehicles present at minute t .

The constraints of the MILP are as follows. First, the total power allocated at any minute cannot exceed the grid limit, i.e.,

$$\sum_{i \in N_t} p_{i,t} \leq P_{\text{grid}} \quad \forall t \in [T].$$

A vehicle can only draw power within its minimum and maximum limits. Using the binary state variable $s_{i,t}$, these bounds are enforced over each vehicle's connection window T_i as follows:

$$p_{i,t} \geq s_{i,t} \cdot P_{\min,i} \quad \forall i \in [n], \forall t \in T_i,$$

$$p_{i,t} \leq s_{i,t} \cdot P_{\max,i} \quad \forall i \in [n], \forall t \in T_i.$$

This ensures that $p_{i,t}$ is either exactly zero or restricted within a valid operating range. We calculate the total energy delivered to a vehicle, using a variable $E_{\text{total},i}$, as the sum of the power it receives across its connection window, scaled by $1/60$ to convert kW-minutes into kWh. An upper-bound constraint ensures the system never delivers more energy than requested:

$$E_{\text{total},i} = \sum_{t \in T_i} \left(p_{i,t} \cdot \frac{1}{60} \right) \quad \forall i \in [n],$$

$$E_{\text{total},i} \leq E_{\text{req},i} \quad \forall i \in [n].$$

Variable y_i needs to be one if and only if the total energy delivered is at least 90% of the requested energy. The following constraints couples y_i with this goal:

$$E_{\text{total},i} \geq (0.9 \cdot E_{\text{req},i}) \cdot y_i \quad \forall i \in [n].$$

Note that this inequality allows $y_i = 0$ even when the energy target is fulfilled. However, because our objective function maximizes the sum of y_i , the optimization process naturally forces y_i to 1 whenever a schedule achieves the 90% threshold.

Our objective function is to maximize, lexicographically, $\left(\sum_{i \in [n]} y_i, \sum_{i \in [n]} E_{\text{total},i}\right)$.

For our experiments, we solve the MILPs using the Gurobi optimization solver [7].

3 NP-Hardness

We establish that the offline EV charging scheduling problem studied in this paper is NP-hard. We show that the problem is NP-hard even if the only optimization goal is the number of vehicles that get 90% of the requested energy.

► **Theorem 1.** *The offline EV charging scheduling problem is NP-hard.*

Proof. We reduce from *3-Partition*, which is strongly NP-hard [5]. An instance of *3-Partition* consists of $3m$ positive integers $a_1, \dots, a_{3m}$ satisfying $\sum_{i=1}^{3m} a_i = mB$ and $B/4 < a_i < B/2$ for all i . The question is whether these integers can be partitioned into m groups of three, each summing to exactly B .

Construction. Given a *3-Partition* instance, we construct an EV scheduling instance as follows. We set the planning horizon to $T = m$ minutes, the number of vehicles to $n = 3m$, and the grid capacity to $P_{\text{grid}} = B$. All vehicles are present for the full horizon, i.e., $t_{\text{arr},i} = 1$ and $t_{\text{dep},i} = m$ for every $i \in [n]$. For each vehicle $i \in \{1, \dots, 3m\}$, we set

$$P_{\min,i} = P_{\max,i} = a_i, \quad \text{and} \quad E_{\text{req},i} = \frac{a_i}{60} \text{ kWh}. \quad (1)$$

Setting $P_{\min,i} = P_{\max,i} = a_i$ is a special case of the generic constraints of the problem: a vehicle charged for one minute at rate a_i receives $a_i \times \frac{1}{60} = E_{\text{req},i}$ kWh, which equals $100\% \geq 90\%$ of its requested energy.

We now show that a valid 3-partition exists if and only if all $3m$ vehicles get at least 90% of requested energy.

Suppose first that a valid 3-partition exists, with groups $G_1, \dots, G_m$ each summing to B . Schedule the vehicles in group G_t to charge during minute t . At each minute, the total allocated power is $\sum_{i \in G_t} a_i = B = P_{\text{grid}}$, satisfying the grid capacity constraint. Each vehicle i is charged for exactly one minute at its fixed rate a_i , receiving $a_i/60 = E_{\text{req},i}$ kWh. Consequently, all $3m$ vehicles meet the 90% threshold of requested energy.

For the other direction, suppose that all $3m$ vehicles get at least 90% of requested energy. Since $P_{\min,i} = P_{\max,i} = a_i$, vehicle i receives power only at rate a_i or not at all; each active minute therefore contributes exactly $a_i/60$ kWh. To attain the minimum $0.9 \cdot a_i/60$ kWh, each vehicle must be charged for at least one complete minute. Because $B/4 < a_i$ for all i , any four vehicles require combined power exceeding $4 \cdot (B/4) = B = P_{\text{grid}}$, so at most three vehicles can charge simultaneously. With m minutes and at most three vehicles per minute, the total number of available vehicle-charging slots (minutes) is at most $3m$. Since each of the $3m$ vehicles requires at least one slot, each vehicle is charged for *exactly* one minute, and no slot is left unused. The vehicles assigned to minute t form a group G_t that partitions $\{1, \dots, 3m\}$, with $\sum_{i \in G_t} a_i \leq B$ by the grid constraints. Summing over all m groups gives $\sum_{t=1}^m \sum_{i \in G_t} a_i = mB$, so each group must sum to *exactly* B . This constitutes a valid 3-partition. ◀

4 The Online Optimization Problem

While a solution to the offline problem establishes a theoretical upper bound on performance of any dynamic load-balancing algorithm, its reliance on perfect future knowledge (and also the computational cost of solving a large integer program at runtime) makes it not viable for real-time deployment. Practical facility management requires online algorithms capable of making decisions using only current-state knowledge and historical data. We compare the following approaches for the online optimization.

4.1 Static Load Balancing

The simplest approach is static, equal-share load balancing (SLB), where the grid capacity P_{grid} is divided evenly among all active vehicles. While easy to implement, this approach struggles as the facility scales. When congestion causes the equal-share allocation to drop below the hardware minimum threshold $P_{\text{min},i}$, the vehicles put themselves into an idle state (not using the power to charge). In our experiments, we use a slightly improved static load balancing, in which the system recognizes when a new vehicle would put (some) cars to idle state, and the system simply denies power to newly arriving vehicles to maintain a safe minimum charge rate for all ongoing sessions. When a vehicle departs, the previously denied vehicles are re-considered for being charged by the system based on the first-come, first-served principle.

4.2 Earliest Deadline First

To address the inadequacies of static load balancing, we evaluate an earliest deadline first (EDF) algorithm. Originally formalized by Liu and Layland [11], standard EDF is a greedy heuristic that sorts present tasks based on their deadlines, allocating full capacity sequentially to the most urgent job until the resource limit is exhausted.

To ensure a fair comparison against the MILP's hierarchical objective, we design a modified two-tier EDF. At each time step t , the algorithm partitions connected vehicles into two priority queues. The *SLA queue* contains vehicles that have not yet received 90% or more of the requested energy. The *Top-off queue* contains vehicles that received at least 90% of the requested energy, but not yet 100%.

We iterate through the SLA queue in ascending order of departure time. For each considered vehicle i , we allocate power at its maximum rate $P_{\text{max},i}$, or the remaining grid capacity if that is smaller, provided the allocation meets the minimum threshold $P_{\text{min},i}$. We then subtract the allocated power from the remaining grid budget and continue to the next vehicle. Once all vehicles in the SLA queue have been served, we repeat the same procedure for the top-off queue using any remaining capacity.

A limitation of this approach is that it allocates power strictly by urgency, with no regard for whether a vehicle can still reach its target. For instances where not all vehicles can obtain 90% of requested energy, EDF may spend grid capacity on a vehicle for which the 90% target is not reachable anymore, regardless of future power allocations, at the expense of a vehicle that could still have been satisfied. We address this with a feasibility-aware extension, called EDF-FC.

4.3 Earliest Deadline First with Feasibility Check

At each minute t , before sorting the SLA queue, EDF-FC evaluates whether each vehicle below its requested 90% target can still reach the target under the best possible future allocation, i.e., if it were to receive its maximum power for the remainder of its stay. Let

$\tau_i(t) = t_{\text{dep},i} - t$ denote the vehicle's remaining connection time in minutes. The maximum energy it could still receive is

$$E_i^{\text{max}}(t) = \tau_i(t) \cdot P_{\text{max},i} \cdot \frac{1}{60}.$$

A vehicle is *feasible* if $E_{\text{total},i}(t) + E_i^{\text{max}}(t) \geq 0.9 \cdot E_{\text{req},i}$, and *infeasible* otherwise. This reclassifies the original SLA queue into two tiers, evaluated in this priority order. *Feasible-SLA queue* contains vehicles from the SLA queue that are feasible. *Infeasible queue* contains vehicles from SLA that are infeasible. *Top-off queue* is the same queue as defined before.

Within each queue, vehicles are ordered by ascending departure time, exactly as in standard EDF. Vehicles in the infeasible queue are not excluded from charging, they are still served with any capacity left over from the feasible-SLA queue. After that, EDF-FC considers the top-off queue. It is a natural idea to merge the infeasible queue with the top-off queue. In our internal experiments, the variant of EDF-FC where these two queues are merged did not provide any statistically meaningful difference.

4.4 Imitation Learning

To investigate whether a learned policy can approximate MILP-quality decisions at a negligible runtime cost, we developed an imitation learning (IL) approach using a random forest classifier [2]. We treat the offline MILP as an oracle: for each vehicle at each minute of the training dataset, the MILP's binary charging decision ($s_{i,t} \in \{0,1\}$) serves as our ground-truth label. We then train the classifier to reproduce these decisions using features observable in real time, effectively learning the MILP's prioritization logic without invoking the solver at runtime.

We evaluate two architectures that differ in their output structure. The *individual-vehicle architecture* is a single-target binary classification problem: it takes one vehicle's features alongside global facility state variables and outputs a single charge or no-charge decision for that vehicle. The *facility-wide architecture* is a multi-output binary classification problem: it takes all charger slots as fixed simultaneous inputs, padding empty slots with zeros and a binary occupancy flag, and outputs one binary decision per slot in a single inference call.

The training data is imbalanced, approximately 74% of labels are idle (0) and 26% are active charge (1). Macro F1-score is used as the optimization criterion during hyperparameter search, which weights both classes equally and is appropriate given this imbalance.

Training labels were derived from MILP solutions over two weeks of historical demand data. A time-based split was used: one additional week was held out for validation and one for testing, since EV arrival patterns are strongly time-dependent and a random split would allow future data to leak into training. Hyperparameters were tuned via randomized search using a predefined train-validation split, training on the two-week period and evaluating each hyperparameter combination on the held-out validation week. This ensures that hyperparameter selection is based on generalization to future unseen data rather than within-training performance. The hyperparameters searched were *number of trees*, *maximum tree depth*, *minimum samples per split*, *minimum samples per leaf*, and *class weight*. The final model was retrained on the full two weeks of training data using the best hyperparameters found, and evaluated on the test week. The full feature list is reported in Appendix 2.

For both architectures, the input features at each time step include temporal context (hour of day, day of week), facility state (number of connected vehicles, grid utilization ratio, total capacity), and per-vehicle state (time remaining until departure, energy still needed, laxity ratio, vehicle type, and hardware power bounds). The *laxity ratio* is the time remaining divided by the energy still needed, with a lower value indicating higher charging urgency.

To guarantee physical grid compliance, both models output binary decisions rather than power values, preventing the classifier from producing physically impossible allocations. These decisions are converted into actual power values using a greedy post-processing algorithm. It iterates through vehicles the classifier marked as active, sorted by laxity ratio in ascending order. For each vehicle, it allocates power greedily within the bounds $[P_{\min,i}, P_{\max,i}]$, respecting the remaining grid budget. If a vehicle cannot receive at least $P_{\min,i}$, due to insufficient remaining capacity, it is skipped. Any residual capacity is then distributed to vehicles below their 90% energy target, again sorted by laxity ratio.

4.5 Event-Driven Model Predictive Control

The offline MILP from Section 2 assumes all vehicle data is known in advance. We now adapt this formulation for online use through a model predictive control (MPC) framework, an optimization-based control approach that repeatedly solves a planning problem over a future time horizon and executes the first step(s) of the resulting plan, before re-solving when new information arrives.

At any moment during facility operation, the set of currently connected vehicles and their remaining energy needs are fully known. We can therefore treat the remaining planning window as a new offline instance: given the vehicles currently present, their remaining requested energy, and their departure times, we solve the MILP from the current minute forward to produce an optimal schedule for the rest of their stay. The facility then executes this schedule minute-by-minute until something changes.

The key question is when to re-solve. If nothing changes between minutes, the previously computed schedule remains optimal and there is no reason to recompute it. We only re-solve upon arrival of a new vehicle.

To keep each re-solve computationally small, we separate connected vehicles into two groups. Vehicles that have not yet reached their 90% target are passed into the MILP. Vehicles that have already met the 90% target are handled by a greedy post-processing step: after the MILP runs, any remaining grid capacity is distributed to these vehicles in order of earliest departure time, up to their maximum power. Our choice for using greedy heuristic after the MILP is executed is motivated by the fact that at that moment the only goal is to maximize the amount of delivered energy, for which the greedy allocation based on the earliest departure time performed very well in our internal evaluations.

4.6 Stochastic Event-Driven Model Predictive Control

The event-driven MPC only considers vehicles that are physically present at the moment of re-solving. It does not account for vehicles that are likely to arrive soon. Without this knowledge, the optimizer may assign power slowly to vehicles that appear to have sufficient time remaining to meet their 90% charge target. When new vehicles arrive and share the same grid budget, those vehicles may no longer receive enough power to finish in time, missing a target that was achievable under the original schedule. To address this, we extend the event-driven MPC by adding predicted future vehicles to each re-solve instance.

When an event triggers a re-solve, we generate a set of synthetic future vehicles representing expected arrivals over the next 4 hours. This prediction limit was chosen to match the median observed stay duration in the historical data. These synthetic vehicles are added to the MILP alongside the physically present ones, so the optimizer accounts for expected future demand when computing the current schedule. Only the schedule computed for real vehicles is executed: synthetic vehicles influence the allocation but are never physically charged.

To generate realistic synthetic arrivals, we model the arrival process separately for three day groups (Monday through Thursday, Friday, and weekend), identified through a preliminary similarity analysis of average hourly arrival counts across days of the week. Within each day group, we model the number of arrivals per minute using distributions fitted to historical data. We first tested whether a Poisson distribution was appropriate for each hour of the day using chi-squared goodness-of-fit tests on per-minute arrival counts and Kolmogorov-Smirnov tests on inter-arrival times. For off-peak hours, the Poisson assumption held. For peak hours, arrival counts showed over-dispersion, meaning the variance exceeded the mean, which a Poisson distribution cannot capture. We therefore fitted a negative binomial distribution to peak hours, which adds a dispersion parameter and was found to pass the goodness-of-fit tests across all hours that had previously failed. The breakdown of these distribution selections and their parameter sets are detailed in Table 3 and Table 4 of the appendix.

The arrival rate varies smoothly over time within each day group, estimated using kernel density estimation [15] fitted to historical arrival timestamps. For each synthetic vehicle, attributes such as stay duration, energy requested, and vehicle type are sampled from historical profiles for that hour of the day.

5 Experimental Results and Comparative Analysis

We evaluate how much each scheduling strategy improves upon the static load-balancing baseline as the number of chargers grows from 6 to 40 under the fixed 50 kW grid limit. Fig. 1 shows the number of vehicles meeting the SLA target and Fig. 2 shows total energy delivered for each strategy across this range. Table 1 reports detailed performance for 27 chargers, selected simply because it is the last point at which the offline MILP meets the SLA target for at least 95% of all electric vehicles (here, 95% is an arbitrary, close-to-100 threshold). Full numerical results across all capacities are reported in Table 5 of the Appendix.

All results are based on a single demand dataset from Week 51, 2024. The data were not used for training of the imitation-learning approach. Each strategy produces a fixed result on this input. The stochastic MPC involves sampling and the results reflect a single run.

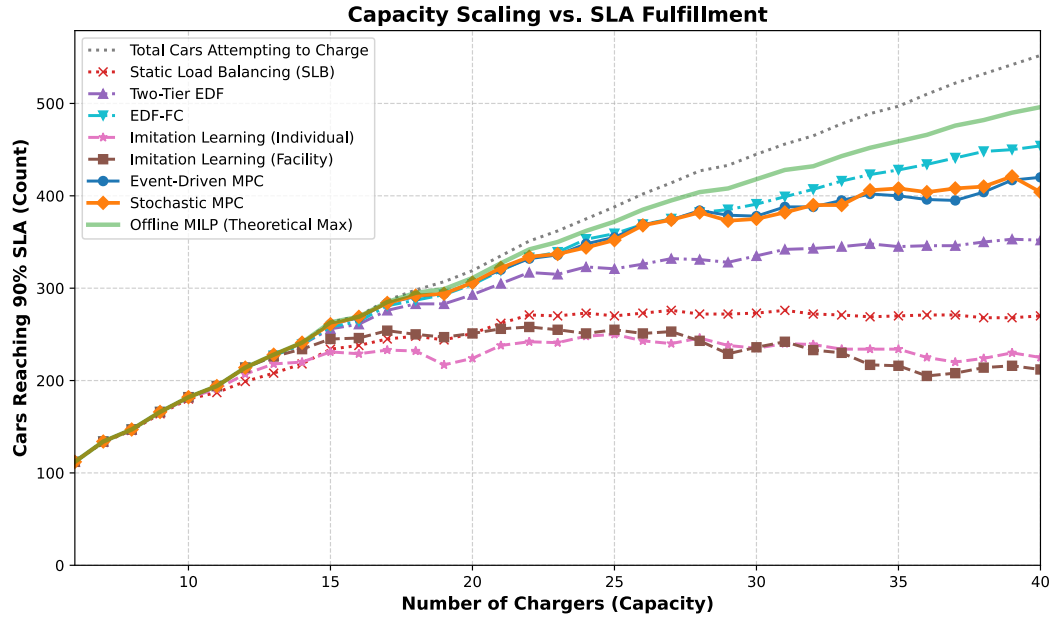
Table 1 Performance at 27 Chargers.

Scheduling Method	90% Charged (Count)	Energy Delivered (kWh)	Total Week Runtime (s)	Mean Step Latency (ms)
Offline MILP	395	5,233.5	-	-
Static Load Balancing (SLB)	276	4,745.9	17.21	0.0941
Two-Tier EDF	332	5,184.2	5.21	0.0090
EDF-FC	375	4,930.9	19.53	0.0356
Imitation Learning (Individual)	240	4,753.8	266.86	19.1075
Imitation Learning (Facility)	253	5,254.4	416.93	30.0551
Event-Driven MPC	374	5,279.8	870.69	0.0344 / 984.3400*
Stochastic Event MPC	374	5,269.3	1,170.25	0.0460 / 1,164.5453*

* Note: Dual values represent *tracking Step* / *Optimization Step* latencies.

5.1 Baseline versus the Offline MILP

We first establish the performance bounds on our dataset. As shown in Fig. 1, SLB satisfies close to 100% of vehicles up to roughly 10 chargers. Beyond this point, total demand frequently exceeds the fixed grid limit. Under SLB, when there is not enough power for



■ **Figure 1** Number of vehicles fulfilling the 90% SLA across number of chargers (6 to 40).

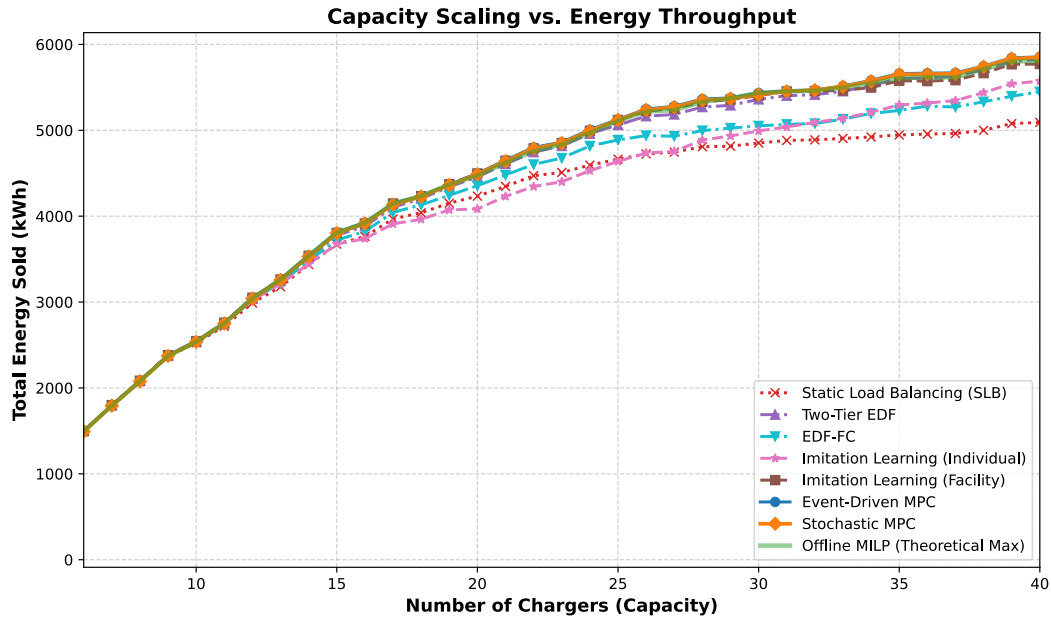
everyone to receive the minimum required charge rate, newly arriving vehicles are denied power so that vehicles already charging can continue to do so. While this protects ongoing sessions, it prevents new arrivals from charging during peak demand, causing SLA fulfillment to plateau after roughly 22 chargers. Interestingly, Fig. 2 shows a matching plateau in energy delivered, but only after 30 chargers. Overall, the results indicate that this baseline strategy cannot effectively scale under congestion on our dataset.

The offline MILP establishes the upper bound, maintaining a 95% of vehicles meeting the SLA target up to 27 chargers and maintaining a 90% of vehicles meeting the SLA target up to 39 chargers on our dataset. At 27 chargers, MILP satisfies 395 vehicles and delivers 5,233.5 kWh, compared to SLB's 276 vehicles and 4,745.9 kWh, a gap of 119 vehicles and 487.6 kWh that represents the maximum recoverable by any online strategy.

5.2 Earliest Deadline First

Despite its simplicity, EDF performs well and far outperforms SLB on our dataset. It satisfies 332 vehicles at 27 chargers, recovering 47% of the gap to the offline MILP in number of vehicles meeting the SLA target (56 vehicles recovered out of the gap of 119 left by the baseline SLB), and closely tracks the offline MILP's energy throughput across the full range tested (Fig. 2). That EDF performs well on total energy-delivered (and, in some cases even outperforms MILP) should not be surprising, since EDF's goal is to allocate as much energy to early-leaving vehicles, thus implicitly aiming to maximize the total-energy delivered as the only optimization goal.

Yet, a visible gap remains in absolute fulfillment of reaching 90% of requested energy (Fig. 1). EDF acts entirely on deadlines and has no way of checking whether a vehicle can still reach its requested energy given its remaining time and energy need. Under heavy congestion it allocates power to vehicles that cannot be satisfied, leaving less capacity for vehicles that could have been.



■ **Figure 2** Total energy sold (kWh) across charger capacities (6 to 40).

5.3 Earliest Deadline First with Feasibility Check

EDF-FC satisfies 375 vehicles at 27 chargers, compared to 332 for standard EDF, closing 43 of the 63-vehicle gap to the offline MILP that standard EDF left unaddressed at this capacity (68%). Energy delivered falls slightly to 4,930.9 kWh, 4.9% below standard EDF's 5,184.2 kWh, despite the higher number of vehicles reaching the SLA target. One of the reasons for this difference is that, since EDF-FC prioritizes charging first the vehicles that can still potentially reach the SLA target, it can happen that a vehicle with early departure does not get charged (because it cannot reach the SLA target) and leaves, and instead another vehicle with later departure gets charged that could have been charged at later time slots as well.

This advantage grows with capacity. As shown in Fig. 1, EDF-FC tracks the MPC strategies closely in terms of the number of vehicles getting 90% of requested energy at lower charger counts, then pulls ahead of both as the facility scales, with the margin widening through the upper end of the tested range. Fig 2 shows the corresponding cost of this advantage: EDF-FC's energy curve sits consistently below the MPC curves across the same range, mirroring the trade-off seen at 27 chargers.

5.4 Imitation Learning Architectures

The imitation learning (IL) strategies failed to match EDF in the number of vehicles meeting the SLA target on our dataset. Both architectures output a binary charge-or-do-not-charge decision per vehicle, and a greedy post-processor converts these into power values without optimizing over the full set of vehicles. The priority ordering learned by the classifier is therefore partly lost, which is reflected in the results: both IL variants satisfy fewer vehicles than EDF despite having access to the same information.

The two architectures diverge on energy delivered. The individual architecture left grid capacity consistently unused, delivering 4,753.8 kWh at 27 chargers. The facility-wide architecture delivered 5,254.4 kWh, actually surpassing the offline MILP's energy delivered,

by learning to activate as many vehicles as possible. This is possible because the MILP’s main objective is to maximize the number of vehicles meeting the SLA target, while IL does not have this direct optimization. However, activating many vehicles without correctly prioritizing them did not translate into SLA gains, as shown in Fig. 1.

5.5 Event Driven and Stochastic MPC

The MPC strategies achieve, together with EDF-FC, the highest number of vehicles meeting the SLA target among all online strategies, matching the offline MILP closely across the full range (Fig. 1). At 27 chargers, the event-driven MPC satisfies 374 vehicles, closing most of the gap to the offline MILP’s 395 vehicles, while delivering 5,279.8 kWh against the offline MILP’s 5,233.5 kWh (Fig. 2), which is by far the most energy delivered among all tested algorithms.

Adding predicted future arrivals in the stochastic MPC yields results almost identical to the reactive version. At 27 chargers both satisfy 374 vehicles, with the stochastic variant delivering 10.5 kWh less. Across the full capacity range the lines overlap almost entirely in Fig. 1, with differences appearing only at a few isolated points. The added predictive complexity offers little advantage, and in some cases even disadvantage, over the reactive event-driven approach on this dataset.

5.6 Computational Feasibility and Trade-offs

Table 1 reports computational performance at 27 chargers. The mean step latency is the average time to compute a power allocation at a single time step.

EDF requires just 0.009 ms per step and completes the full week in 5.21 seconds, making it effectively instantaneous. It satisfies 332 vehicles, recovering most of the gap to the offline MILP without any solver dependency.

EDF-FC adds a small, constant overhead over standard EDF, since the feasibility check is an algorithmic comparison per vehicle per step. Mean step latency rises from 0.0090 ms to 0.0356 ms, and total weekly runtime rises from 5.21 to 19.53 seconds at 27 chargers. Both remain negligible in absolute terms and are cheaper than the IL or MPC strategies, so the feasibility check preserves EDF’s advantage of being solver-free with near-instantaneous computation.

The IL strategies require between 19 and 30 ms per step. The bottleneck is not classifier inference but data preparation: at every time step the controller reconstructs all feature vectors in memory for every active vehicle. Given that their SLA fulfillment actually underperforms the SLB baseline under congestion, this cost is difficult to justify.

The MPC strategies show a split: on minutes where no event has occurred, both variants follow their pre-calculated schedule at 0.034 ms and 0.046 ms per step. At event moments, the event-driven MPC averages 984 ms per optimization step and the stochastic MPC averages 1,164 ms. Because re-solves happen only when the facility state changes, the vast majority of steps run at the tracking rate. Total weekly runtime is 870 seconds for the event-driven MPC and 1,170 seconds for the stochastic variant. The stochastic MPC takes 18% longer per optimization step and adds nearly 300 seconds of total runtime in exchange for satisfying the same number of vehicles and delivering 10.5 kWh less energy.

6 Discussion and Conclusion

The designed heuristic and MPC algorithms outperform, not surprisingly, the SLB system, which confirms that the scheduling algorithms, not the grid limit itself, is the main barrier to expanding the parking facilities.

EDF achieves very good results on real-world data at almost no computational cost, but its main weakness is that it allocates power purely by deadline urgency, with no regard for whether a vehicle can still reach the 90% target given its remaining time and energy need. The improved EDF-FC tackles this issue, quite successfully. At 27 chargers, EDF-FC closes 43 of the 63 vehicles standard EDF left unclosed against the offline MILP, a 68% recovery of that gap, at essentially no added computational cost. Most of EDF's shortfall at this capacity was therefore attributable to misallocated capacity rather than to its lack of joint optimization across vehicles.

A more unexpected result, visible in Fig. 1, is that EDF-FC does not only close ground on the MPC strategies, it overtakes them as the facility scales, with the margin widening toward the upper end of the tested range. This is worth treating with some caution: it is a single-dataset result, and the event-driven MPC theoretically solves an exact lexicographic optimization over all present vehicles at every re-solve, which cannot be worse than a greedy heuristic if no new cars arrive. A plausible explanation for the better performance of EDF-FC is that the MPC's advantage is only local; it optimizes exactly for the vehicles present at the moment of each re-solve, but has no visibility into vehicles that will arrive before the plan is next revisited, so a schedule optimal at the moment it is computed can become suboptimal once new vehicles arrive.

The stochastic MPC adds predicted future arrivals to the event-driven MPC but produces nearly identical results. The arrival distribution was validated and the prediction model is statistically well-grounded, so the issue shall not be the prediction quality. The event-driven MPC already re-solves the moment a new vehicle arrives, so the window in which a prediction could change the schedule is very short. A more accurate forecasting model might improve this slightly, but is unlikely to change the finding that non-predictive event-driven optimization is sufficient under current demand patterns.

The IL models underperformed in all areas. The core issue is that a binary classifier cannot directly control power, so the optimization intelligence in the learned policy gets lost when the post-processor converts decisions into actual kilowatts. A natural extension and a possible future work are machine learning models that output power values directly rather than binary decisions, though any machine learning approach on physical hardware must guarantee that hardware constraints are never violated, which adds further engineering complexity.

All experimental results assume a fixed target of 90% of requested energy. Experiments with other targets is a natural future work, albeit our preliminary experiments show little changes in how the scheduling algorithms mutually compare.

References

- 1 Bahram Alinia, Mohammad Sadegh Talebi, Mohammad H. Hajiesmaili, Ali Yekkehkhany, and Noel Crespi. Competitive online scheduling algorithms with applications in deadline-constrained EV charging. In *Proceedings of the 26th IEEE/ACM International Workshop on Quality of Service (IWQoS)*, pages 1–10, 2018. doi:10.1109/IWQoS.2018.8624184.
- 2 Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

- 3 Shiyao Chen, Lang Tong, and Ting He. Optimal deadline scheduling with commitment. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 111–118, 2011. doi:10.1109/Allerton.2011.6120157.
- 4 Jianzhong Du and Joseph Y.-T Leung. Minimizing the number of late jobs on unrelated machines. *Operations Research Letters*, 10(3):153–158, 1991. doi:10.1016/0167-6377(91)90032-K.
- 5 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, CA, USA, 1979.
- 6 Mahmoud Golabi, Abdenmour Azerine, Ammar Oulamara, and Lhassane Idoumghar. Advanced models and a hybrid method for electric vehicle charging scheduling with diverse charger characteristics. *RAIRO-Oper. Res.*, 59(3):1681–1701, 2025. doi:10.1051/ro/2025056.
- 7 Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2024. URL: <https://www.gurobi.com>.
- 8 Alejandro Hernández-Arauzo, Jorge Puente, Ramiro Varela, and Javier Sedano. Electric vehicle charging under power and balance constraints as dynamic scheduling. *Computers & Industrial Engineering*, 85:306–315, 2015. doi:10.1016/j.cie.2015.04.002.
- 9 International Electrotechnical Commission. IEC 61851-1: Electric vehicle conductive charging system—part 1: General requirements. IEC Standard 61851-1, International Electrotechnical Commission, 2017.
- 10 Bruna Cavalcanti Lauro. Semester report of the ke-at-work honour program. Technical report, Department of Advanced Computing Sciences, Maastricht University, Maastricht, The Netherlands, 2025.
- 11 Chang L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973. doi:10.1145/321738.321743.
- 12 Francesco Marra, Guang Ya Yang, Chresten Træholt, Esben Larsen, Claus Nygaard Rasmussen, and Shi You. Demand profile study of battery electric vehicle under different charging options. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, 2012. doi:10.1109/PESGM.2012.6345063.
- 13 Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, New York, NY, USA, 2 edition, 2004. doi:10.1007/978-1-4757-4145-2.
- 14 Martijn H.H. Schoot Uiterkamp, Thijs van der Klauw, Marco E.T. Gerards, and Johann L. Hurink. Offline and online scheduling of electric vehicle charging with a minimum charging threshold. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2018. doi:10.1109/SmartGridComm.2018.8587556.
- 15 Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London, UK, 1986.
- 16 I. Zaidi, A. Oulamara, L. Idoumghar, and M. Basset. Maximizing the number of satisfied charging demands of electric vehicles on identical chargers. *Omega*, 127, 2024. doi:10.1016/j.omega.2024.103106.
- 17 Imene Zaidi. *Electric Vehicle Charging Scheduling Problems: Complexity and Solution Approaches*. Phd thesis, L’université de Haute Alsace, Mulhouse, France, 2022.

A Appendix

A.1 Imitation Learning Model Inputs and Features

Table 2 describes the input features used by both machine learning models to make scheduling decisions.

■ **Table 2** Features Used by the Machine Learning Models.

Feature Group	Feature Name	Description
Temporal Context	hour_of_day	Maps the 24-hour daily demand cycle ($0 \dots 23$).
	day_of_week	The days of the week ($0 \dots 6$).
Facility State	connected_ev_count	The total number of connected vehicles ($ N_t $).
	grid_utilization	The total power demand relative to the limit P_{grid} .
	available_capacity	The remaining power headroom at the transformer (kW).
Per-Vehicle State	time_to_departure	Minutes until the vehicle's departure $t_{\text{dep},i}$.
	energy_remaining	Unfulfilled energy required to hit the user target ($E_{\text{req},i}$).
	laxity_ratio	Remaining parking time over remaining energy needed.
	vehicle_type	Fleet category identifier ($\text{BEV} = 1, \text{PHEV} = 0$).
	power_bounds	Bounds on the charge rates ($P_{\text{min},i}, P_{\text{max},i}$).

A.2 Arrival Process Modeling and Fitted Parameters

This section describes the probability distributions used to model future vehicle arrivals in the stochastic MPC framework. We split the week into three day-groups: Early Week (Monday-Thursday), Friday, and Weekend (Saturday-Sunday). Out of the 72 hourly blocks in a week, 44 low-demand hours are modeled using a Poisson distribution (λ), and 28 busy hours are modeled using a Negative Binomial distribution (r, p). Table 3 shows how many hours use each distribution type. Table 4 lists the exact calculated parameters for selected hours across the week.

■ **Table 3** Distribution Choices for the 72 Hourly Blocks.

Day-Group Segment	Poisson Hours	Negative Binomial Hours	Total
Early Week (Mon–Thu)	15 Hours	9 Hours	24 Hours
Friday	14 Hours	10 Hours	24 Hours
Weekend (Sat–Sun)	15 Hours	9 Hours	24 Hours
Total	44 Hours	28 Hours	72 Hours

A.3 Complete Capacity Sensitivity Data Matrix

Table 5 provides the comprehensive numerical results for all evaluated scheduling strategies across the full sensitivity spectrum from 6 to 40 physical chargers under the fixed 50 kW transformer constraint.

■ **Table 4** Fitted Parameters for Selected Hours.

Day-Group	Hour	Time Description	Distribution	Parameters
Early Week	0	Night	Poisson	$\lambda = 0.0002$
	5	Morning Transition	Negative Binomial	$r = 0.9890, p = 0.9343$
	7	Morning Rush	Negative Binomial	$r = 2.5900, p = 0.9389$
	11	Mid-Day	Negative Binomial	$r = 0.5310, p = 0.8294$
	16	Afternoon	Poisson	$\lambda = 0.0658$
	18	Evening	Negative Binomial	$r = 0.8530, p = 0.9540$
Friday	0	Night	Poisson	$\lambda = 0.0003$
	7	Morning Rush	Poisson	$\lambda = 0.1535$
	10	Morning Peak	Negative Binomial	$r = 3.2560, p = 0.9358$
	13	Mid-Day	Poisson	$\lambda = 0.1288$
	19	Evening Peak	Negative Binomial	$r = 0.7010, p = 0.9362$
Weekend	1	Night	Poisson	$\lambda = 0.0000$
	5	Morning Transition	Negative Binomial	$r = 0.0450, p = 0.8351$
	9	Morning Peak	Negative Binomial	$r = 2.2790, p = 0.9108$
	12	Noon Peak	Negative Binomial	$r = 4.9760, p = 0.9194$
	16	Afternoon	Poisson	$\lambda = 0.0792$

■ **Table 5** SLA Fulfillment and Energy Delivered per Strategy Across Changer Capacities (6 to 40).

Cap.	Total Cars	Offline MILP		SLB Baseline		Two-Tier EDF		EDF-FC		IL Individual		IL Facility		Event MPC		Stochastic MPC	
		SLA	Energy	SLA	Energy	SLA	Energy	SLA	Energy	SLA	Energy	SLA	Energy	SLA	Energy	SLA	Energy
6	112	112	1492.8	112	1492.8	112	1498.4	112	1493.3	112	1498.4	112	1498.6	112	1498.4	112	1492.8
7	134	134	1793.3	133	1792.1	134	1799.0	134	1792.6	134	1799.7	134	1799.9	134	1799.7	134	1793.3
8	147	147	2077.2	146	2075.0	147	2081.3	147	2075.3	147	2083.1	147	2084.5	147	2084.1	147	2077.2
9	166	166	2373.2	164	2365.9	166	2377.1	166	2369.5	165	2376.8	166	2381.5	166	2380.5	166	2372.8
10	182	182	2536.4	180	2526.0	182	2540.8	182	2532.5	181	2538.3	182	2545.1	182	2545.1	182	2536.2
11	194	194	2750.6	187	2717.5	194	2749.2	194	2739.3	191	2742.5	194	2759.9	194	2760.2	194	2750.5
12	214	214	3041.0	199	2989.4	214	3036.7	214	3025.6	207	3022.4	214	3049.8	214	3050.8	214	3039.8
13	228	228	3261.2	208	3178.3	227	3243.7	227	3223.9	218	3211.3	226	3263.3	228	3266.1	228	3256.4
14	241	241	3539.1	218	3437.4	239	3510.7	239	3479.4	220	3442.9	234	3538.4	241	3542.3	241	3531.0
15	263	263	3807.4	234	3674.6	256	3776.5	258	3721.7	231	3674.8	245	3804.1	261	3813.0	261	3801.4
16	270	269	3921.7	238	3763.7	261	3883.7	264	3821.8	229	3739.0	246	3913.8	268	3925.6	269	3914.4
17	287	285	4148.9	245	3967.5	276	4102.8	281	4044.6	233	3910.4	254	4147.6	284	4146.7	284	4134.5
18	298	295	4235.1	248	4040.1	283	4204.1	287	4131.1	232	3963.3	250	4232.9	291	4235.8	292	4227.6
19	307	299	4368.3	244	4151.2	283	4340.6	293	4243.3	217	4073.0	247	4373.4	294	4365.3	294	4361.2
20	319	311	4470.5	251	4233.4	293	4456.7	304	4354.6	224	4083.9	251	4497.7	306	4497.5	306	4487.5
21	335	327	4621.9	262	4346.9	305	4607.9	319	4479.5	238	4230.3	256	4645.4	320	4651.1	322	4639.3
22	351	342	4756.3	271	4471.0	317	4745.5	333	4602.2	242	4344.7	258	4789.6	332	4801.1	334	4787.2
23	362	350	4832.1	270	4509.1	315	4817.0	339	4676.6	241	4400.8	255	4849.8	336	4861.7	337	4851.4
24	375	362	4968.1	273	4594.2	323	4960.3	353	4817.7	248	4530.0	251	4993.1	348	5006.2	344	4992.1
25	388	372	5110.1	270	4664.3	321	5062.1	359	4887.3	250	4637.2	255	5119.7	355	5128.1	352	5120.1
26	402	385	5218.0	273	4724.6	326	5163.8	369	4938.1	243	4735.9	251	5222.7	369	5247.4	368	5236.7
27	414	395	5233.5	276	4745.9	332	5184.2	375	4930.9	240	4753.8	253	5254.4	374	5279.8	374	5269.3
28	427	404	5330.8	272	4808.7	331	5270.3	382	4996.3	246	4886.1	243	5331.6	384	5364.4	382	5350.6
29	433	408	5373.1	272	4814.5	328	5290.2	385	5025.7	238	4933.7	229	5357.6	379	5377.2	373	5368.6
30	445	418	5436.8	273	4852.1	335	5360.2	391	5050.8	235	4992.4	236	5403.9	378	5434.0	375	5416.8
31	456	428	5452.5	276	4881.9	342	5402.6	399	5069.5	240	5038.8	242	5452.0	388	5461.7	382	5453.3
32	465	432	5460.0	272	4888.7	343	5414.8	407	5078.2	239	5091.4	233	5456.6	388	5469.4	390	5467.3
33	478	443	5496.5	271	4905.3	345	5454.3	416	5130.2	234	5139.0	230	5464.4	395	5515.6	390	5508.7
34	489	452	5546.6	269	4921.9	348	5521.2	423	5196.1	234	5209.9	217	5497.2	402	5581.7	406	5570.6
35	497	459	5606.7	270	4946.3	345	5602.5	428	5230.8	234	5294.5	216	5576.0	400	5660.9	408	5651.0
36	510	466	5617.7	271	4955.5	346	5610.3	434	5282.0	225	5317.2	205	5572.7	396	5665.2	404	5652.8
37	522	476	5614.5	271	4962.9	346	5626.6	441	5271.1	220	5346.5	208	5583.3	395	5669.1	408	5657.9
38	532	482	5710.8	268	4999.3	350	5708.7	448	5331.7	224	5439.1	214	5665.1	404	5747.4	410	5739.5
39	542	490	5798.8	268	5078.4	353	5800.9	450	5396.9	230	5542.0	216	5768.3	417	5844.8	421	5834.2
40	552	496	5807.1	270	5091.1	352	5828.2	454	5448.2	225	5572.6	212	5771.1	420	5854.3	404	5847.0

ASPaeroFlow: Decomposition Heuristics for Joint Air Traffic Flow & Capacity Management

Alexander Beiser ✉ 

TU Wien, Vienna, Austria

Markus Hecher ✉ 

University of Potsdam, Potsdam, Germany

CNRS, Artois University (CRIL), Artois, France

Nysret Musliu ✉ 

TU Wien, Vienna, Austria

Georg Trausmuth ✉

Frequentis AG, Vienna, Austria

Stefan Woltran ✉ 

TU Wien, Vienna, Austria

Abstract

While mathematical models act as vital decision support systems for operational Air Traffic Flow and Capacity Management (ATFCM), existing approaches isolate Air Traffic Flow Management (ATFM) from Dynamic Airspace Configuration (DAC). This separation introduces an unresolved circular dependency between fixed-demand and fixed-capacity assumptions. Although joint optimization resolves this gap, the enlarged search space renders exact models computationally intractable for medium- to large-scale instances. To bridge this gap, we propose *ASPaeroFlow*: a heuristic for the joint ATFCM; it combines instance-space decomposition heuristics with a local exact approach using Answer Set Programming. We benchmark ASPaeroFlow from small to industry-sized instances and compare it with exact and alternative approaches. The results indicate that (1) the heuristic provides a computational middle ground between exact methods and operational baselines; (2) simultaneous optimization can outperform sequential optimization on joint ATFCM; and (3) an ablation study indicates that DAC has a larger impact on solution quality than flow measures.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence

Keywords and phrases Air Traffic Flow and Capacity Management, ATFCM, Answer Set Programming, Heuristics, Open Data

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.10

Related Version *Extended Version*: <https://arxiv.org/abs/2608.09315>

Supplementary Material *Dataset*: <https://doi.org/10.5281/zenodo.21869481>

Funding This research was supported by Frequentis and the Austrian Science Fund (FWF), grant 10.55776/COE12.

Markus Hecher: Hecher has been supported by the French National Research Agency (ANR), grant ANR-25-CE23-7647-01, and the Austrian Science Fund (FWF), grant 10.55776/J4656.

Acknowledgements We thank Dr. Alba Agustín for sharing the literature benchmarks.

1 Introduction

Operational *Air Traffic Flow and Capacity Management* (ATFCM) aims to balance demand and capacity to ensure the safe and efficient flow of air traffic. Demand is the number of flights intending to traverse through a *sector*, while capacity is the maximum number of



© Alexander Beiser, Markus Hecher, Nysret Musliu, Georg Trausmuth, and Stefan Woltran; licensed under Creative Commons License CC-BY 4.0

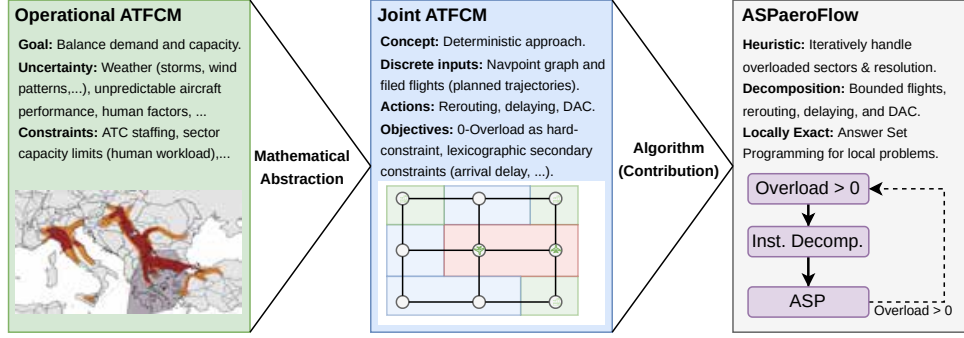
26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 10; pp. 10:1–10:20

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Illustrative schematics of the problem setting including the main contribution.

flights a sector can safely handle. Sector capacity is limited by human factors, as *Air Traffic Controllers* (ATCs) are responsible for ensuring separation between aircraft. *Overloading* a sector means that more flights are in the sector than can be handled by ATCs. To ensure safety, demand must remain below capacity at all times [13].

Automated decision support systems help operational ATFCM by optimizing on mathematical models, which suggest aircraft reroutings, delayings, or sector restructurings to optimize airspace usage. However, the operationally used algorithm *Computer-Assisted Slot Allocation* (CASA), which works according to a “First Planned - First Served” scheme, has been shown to yield unsatisfactory results compared to more sophisticated optimization models [19, 10]. To circumvent this limit, state-of-the-art (SOTA) work proposes mathematical models for optimizing ATFCM. Still, SOTA approaches can be largely grouped into one of two classes: work focusing on the aircraft-level (so-called *Air Traffic Flow Management* (ATFM)) via delays and reroutes [8, 2, 21] and work focusing on the sector-level (so-called *Dynamic Airspace Configuration* (DAC)) [16, 33, 27, 11]. Treating ATFM and DAC separately introduces unresolved circular dependencies and makes it hard to compare the actual benefits of ATFM/DAC. In contrast, *Joint ATFCM* methods [5] would (1) resolve the circular dependency between fixed-capacity and fixed-demand assumptions, (2) render algorithms for isolated subproblems comparable against a joint benchmark, and (3) enable simultaneous optimization paradigms, which exploit the enlarged search space. However, while joint ATFCM models exist, their enlarged search space renders them computationally intractable for medium- and large-scale instances when solved exactly. Therefore, the primary research gap is the lack of *scalable* joint ATFCM methods for industry-sized instances.

Contributions. This paper partially addresses this gap by proposing *ASPaeroFlow*: a heuristic for *joint ATFCM* which optimizes ATFM and DAC simultaneously. See Figure 1 for the conceptual steps. In detail, our contributions are as follows:

- **ASPaeroFlow.** We propose heuristics that derive 0-overload solutions from initial flight schedules and sector configurations. The approach employs an iterative instance-space decomposition to partition global airspace overloads into localized, timestep-specific subproblems. These tractable instances are solved exactly via the Logic Programming technique Answer Set Programming (ASP) [17], leveraging its expressive rule-based modeling and native integration of soft constraints for domain-specific optimization.
- **Data Generator.** We generate realistic industry-sized (large) instances on real-world navpoint graphs for scaling analysis.

- **Benchmarks & Ablation.** We evaluate ASPaeroFlow in small [5] to large instances, and other instances from the literature [2]. ASPaeroFlow bridges the scalability challenge of exact models while significantly outperforming operational baselines. Further, the experiments suggest that simultaneous optimization outperforms sequential approaches when DAC alone does not suffice, and an ablation study highlights DAC as the primary driver for overload reduction.

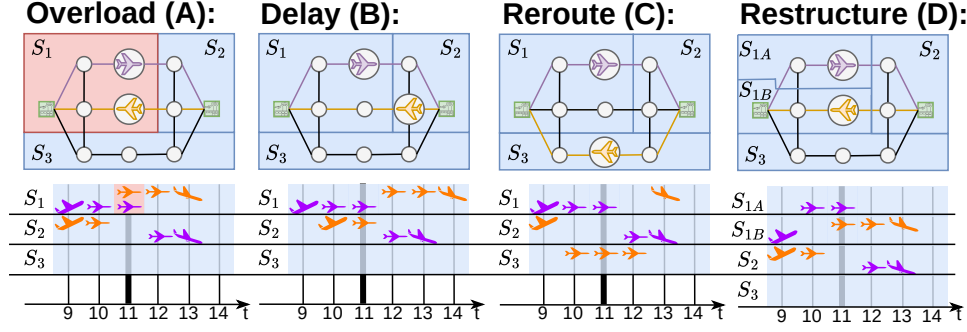
Related Work. **Dynamic Airspace Configuration (DAC)** constructs airspace from atomic blocks, distinguishing it from free-partitioning DAS [26]. Automated sector design utilizes genetic algorithms, Voronoi diagrams [30, 33, 32, 11], graph methods [20], machine learning [31], and MIP [27]. Heuristics for sector design exist: sectors should be connected; flights should stay in a sector for a minimum amount of time and not re-enter [30]. Joint ATFCM constructs composite sectors (containing multiple atomic sectors) from atomic sectors (which represent exactly one navpoint vertex). ASPaeroFlow ensures connected sectors. **Flow Optimization (ATFM)** aims at routing network flow (aircraft) optimally, while adhering to side constraints. The operationally used *Computer-Assisted Slot Allocation (CASA)* heuristics work according to the “First Planned - First Served” principle and delay aircraft [19, 10]. Optimization models addressing delay and rerouting explore fairness [6], lexicographic objectives [15], and computational aspects [7, 9, 14, 3]. It was shown that the restriction to delaying is already *NP-hard* [8]. Crucially, most existing methods integrate DAC solely as a fixed input to ATFM [2, 21]; the recently proposed joint ATFCM model [5] circumvents this, however, *joint sequential or simultaneous optimization for industry-sized instances remains an open challenge*, as exact methods struggle already on small instances. **Answer Set Programming (ASP)** is a symbolic-AI paradigm rooted in logic programming offering exact optimization with high interpretability, deployed for example in industrial scheduling [4, 12, 1]. Regarding aviation, proposals for the usage of ASP for efficient drone scheduling [28] or optimization for mission schedules on vertiports [25] exist. We use ASP for several reasons, including its “natural” problem modeling, rapid prototyping, and future Explainable AI (XAI) integration potential. To circumvent ASP’s primary scalability limitation – the *grounding bottleneck* [29] – we embed it within a problem-decomposition heuristic [18].

Next, we present the necessary preliminaries (Section 2), followed by the introduction of the ATFCM model in Section 3 and ASPaeroFlow (Section 4). We present the experiments and their results in Section 5 and then conclude (Section 6).

2 Preliminaries

Graph Theory. We consider weighted undirected graphs $G = (V, E)$ using standard Euclidean or geodesic distance metrics. A trajectory tr is a simple path augmented with strictly increasing timestamps, yielding a sequence of (v_i, t_i) pairs.

Answer Set Programming (ASP). ASP [24] is a declarative, logic-based problem-solving paradigm. We omit formal semantics (e.g., stable models) for brevity, referring to standard literature [17], and focus on the syntax necessary for our encodings. An ASP program consists of rules $H \leftarrow B$, where the head H is derived if the body B is satisfied. Variables are capitalized. Generating an answer set (solution) to an ASP program is usually done by first *grounding* (instantiating the variables) and then solving. We can use systems such as Clingo [22] to generate an answer set. We utilize standard extensions [23] such as constraints



■ **Figure 2** Schematics of the ATFCM problem with available actions. At time 11 an overload in sector S_1 occurs (A). The model allows for three ways to resolve this issue: Apply a delay to a flight (B), reroute a flight (C), or restructure the airspace (D).

(empty head, which invalidate a solution if the body is true), choice rules (to define search spaces), aggregates, and soft constraints (for lexicographic optimization). The following snippet illustrates these constructs.¹

```

1 % Facts (no variables and no body) describing the input data:
2 config(0,10). config(1,5). fpl(0,5,1). fplt(0,7,2).
3 % Rule expressing flights are rerouted unless they choose traj. 0.
4 reroute(ID) ← flightID(ID), not chtrj(ID,0).
5 % Choice rule: Choose exactly one trajectory P for each flight ID.
6 1{chtrj(ID,P):trj(ID,P)}1 ← flightID(ID).
7 % Rule with Aggregate: Count demand and flag overloads.
8 overload(SEC,T,LOAD-CAP) ← sec_cap(SEC,CAP,T),#count{ID:sec_f(ID,SEC,T)}=LOAD,LOAD>CAP.
9 % Constraint: Reject solutions where a flight does not occur.
10 ← flightID(ID), not flight_occurs(ID).
11 % Soft Constraint: Minimize overload at priority level 10.
12 ← overload(SEC,T,OVER). [OVER@10,SEC,T]

```

3 Joint ATFCM Model

We present the Joint ATFCM model [5], where Figure 2 shows our running example.

► **Definition 1** (Navpoint Graph). *The navpoint graph $G = (V, E)$ is an undirected graph, where each vertex is labeled with spatial geodesic coordinates. We distinguish between airports and en-route vertices, where with $\text{airport}(G)$ and $\text{en-route}(G)$ we denote the respective sets. Let $d : V^2 \rightarrow \mathbb{R}$ be the distance function.*

► **Example 2.** Figure 2 shows a navpoint graph G (gray). Let $G = (V, E)$ with $V = \{a_0, a_1, v_0, \dots, v_8\}$, where $\{a_0, a_1\} \subseteq \text{airport}(G)$ and $\{v_0, \dots, v_8\} \subseteq \text{en-route}(G)$. Further, $E = \{(a_0, v_0), (a_0, v_3), (a_0, v_6)\} \cup \{(v_0, v_1), (v_1, v_2), (v_3, v_4), (v_4, v_5), (v_6, v_7), (v_7, v_8)\} \cup \{(v_2, a_1), (v_5, a_1), (v_8, a_1)\}$. We assume $\forall v_i, v_j \in V : d(v_i, v_j) = 1$

► **Definition 3** (Time model). *T_{gran} discretizes time into T_{gran} -timesteps per hour. We assume one-day instances; then the set of timesteps T is $T := \{t \mid t \in \mathbb{N}, 0 \leq t \leq T_{\text{gran}} \cdot 24\}$.*

► **Example 4.** For $T_{\text{gran}}=4$ we make 15-minute timesteps, i.e., $4 \cdot 24 + 1 = 97 = |T|$ timesteps.

¹ Note that the predicates used in this snippet (e.g., `reroute/1`, `overload/3`) are illustrative; the complete ASP encoding for our model is defined later in Section 4.4. and the appendix.

► **Definition 5** (Sector Configuration). A sector i is a set of vertices $\text{sec}(i, t) \subseteq V$, identified by a representative sector vertex $i \in V$, where $i \in \text{sec}(i, t)$ ensures the sector index directly corresponds to one of its contained vertices. ATC-teams have a fixed capacity and have a fixed location, implying atomic capacities of vertices c_i for $i \in V$ ($t = 0$): $\text{cap}(i, 0) = c_i$. A composite sector has capacity $\text{cap}(i, t) = \max_{j \in \text{sec}(i, t)} \text{cap}(j, 0)$, simulating that the best ATC-team of a region takes command. To get the sector i for a vertex $v \in V$ for $t \in T$, we define $i = \text{sec}^{-1}(v, t)$.

► **Example 6.** Ctd. example. Static model: $\forall t \in T$. $\text{sec}(a_0, t) = \{a_0\}$, $\text{sec}(a_1, t) = \{a_1\}$, $\text{sec}(v_0, t) = \{v_0, v_1, v_3, v_4\}$ (S_1), $\text{sec}(v_2, t) = \{v_2, v_5\}$ (S_2), and $\text{sec}(v_6, t) = \{v_6, v_7, v_8\}$ (S_3). Dynamic model, for: $9 \leq t \leq 13$: $\text{sec}(v_0, t) = \{v_0, v_1\}$ (S_{1A}) and $\text{sec}(v_3, t) = \{v_3, v_4\}$ (S_{1B}). Atomic capacity: $\forall v \in V : \text{cap}(v, 0) = 1$. Static model: $\forall t > 0$: $\text{cap}(a_0, t) = \text{cap}(a_1, t) = \text{cap}(v_0, t) = \text{cap}(v_2, t) = \text{cap}(v_6, t) = 1$, with a combined capacity of 5. Dynamic model (**SEC**): for $9 \leq t \leq 13$ the combined capacity is 6 (increase of capacity through splitting).

► **Definition 7** (Flight Model). Flights $f \in F$ are modeled as trajectories on the navpoint graph. Let $f \in F$ be a flight and f_{id} be its flight id: $f = (id_f, ((v_0, t_0), (v_1, t_1), \dots, (v_n, t_n)))$.

► **Definition 8** (Aircraft Model). We map flights to aircraft (as a single physical aircraft can have multiple flights per day, meaning a delay in an early flight can propagate downstream): an aircraft $a \in A$ is a three tuple (id_a, v_a, F_a) , where id_a is the unique id, v_a is the velocity, and F_a is its set of flights.

► **Example 9.** Ctd. example. Let $A = \{p_0, p_1\}$ with $p_0 = (id_{p_0}, v_{p_0}, \{f_0\})$ and $p_1 = (id_{p_1}, v_{p_1}, \{f_1\})$. Let $f_0, f_1 \in F$ with $f_0 = (0, ((a_0, 9), (v_0, 10), (v_1, 11), (v_2, 12), (a_1, 13)))$ and $f_1 = (1, ((a_1, 9), (v_5, 10), (v_4, 11), (v_3, 12), (a_0, 13)))$.

► **Definition 10** (ATFCM Instance). An ATFCM instance is $I = (G, T_{gran}, T, \text{sec}, \text{cap}, A)$, where G is a navpoint graph, T_{gran} defines the time granularity, T the timesteps, sec the initial sector configuration, cap the atomic capacities, and A the aircraft. We denote with F the set of filed flights. All ids are unique.

► **Definition 11** (ATFCM Solution). The ATFCM solution is given by: $\mathbf{S} = (\mathbf{T}, \mathbf{SEC}, \mathbf{A})$, where $\mathbf{T}$ are the solution timesteps, $\mathbf{SEC}$ is the solution DAC, and $\mathbf{A}$ are the solution aircraft. $\mathbf{T}$ might be expanded beyond the initial T (an instance is always solvable). The flown flights $f \in \mathbf{F} = \bigcup_{(id_a, v_a, F_a) \in \mathbf{A}} F_a$ may be delayed. If a flight is delayed and/or rerouted, this is indicated by $r_f = 1$. Sectors in $\mathbf{SEC}$ are allowed to be changed. We require that airport sectors remain atomic and en-route sectors remain connected to partially accommodate operational requirements. **Demand.** Let $f \in \mathbf{F}$ be a flight reaching navpoint $v_a \in V$ at time t_a and then flying to navpoint $v_b \in V$, which it reaches at time t_b . Assuming that $\Delta t = t_b - t_a$, then f is in the timespan $t \in [t_a, t_a + \frac{\Delta t}{2}]$ in sector $s_i = \text{sec}^{-1}(v_a, t)$ ($\text{over}(f, s_i, t) = 1$), and for $t' \in [t_a + \frac{\Delta t}{2} + 1, t_b]$ in sector $s_{i+1} = \text{sec}^{-1}(v_b, t')$ ($\text{over}(f, s_{i+1}, t') = 1$). For a time $t \in T$, and a sector $i \in V$, we measure demand $q(i, t)$, as the number of aircraft currently in the sector, i.e., $q(i, t) = \sum_{f \in \mathbf{F}} \text{over}(f, i, t)$. For each timestep $t \in \mathbf{T}$, we define with $\mathbf{SEC}$ the DAC. We require for a solution $\mathbf{S}$ to be accepted that for all $t \in \mathbf{T}$, $i \in V$, demand $q(i, t) \leq \text{cap}(i, t)$.

► **Example 12.** Overload (A): Let $\mathbf{S} = (T, \text{sec}, A)$, then $q(v_0, 11) = 2 > \text{cap}(v_0, 11) = 1$. Delay (B): Let $\mathbf{S} = (T, \text{sec}, \mathbf{A})$ with $\mathbf{A}$, where $\mathbf{F} = \{f_0, f'_1\}$ s.t. $f'_1 = (1, ((a_1, 10), (v_5, 11), (v_4, 12), (v_3, 13), (a_0, 14)))$. Reroute (C): Let $\mathbf{S} = (T, \text{sec}, \mathbf{A})$ with $\mathbf{A}$, where $\mathbf{F} = \{f_0, f'_1\}$ s.t. $f'_1 = (1, ((a_1, 9), (v_8, 10), (v_7, 11), (v_6, 12), (a_0, 13)))$. Reconfig (D): Let $\mathbf{S} = (T, \mathbf{SEC}, A)$ with $\mathbf{SEC}$ (Ex. 6).

■ **Table 1** Representative parameter bounds used in $ASPaeroFlow_{r,d}$'s local ASP subproblems. Exact values are variant dependent (cf. Section 5).

Chosen bounds in instance-space decomposition		
Parameter	Operational rationale	$ASPaeroFlow_{r,d}$
$f_{\max}$	Limit the local subproblem to flights that actually contribute to the current overload; keeps grounding and solving stable while capturing the dominant contributors.	2 flights
$d_{\max}$	Strategic regulations usually consider bounded departure shifts; a short delay window provides a realistic control measure without exploding combinations.	5 steps
$r_{\max}$	A small set of diverse candidate routes (e.g., k -shortest with similarity filtering) reflects that only few reroute options are operationally acceptable and keeps the action set manageable.	3 routes
$p_{\max}$	Use a small set of sector split options for the overloaded sector; captures the main capacity measure (partitioning) while avoiding enumeration of arbitrary partitions.	2 splits

► **Definition 13** (Optimization Problem). *A solution $\mathbf{S}$ is optimized according to efficiency (Arrival Delay, Active en-route sectors, and Navpoint-sector changes) and stability/fairness (Reroutes/delays and Reconfigurations) soft constraints. An accepted solution $\mathbf{S}$ is strictly preferred to $\mathbf{S}'$ ($\mathbf{S} \prec \mathbf{S}'$) if its objective tuple is lexicographically smaller: $(K_d, K_{\#s}, K_{ds}, K_r, K_c)\mathbf{S} \prec_{lex} (\dots)\mathbf{S}'$. Let t_f^a and t_f^e be the actual and expected arrival times. The individual criteria to be minimized are defined as follows, utilizing indicator function $\mathbb{I}(\cdot)$:*

$$K_d = \sum_{f \in \mathbf{F}} d_f = \sum_{f \in \mathbf{F}} t_f^a - t_f^e \quad (\text{Arrival delay})$$

$$K_{\#s} = \sum_{t \in \mathbf{T}_{>0}} \sum_{i \in V} \mathbb{I}(|\mathbf{SEC}(i, t)| > 0) \quad (\text{Active Sectors})$$

$$K_{ds} = \sum_{t \in \mathbf{T}_{>1}} \sum_{v \in V} \mathbb{I}(\mathbf{SEC}^{-1}(v, t-1) \neq \mathbf{SEC}^{-1}(v, t)) \quad (\text{Navpoint-sector changes})$$

$$K_r = \sum_{f \in \mathbf{F}} r_f \quad (\text{Reroutes/delays})$$

$$K_c = \sum_{t \in \mathbf{T}_{\geq 0}} \sum_{v \in V} \mathbb{I}(\mathbf{sec}^{-1}(v, t) \neq \mathbf{SEC}^{-1}(v, t)) \quad (\text{Reconfigurations})$$

where $\mathbf{SEC}^{-1}(v, t)$ returns the representative sector index i containing vertex v at time t .

4 New Decomposition Heuristics

Although some large-scale ATFM models can be solved exactly by exploiting inherent structures [3], we expect that integrating DAC introduces dense coupling constraints. These constraints likely create structural difficulties that prevent scaling exact solution approaches to medium- and large-scale instances. Moreover, the required sequential bounding for the five-level lexicographic objective severely degrades solver performance. Consequently, exact methods suffer from combinatorial explosion, rendering multi-objective exact solving computationally intractable even for relatively small instances [5]. To ensure operational feasibility, we propose a heuristic approach: $ASPaeroFlow^2$. $ASPaeroFlow$ decomposes the

² GitHub Link: <https://github.com/alex14123/ASPaeroFlow-Optimizer>

■ **Algorithm 4.1** ASPaeroFlow.

Data: Instance $\mathcal{I}$, Encoding E , Parameters $\mathcal{P}$

```

1  $\mathbf{M} \leftarrow \text{convertInstance}(\mathcal{I})$  ;       $\text{overloads} \leftarrow \text{computeOverloads}(\mathbf{M})$  ;
2  $\mathcal{P}' \leftarrow \text{copy}(\mathcal{P})$  ;
3 while  $\text{overloads} > 0$  do
4    $I \leftarrow \text{decomposeInstance}(\mathbf{M}, \mathcal{P}')$  ;
5    $\mathbf{M}' \leftarrow \text{LocalOptimization}(I, E)$  ;       $\text{overloads}' \leftarrow \text{computeOverloads}(\mathbf{M}')$  ;
6   if  $\text{overloads}' < \text{overloads}$  then
7      $\mathbf{M} \leftarrow \mathbf{M}'$  ;       $\text{overloads} \leftarrow \text{overloads}'$  ;
8      $\mathcal{P}' \leftarrow \text{copy}(\mathcal{P})$  ;
9   else if  $\text{exhausted}(\mathcal{P}')$  then
10    return  $(\mathbf{M}, \text{False})$  ;      // Termination with residual overload
11  else
12     $\mathcal{P}' \leftarrow \text{adjustParameters}(\mathcal{P}')$  ;
13 return  $(\mathbf{M}, \text{True})$  ;      // Successful termination

```

problem around the next overloaded sector, building a bounded local subproblem containing only the most relevant flights and the overloaded sector. Specifically, we cap (i) the number of considered flights ($f_{\max}$), (ii) the delay window per flight ($d_{\max}$), (iii) the number of alternative routes per flight ($r_{\max}$), and (iv) the number of partition options for the overloaded sector ($p_{\max}$); see also Table 1. An exact solver then selects the best local solution under our lexicographic objective, and the procedure iterates until all overloads are resolved. The chosen local bounds balance search-space exploration with solver runtimes. Empirically, increasing the flight limit ($f_{\max} > 2$) or the partitioning limit ($p_{\max} > 2$) drastically increases solving times. Conversely, the routing ($r_{\max}$) and delay ($d_{\max}$) bounds are computationally more forgiving; the chosen parameters were determined via empirical parameter tuning to ensure rapid local convergence while maintaining high solution quality.

4.1 Optimization Loop

Algorithm 4.1 shows the overall algorithm. In Lines (1)–(2) we do pre-processing steps, by converting the instance to the internal matrix representation $\mathbf{M}$ and computing the overloads. We consider as overloads the sum of exceedances: $K_o = \sum_{t \in T, i \in V} \max\{0, q(i, t) - \text{cap}(i, t)\}$. The algorithm iteratively solves overloads until $K_o = 0$ (Line (3)). The decomposed instance I is computed by instance space decomposition (Line (4) – Algorithm 4.2), which is then given with the encoding E to the local optimization procedure (Line (5)). The local optimization procedure computes a new candidate solution $\mathbf{M}'$. A *correct local optimization encoding* selects among its available choices the solution that reduces the magnitude of overload the most, followed by adhering to the criteria in Definition 13. The candidate solution $\mathbf{M}'$ is accepted if it has fewer overloads (Lines (6)–(8)). Otherwise, the parameters of the decomposition are adapted to allow for other solutions to be explored to ensure termination (Line (12)).

4.2 Instance Space Decomposition

Algorithm 4.2 shows our instance decomposition approach. We reduce the search space, by limiting the number of investigated flights $f_{\max}$, the delay $d_{\max}$, the alternative routes $r_{\max}$, and the possible DAC partitions $p_{\max}$ (Line (1)). Line (2) obtains the time t and sector s of the first overloaded sector (lexicographically time and sector-wise). In Line (3) we limit the search space to $f_{\max}$ flights. The next part in Lines (4)–(12) is to generate the instance for

■ **Algorithm 4.2** Instance Space Decomposition.

Data: Converted Instance $\mathbf{M}$, Parameters $\mathcal{P}$

```

1  $f_{\max}, d_{\max}, r_{\max}, p_{\max} \leftarrow \mathcal{P}$ ;
2  $s, t \leftarrow \text{computeFirstOverload}(\mathbf{M})$ ;
3  $F \leftarrow \text{flightsInOverload}(\mathbf{M}, s, t)$        $F \leftarrow \text{sort}(F)$        $F \leftarrow F[: f_{\max}]$ ;
4  $I_F \leftarrow \emptyset$ ;
5 for  $f$  in  $F$  do
6    $\text{config} \leftarrow 0$ ;
7    $R \leftarrow \text{bestRoutes}(f, G, r_{\max})$ ;
8   for  $d$  in  $\text{range}(d_{\max} + 1)$  do
9     for  $r$  in  $R$  do
10       $f' \leftarrow \text{computeNavpointFlightPlan}(f, d, r)$ ;
11       $I_F \leftarrow I_F \cup \{f'\}$ ;
12       $\text{config} \leftarrow \text{config} + 1$ ;
13  $I_S \leftarrow \emptyset$ ;
14 for  $p$  in  $\text{range}(1, p_{\max} + 1)$  do
15    $\text{partition}_p \leftarrow \text{computePartition}(s, p, G)$ ;
16    $I_S \leftarrow I_S \cup \{\text{partition}_p\}$ ;
17  $I \leftarrow I_F \cup I_S$ ;
18 return  $I$ ;

```

each of the $f_{\max}$ flights, by computing the alternatives $r_{\max}$ and considering the maximum delay $d_{\max}$. This is followed by the generation of the instance for DAC, where we consider $p_{\max}$ partitions (Lines (13)–(16)), ranging from the current configuration $p = 1$ to a partition of the sector s into $p_{\max}$ parts. Finally, we combine the instance in Line (17) and return it.

Reroute Generation. We compute the $r_{\max}$ routes (Algorithm 4.2 in Line 7), using an iterative algorithm that computes the shortest routes on G between departure and destination airports. On each iteration the current shortest path p is penalized and a diverse set of routes is ensured by only accepting a path p if its Jaccard similarity to any other already considered path p' is small enough: $\frac{p' \cap p}{p' \cup p} < j_m$, where $j_m = 0.6$. Let $w(p)$ be the path weight, then each edge e of $e \in p$ is penalized by $w_{t+1}(e) \leftarrow w_t(e) + 0.1 \frac{w(p)}{|p|}$, where $w_0(e) = d(e)$.

Alternative Airspace Configurations. The $p_{\max}$ graph partitions are computed in Algorithm 4.2 in Line (15) by an iterative breadth-first search algorithm. We partition a sector s into p sectors ($\{s_1, \dots, s_p\} = S$) of approximate size $\frac{|sec(s, t)|}{p}$ iteratively. Assuming we generated $|S| = i - 1$ sectors so far, we add a new sector s_i via two steps: first, we select a $seed \in sec(s, t) \setminus (\cup_{s_j \in S} sec(s_j, t))$ vertex and add it to the new sector $s_i = \{seed\}$. Second, we add vertices $v \in sec(s, t) \setminus (\cup_{s_j \in S} sec(s_j, t))$ to s_i in a breadth-first manner (ensuring connection of s_i) until $|sec(s_i, t)| > \frac{|sec(s, t)|}{p}$; then we add s_i to S .

Adjust Parameters. If search stalls, `adjustParameters` expands the temporal horizon using a rolling delay (+5 timesteps). After 10 non-improvement steps, the flight limit reduces to $f_{\max} = 1$. Assuming any flight can be routed in isolation, the algorithm *terminates* and returns $\mathbf{M}$ (Line 10) if $f_{\max} = 1$, the subsequent step yields no improvement, and the single flight f remains unroutable due to absolute capacity limits (e.g., zero-capacity sectors).

4.3 Correctness

We provide a brief argument for correctness of the algorithm. Let $\mathcal{I}$ be an instance of the optimization problem and let $\mathcal{S}$ be the set of solutions. For the following proofs, we take the operational assumption that all filed flights can be flown in principle; meaning that

$\forall v \in V : \text{cap}(v, 0) \geq 1$. We refer to this as *operational feasibility*. Operational feasibility guarantees that a single flight from an overloaded sector can always be delayed to reduce the magnitude of overload – consequently, if delaying the single flight does not lead to a reduction in overload magnitude, the algorithm can safely terminate.

► **Theorem 14.** *Let $\mathcal{I}$ be an instance of the optimization problem and let the operational feasibility assumption hold. Then, for any correct encoding E , if ASPaeroFlow (Algorithm 4.1) produces an output, its output is a valid solution.*

Proof (Sketch). The Algorithm 4.1 returns the solution $\mathbf{M}$ in Line 13, or in Line 10. We prove that with the operational feasibility assumption, if Algorithm 4.1 terminates, it terminates by returning the solution $\mathbf{M}$ in Line 13: Towards a contradiction assume Algorithm 4.1 terminates prematurely on exhaustion (Line 10). For this, it must be the case that for at least 10 subsequent steps the algorithm did not experience a reduction of overload, although it increased the time window start to t_s and reduced the number of considered aircraft to $f_{\max} = 1$. Let the considered flight for optimization be $f = (id, tr)$ for sector $i \in V$ and time $t' \in \mathbf{T}$. Further, let f' be the flight in the current solution $\mathbf{M}$ with the maximum actual arrival delay $t_{f'}^a$. Then, to exit prematurely on exhaustion in Line 10, flight f cannot be scheduled after flight f' , so it must hold that whenever $t_s > t_{f'}^a$, no improvement is found. However, such an improvement is found by local optimization due to the operational assumption: adjust tr to tr_s , by keeping the path, but starting at t_s , the overload of i decreases and no increase in overload of any other sector is experienced, as by the operational feasibility assumption $\forall (v, t) \in tr_s : \text{cap}(v, t) \geq q(v, t) = 1$. This is a contradiction to the assumption.

It remains to argue that the returned $\mathbf{M}$ from Line 13 is indeed a valid solution (Def. 11). First, there are no overloads remaining in $\mathbf{M}$ due to Line 3. Further, **A** and **SEC** are valid. **A** is valid, as (a) the instance space decomposition generates valid trajectories and ensures that for each considered flight its subsequent flights are also part of the instance passed to local optimization and thereby cannot overlap (Lines (5)–(12) of Algorithm 4.2) and (b) local optimization ensures that exactly one trajectory is selected for each flight. Further, **SEC** is valid as bounded connected partitions introduced by Lines (13)–(16) of Algorithm 4.2, combined with local optimization ensure valid partitions. ◀

► **Observation 15.** *Let $\mathcal{I}$ be an instance of the optimization problem. For any correct encoding E without assuming operational feasibility, Algorithm 4.1 may return an incorrect solution.*

Proof (Sketch). When the operational feasibility assumption is violated, it is not guaranteed that a flight f can actually occur; however, it can be the case that ASPaeroFlow terminates prematurely on exhaustion (Line 10) in Algorithm 4.1, thereby producing an invalid solution $\mathbf{S}$, although in principle a solution $\mathbf{S}'$ exists by Def. 11: this can happen by (1) ASPaeroFlow not considering routes, which are outside of the bounded rerouting $r_{\max}$ options, or (2) ASPaeroFlow not considering sector configurations, which are not evaluated due to bounded sectorization $p_{\max}$. ◀

► **Theorem 16.** *Let $\mathcal{I}$ be an instance of the optimization problem. Then, for any correct encoding E , ASPaeroFlow (Algorithm 4.1) terminates.*

Proof (Sketch). In each iteration of Line 3 of Algorithm 4.1 either the overload strictly decreases (a new solution is accepted as the current one and the parameter bounds are reset to their initial values), or the parameter bounds are expanded via *adjustParameters*. If the total

overload reaches 0, the algorithm terminates successfully (Line 13). Otherwise, the algorithm ensures $f_{\max} = 1$ by setting it after 10 non-improvement steps via *adjustParameters*. The *exhausted* condition then becomes true if the earliest starting time t_f of the single ($f_{\max} = 1$) considered flight f is strictly later than the arrival time t_f^a , of any other flight f' : $t_f > t_f^a$, (and no improvement occurs). When exhausted, the algorithm terminates with a residual overload (Line 10). ◀

4.4 Local Optimization via ASP

We proceed to describe an encoding in ASP. Conceptually, ASP is similar to Boolean satisfiability, but allows for more natural problem modeling, rapid integration of changed problem definitions, and directly provides minimization statements.

For brevity, the detailed generation of facts, trajectory guesses, and structural mappings are deferred to the appendix. Here, we focus on the optimization objectives. We treat the minimization of the magnitude of overload as the primary objective – as the ASP model operates locally and cannot solve global overloads. Besides that, we minimize lexicographically according to the ATFCM^{OPT} problem. Lines (1) and (2) describe minimization of overloads, while Line (3) describes minimization of arrival delay. By construction, Line (4) minimizes both the number of sectors and navpoint-sector changes. Finally, Line (5) minimizes reroutes/delays and Line (6) minimizes divergences from *sec*.

```

1  $\leftarrow$  overload(X,T,OVER). [OVER@10,X,T]
2  $\leftarrow$  chc(CONFIG), config(CONFIG,TOTAL_OVER). [TOTAL_OVER@10,CONFIG]
3  $\leftarrow$  arrival_delay(ID,DIFF). [DIFF@9,ID]
4  $\leftarrow$  chc(CONFIG), config_number_sectors(CONFIG,NUMBER). [NUMBER@8,CONFIG]
5  $\leftarrow$  reroute(ID). [1@7,ID]
6  $\leftarrow$  chc(CONFIG). [CONFIG@6,CONFIG]
```

► **Lemma 17.** *The ASP encoding is a correct local optimization encoding.*

Proof (Sketch). We argue for two properties: (a) the found solution $\mathbf{M}'$ locally adheres to the specifications of an ATFCM solution (Def. 11) except the overload; and (b) the found solution $\mathbf{M}'$ is locally optimal s.t. it reduces overloads while adhering to side constraints (Def. 13). (a): To prevent invalid double assignments or double sector splits, exactly one path ($\text{chtrj}(\text{ID},\text{P})$) and one configuration ($\text{chc}(\text{CONFIG})$) is selected. (b): the primary optimization criterion is to reduce K_o , as finding a globally $K_o = 0$ solution by local optimization cannot be guaranteed; the objective combines local effects computed in ASP ($\text{overload}(\text{X},\text{T},\text{OVER})$) with effects computed by Algorithm 4.2 ($\text{config}(\text{CONFIG},\text{TOTAL_OVER})$). The other optimization criteria adhere to the specification of Def. 13: for $[\text{NUMBER@8},\text{CONFIG}]$ we use the fact that for any two configurations produced by instance space decomposition, if the number of resulting splits is lower ($K_{\#s}$), the number of sector changes is as well (K_{ds}). The number of reroutes K_r directly translates from Def. 13; the number of reconfigurations is minimized by the fact that $\text{chc}(\text{CONFIG})$ for $\text{chc}(0)$ has 0 introduced config changes. ◀

5 Benchmarks and Experiments

To demonstrate the operational feasibility of our approach, we design our computational experiments to evaluate three primary dimensions: (i) the solution quality of our decomposition heuristic relative to exact models, (ii) the scalability of the approach compared to related heuristic approaches across progressively larger, real-world-sized scenarios, and (iii) the effects of integrating DAC into ATFM via an ablation study. To quantify these dimensions, we extract the following evaluation metrics: computational performance (execution time and peak RAM usage) and model metrics (unresolved overloads, K_d , $K_{\#s}$, K_{ds} , K_r , K_c).

■ **Table 2** Compared variants and their action bounds. Global approach search space $\approx d_{\max}^{|F|} \cdot r_{\max}^{|F|} \cdot p_{\max}^{|V|}$, local search space (per iteration, assuming decomposition $f_{\max} = 2, |V| = 1$) $\approx d_{\max}^2 \cdot r_{\max}^2 \cdot p_{\max}$.

Variant	$p_{\max}$	$r_{\max}$	$d_{\max}$	Notes
Basic Baseline				
Initial	0	0	0	Result if no action is taken (naive baseline).
Global (Exact) Approaches [5]				
ASP-P	4	4	expanded	Global decisions; partial DAC; partial Rerouting, partial delaying; delays expanded up to feasibility; initially $d_{\max} \approx T_{gran} \cdot 24$
MIP	1	2	expanded	Global decisions; partial rerouting; small delay bound iteratively expanded up to feasibility; initially $d_{\max} \approx T_{gran} \cdot 24$
Local Heuristic Approaches				
ASPaeroFlow _{r,d}	2	3	5	Default configuration.
ASPaeroFlow _{$\neg r, d_p$}	2	1	1	Only sectorization; if stuck repeatedly delaying possible.
ASPaeroFlow _{$\neg r, d$}	2	1	5	Sectorization and delay (no rerouting).
ASPaeroFlow _{r, d_p}	2	3	1	Sectorization and rerouting; if stuck repeatedly delaying possible.
ASPaeroFlow _{seq}	2	3	5	Sequential execution of DAC and (full) ATFM.
ATFM _{r, d}	1	3	5	Flow-only ATFM (no DAC).
ATFM _{r, d_p}	1	3	1	Rerouting-only baseline (no delay); if stuck repeatedly delaying possible.
ATFM _{$\neg r, d$}	1	1	5	Delay-only ATFM (no rerouting).
CASA	1	1	5	Single flight delaying (decomposition $f_{\max} = 1$).

5.1 Benchmarked Variants

We benchmarked 12 variants. One (*Initial*) is our baseline, if no action is performed. The *exact* solvers comprise *ASP-P*, which is *ASP _{r_p, d_p, s_p}* – a bounded action exact model which balances time to first and time to best solution – and *MIP*, a mapping of state-of-the-art ATFM MIP formulations to the joint ATFCM model (meaning it selects optimal bounded reroutings and delays, but has no sectorization actions) [5]. *ASPaeroFlow* comprises the remaining nine variants under various parameter configurations. Variants executing joint optimization of ATFM and DAC are denoted as *ASPaeroFlow*, while restricted variants optimizing only ATFM are denoted as *ATFM*. Subscripts $r \in \{r, \neg r\}$ and $d \in \{d, d_p\}$ indicate whether rerouting or delaying is permitted. To simulate operational baselines, we include *CASA*, a greedy “First Planned - First Served” simulation restricting optimization to single-flight delays ($f_{\max} = 1$). Furthermore, the *ASPaeroFlow_{seq}* *sequential* variant models standard collaborative decision-making by separating the pipeline: optimizing sectorization first, followed by a *ATFM _{r, d}* flow optimization.

5.2 Experiment Scenarios and Setup

We conducted benchmarks on three sets of instances: *Small* [5], *SOTA* [2], and *Large*. Table 3 shows an overview of the considered instances.

The Small scenario. Regarding traffic density, we vary the number of flights $|F| \in \{10, 20, \dots, 100\}$. For each $|F|$, we evaluate three distinct random seeds. En-route vertices possess a fixed capacity of 1, while airport capacities are assumed to be unconstrained. Time is partitioned into 1-hour timesteps ($T_{gran} = 1$).

SOTA Instances. The SOTA instances are standard benchmarks in the literature [7, 2]. Instances *c01*, *c02*, and *c03* are randomized scenarios designed to test comprehensive ATFCM features. BLO instances are similarly randomized, where mBLO represents specific

10:12 Decomposition Heuristics for Joint Air Traffic Flow & Capacity Management

■ **Table 3** Instance Description: $|V|$ is the sum of navpoint and airport vertices, $|S|$ is the maximum sector size, C the maximum capacity (or range of maximum capacity), TG the time granularity, $|F|$ the number (range) of flights, and $|A|$ the number (range) of aircraft.

Small Instances [5]								
Instance	Navpoints	Airports	$ V $	$ S $	C	T_{gran}	$ F $	$ A $
EA-3x3	9	7	16	3	1	1	10 to 100	6 to 41
CE-5x5	25	7	32	5	1	1	10 to 100	6 to 48
IND-4x10	40	8	48	6	1	1	10 to 100	6 to 43
USA-7x7	49	8	57	7	1	1	10 to 100	5 to 50
EUR-10x10	100	17	117	10	1	1	10 to 100	7 to 53
SOTA Instances [2]								
BLO1	107	40	147	8	34	4	1054	568
BLO2	1172	20	1192	59	88	4	3196	3196
BLO3	2707	30	2737	135	130	4	6475	6475
IC	5932	47	5979	118	25	4	3157	3157
c01	107	40	147	8	34	4	1054	568
c02	180	40	220	7	30	4	1330	671
c03	147	45	192	8	43	4	1545	788
mBLO1	100	4	104	9	67	4	812	812
mBLO3	2707	30	2737	135	130	4	6475	6475
Large Instances								
$P \cdot 1.0 \rightarrow P \cdot 0.1$								
CE-G	49	7	56	11	3 to 1075	4	1000 to 31622	202 to 5978
USA-G	200	8	208	20	5 to 1615	4	1000 to 31622	272 to 8647
EUR-G	800	17	817	85	5 to 1719	4	1000 to 31622	279 to 7506
EA-G	1600	7	1607	34	4 to 2015	4	1000 to 31622	322 to 14007
CE	627	10	637	35	11 to 3716	4	1000 to 31622	594 to 19069
DACH	1283	324	1607	50	3 to 1065	4	1000 to 31622	587 to 18699
EUR	8479	1588	10067	50	2 to 719	4	1000 to 31622	886 to 28020
USA	18101	1509	19610	50	2 to 1042	4	1000 to 31622	976 to 30586

modifications (we do not possess mBLO2). The *IC* instance reflects an industry case, capturing a typical day of flight schedules and paths for a segment of the European ATM network. Following related literature, time is partitioned into 15-minute intervals ($T_{gran} = 4$).

Large instances. The large scenario considers larger graphs with more flights per day. The number of flights is $|F| \in \{1000, 1778, 3162, 5623, 10000, 17782, 31622\}$ flights, where for each $|F|$ we consider three seeds (11904657, 150699, 42), and capacity is relative to nominal capacity (optimal weather/staffing). We scale atomic sector capacities uniformly from 100% down to 10% in 10% decrements ($P \cdot 1.0 \rightarrow P \cdot 0.1$). We set $T_{gran} = 4$ (as for SOTA).

Technical Setup. We implemented the heuristics in Python (3.13.7), and used Clingo (5.8.0) for ASP and Gurobi (13.0.0 build v13.0.0rc1) for MIP computation. All experiments were conducted on the Copperhead CPU Cluster at TU Wien, with 2x Intel Xeon Silver 4314 CPUs, 512GB RAM, and Ubuntu 22.04 (Kernel 5.15.0-131-generic). We considered a TIMEOUT of 1800s and a MEMOUT of 35GB.

5.3 Results and Discussion

Heuristic vs. Exact Approaches. We show the number of solved instances ($\#S$) and tournament wins ($\#W$) in Table 4. While for the small experiments, the ASP-P variant achieves strong results ($\#W : 83$), the ASPaeroFlow $_{r,d_p}$ variant achieves competitive results ($\#W : 51$), while CASA wins only $\#W : 4$. However, this is mostly due to the fact that even for small instances, the ASP-P approach runs into computational solving bottlenecks – even on the small instances, ASP-P is not able to solve all instances in the time limit. Comparing these methods on instances, which ASP-P solves to 0 overload, ASPaeroFlow $_{r,d_p}$ achieves a lower arrival delay (an average value of 25.62 ± 2.25) compared to CASA (an average value of 64.83 ± 10.19). MIP achieves worse results compared to ASP for solved small instances

■ **Table 4** Tournament wins $\#W$ (lexicographic: $K_o \succ K_d \succ K_{\#s} \succ K_{ds} \succ K_r \succ K_c$) and number of solved $\#S$ instances ($K_o = 0$) by scenario.

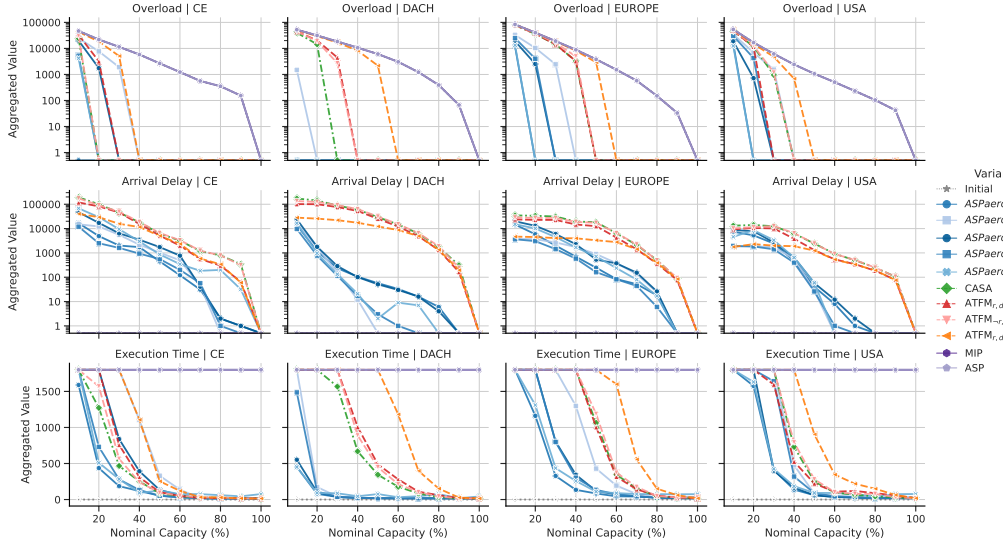
Scenario	#I	ASPaeroFlow _{r,d} #W	ASPaeroFlow _{r,d} #S	ASPaeroFlow _{seq} #W	ASPaeroFlow _{seq} #S	ASPaeroFlow _{r,d_p} #W	ASPaeroFlow _{r,d_p} #S	ASPaeroFlow _{~r,d} #W	ASPaeroFlow _{~r,d} #S	Draws #D
Large Instances (PCAP > 0.5)	840	46	834	343	840	127	831	1	826	250
Large Instances (PCAP ≤ 0.5)	840	211	674	213	685	290	605	6	631	82
Small Instances	200	15	200	13	200	51	200	0	200	8
SOTA Instances	9	0	9	1	9	1	7	0	9	2
Overall	1889	272	1717	570	1734	469	1643	7	1666	342
Scenario	#I	ASPaeroFlow _{~r,d_p} #W	ASPaeroFlow _{~r,d_p} #S	ATFM _{r,d} #W	ATFM _{r,d} #S	ATFM _{r,d_p} #W	ATFM _{r,d_p} #S	ATFM _{~r,d} #W	ATFM _{~r,d} #S	Draws #D
Large Instances (PCAP > 0.5)	840	52	815	2	824	4	754	0	820	250
Large Instances (PCAP ≤ 0.5)	840	37	525	0	407	0	205	0	411	82
Small Instances	200	6	200	0	200	1	200	0	200	8
SOTA Instances	9	0	9	3	9	2	7	0	8	2
Overall	1889	95	1547	5	1440	7	1166	0	1439	342
Scenario	#I	CASA #W	CASA #S	Initial #W	Initial #S	MIP #W	MIP #S	ASP-P #W	ASP-P #S	Draws #D
Large Instances (PCAP > 0.5)	840	0	824	0	168	15	180	0	112	250
Large Instances (PCAP ≤ 0.5)	840	0	448	0	0	1	1	0	3	82
Small Instances	200	4	200	0	1	19	82	83	92	8
SOTA Instances	9	0	9	0	0	0	0	0	0	2
Overall	1889	4	1481	0	169	35	263	83	207	342

■ **Table 5** Ablation study aggregated over large scenarios, showing absolute mean values and SEMs. We study the effects of enabling/disabling rerouting (r), delaying (d), and restructuring (s).

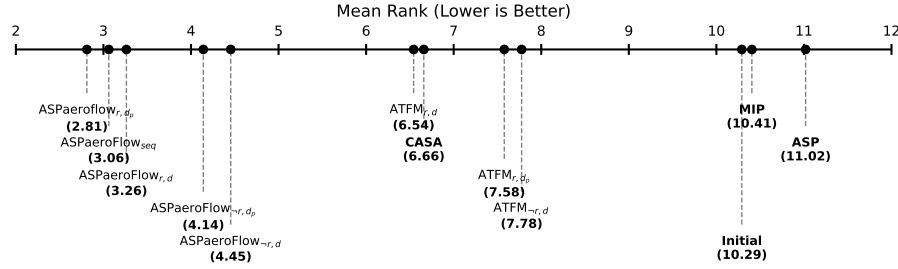
Option	Overload [#]	Arrival Delay [h]	Sector-Number [#]	Sector-Diff [#]	Reroute [#]	Reconfig [#]
r	9696.23 ± 424.57	20410.69 ± 662.55	90925.69 ± 1729.46	159.90 ± 6.00	3250.78 ± 66.75	16016.11 ± 810.76
$\neg r$	10526.43 ± 432.70	18834.70 ± 741.47	84098.67 ± 1629.73	144.91 ± 5.36	2929.04 ± 66.10	15651.06 ± 760.66
d	8995.97 ± 416.58	32960.03 ± 940.81	94563.00 ± 1806.79	162.97 ± 6.27	3465.57 ± 67.44	17611.50 ± 896.61
d_p	11226.69 ± 440.04	6285.36 ± 225.54	80461.35 ± 1539.89	141.84 ± 5.04	2714.25 ± 65.13	14055.68 ± 656.57
s	5269.25 ± 306.63	5192.79 ± 267.19	85727.03 ± 1647.63	304.81 ± 7.61	5035.24 ± 83.87	31667.18 ± 1077.64
$\neg s$	14953.42 ± 516.28	34052.61 ± 924.97	89297.32 ± 1713.15	0.00 ± 0.00	1144.58 ± 25.92	0.00 ± 0.00

on arrival delay (an average value of 5.28 ± 0.52), but is overall able to solve slightly more instances ($\#S : 263$ vs. $\#S : 207$). We emphasize that this should not be interpreted as ASP being in general a better solving technology than MIP; rather, the two approaches operate in fundamentally different feasible decision spaces. Specifically, we attribute the difference in results to ASP-P’s ability to utilize DAC actions, where ASP-P uses on average $K_{ds} = 188.00 \pm 8.60$, compared to 0 for MIP. A similar behavior can also be seen when comparing the results of MIP and ASP-P for EA-3x3 and EUR-10x10. EA-3x3 is a tiny graph with effectively no possibility for DAC ($|V| = 16$ with $\max |S| = 3$), whereas for EUR-10x10 the possibilities for DAC are plenty ($|V| = 117$ with $\max |S| = 10$). As expected, MIP performs better on EA-3x3, whereas ASP-P performs better on EUR-10x10. In the Appendix Table 6 we show the detailed results for the small and SOTA scenarios, for 5 selected variants: Baseline Initial, the best-performing heuristic variant for the small experiments *ASPaeroFlow_{r,d_p}*, CASA, and the exact methods MIP and ASP-P. More detailed results are shown in the supplementary material.

Scaling and Heuristic Approaches. In Figure 3 we show the scaling results comparing nominal capacity. The full scaling results comparing graph and flights size are available in the supplementary material. For the navpoint and flights scaling results, we compare 4 heuristic approaches on 10% nominal capacity (more in the supplementary material), where *ASPaeroFlow_{r,d}* and *ASPaeroFlow_{seq}* are able to solve more instances with lower arrival delay than CASA/ATFM_{r,d}. Interestingly, just comparing these results we can see that simultaneous optimization of *ASPaeroFlow_{r,d}* achieves competitive results compared to *ASPaeroFlow_{seq}*, as it is able to solve more instances on sizes $|V| \in \{CE - G, USA - G, CE\}$, whereas *ASPaeroFlow_{seq}* achieves better results on $|V| \in \{EUR - G, DACH\}$. This behavior can also be seen in Figure 3, where nominal capacity alterations are shown for large graphs



■ **Figure 3** Nominal capacity scaling results on the four real-world topography large instances, comparing heuristic and exact methods. Exact methods run into time-/memouts in this setting.



■ **Figure 4** Significance graph showing the results of the post-hoc pair-wise Wilcoxon tests, where ranks indicate a hierarchy. All differences are statistically significant ($p < 0.05$).

($|V| \in \{CE, DACH, EUR, USA\}$): the ASPaeroFlow variants use approximately an equal execution time, whereas CASA/ATFM-variants are slower. Furthermore, we can observe that the results are instance-dependent and while ASPaeroFlow-variants are able to solve instances with nominal capacity $\geq 20\%$, the methods relying on ATFM-measures are only able to solve instances $\geq 40\%$. On the SOTA scenarios, we show the strongest results in Table 6. ASPaeroFlow_{r,d,p}, ASPaeroFlow_{seq}, ATFM_{r,d,p}, and ATFM_{r,d} achieve the strongest results. The strong results for the ATFM only methods can be explained by studying the differences between the instances. While the generated instances BLOX, c0X, and mBLOX have little to no possibility for DAC, the industrial instance IC allows for plenty of reconfiguration. Therefore, the results of BLOX, c0X, and mBLOX, are similar, where the ATFM methods have a slight edge as they are (implicitly) more likely to consider reroutes – which also explains results of negative delay, which happen as for the SOTA instances the filed path is not necessarily the optimal one. Comparisons to SOTA work besides the paper introducing ASP-P and MIP [5] however, are hardly useful, due to differences in the model stemming from dynamic computation, (slightly) different definitions of demand/capacity measurement, objective functions, and time.

Statistical Significance. To evaluate the statistical significance of the algorithmic differences, we first performed a non-parametric Friedman test. The test indicated a statistically significant difference in the lexicographic mean ranks across the 12 variants ($\chi_F^2(11, N = 1889) = 15459.90, p < 0.001$). Consequently, we conducted post-hoc pairwise Wilcoxon signed-rank tests, applying the Holm-Bonferroni correction to adjust for multiple comparisons. The results demonstrate that all pairwise differences are statistically significant ($p < 0.001$), establishing a strict performance hierarchy of $ASPaeroFlow_{r,d_p}$, $ASPaeroFlow_{seq}$, $ASPaeroFlow_{r,d}$, followed by the others. The final ordering of the variants, based on their mean ranks, is depicted in Figure 4.

Sequential vs. Simultaneous Optimization. We observe that simultaneous $ASPaeroFlow$ is highly competitive to $ASPaeroFlow_{seq}$. However, relying solely on aggregate statistical tests or scaling results across the entire benchmark yields inconclusive results: while all simultaneous variants together solve and win more than $ASPaeroFlow_{seq}$, no single simultaneous variant beats $ASPaeroFlow_{seq}$ (Table 4). This occurs because sequential optimization possesses two inherent advantages: first, if an instance can be solved solely via DAC, sequential optimization achieves superior results since $K_d = 0$; second, the smaller search space per step permits more search steps compared to simultaneous optimization. Conversely, a major deficit exists: if pure DAC fails to eliminate overload, sequential optimization defaults to $ATFM_{r,d}$, which performs worse than simultaneous optimization. We compare in more detail the variants $ASPaeroFlow_{r,d}$ and $ASPaeroFlow_{seq}$: evaluating all instances ($n = 1889$) on the two methods, $ASPaeroFlow_{r,d}$ achieves $\#W : 864$ and $\#S : 1717$, while $ASPaeroFlow_{seq}$ achieves similar results ($\#W : 843, \#S : 1734$); with 182 draws. However, isolating the subset of instances where $ASPaeroFlow_{seq}$ fails to solve via DAC alone ($n = 1318$), $ASPaeroFlow_{r,d}$ demonstrates a clear advantage with $\#W : 864$ ($\#S : 1154$) compared to $\#W : 441$ ($\#S : 1171$) for $ASPaeroFlow_{seq}$; with 13 draws. This divergence is reflected in the statistical analysis. Comparing only these two variants across the full dataset ($n = 1889$), the Wilcoxon signed-rank test yields mean ranks of 1.49 for $ASPaeroFlow_{r,d}$ and 1.51 for $ASPaeroFlow_{seq}$ ($p = 0.61$, not significant). Note that the loss of significance compared to Figure 4 stems exclusively from reducing the test space from $k = 12$ to $k = 2$; without the other variants skewing the global rank distribution, the high number of ties between these two highly competitive variants becomes statistically apparent. Interestingly, when applying the test to the dataset where $ASPaeroFlow_{seq}$ fails to solve via DAC alone ($n = 1318$), the mean ranks shift to 1.34 for $ASPaeroFlow_{r,d}$ and 1.66 for $ASPaeroFlow_{seq}$, yielding a statistically significant difference ($p < 0.001$).

Ablation Study. Table 5 evaluates the impact of enabling rerouting (r), delaying (d), and restructuring (s). Each feature reduces overloads, however for $r/\neg r$ the results are strictly speaking inconclusive due to overlapping SEMs. Restructuring yields the largest reduction in both overload and arrival delay. While the heuristic optimizes locally, a sector split at t_1 persists, providing global temporal benefits by resolving future overloads at $t_2 > t_1$. Conversely, rerouting and delay effects remain temporally localized. For multi-leg flights, local delays may fail to resolve subsequent conflicts automatically and risk propagating downstream, inducing secondary overloads.

6 Conclusion

$ASPaeroFlow$ demonstrates that bounded instance-space decomposition provides a computationally viable heuristic for joint ATFCM, scaling exact ASP to industry-sized instances. Our ablation study reveals that DAC is the primary driver for overload reduction because

sector partitioning provides global temporal benefits, whereas flow measures risk downstream propagation of overloads. Furthermore, simultaneous optimization outperforms sequential approaches when DAC alone fails. Future work will address the open challenges of stochastic disruptions and the integration of *Explainable AI (XAI)* to extract rule-based justifications for automated scheduling decisions.

References

- 1 Dirk Abels, Julian Jordi, Max Ostrowski, Torsten Schaub, Ambra Toletti, and Philipp Wanko. Train Scheduling with Hybrid Answer Set Programming. *TPLP*, 21(3):317–347, 2021. doi:10.1017/S1471068420000046.
- 2 Alba Agustín, Antonio Alonso-Ayuso, Laureano F. Escudero, and Celeste Pizarro. On air traffic flow management with rerouting. Part I: Deterministic case. *Eur. J. Oper. Res.*, 219(1):156–166, 2012. doi:10.1016/J.EJOR.2011.12.021.
- 3 Hamsa Balakrishnan and Bala G. Chandran. Optimal large-scale air traffic flow management. Technical report, Massachusetts Institute of Technology, 2014.
- 4 Lucia Balázová, Richard Comploi-Taupe, Susana Hahn, Nicolas Rühling, and Gottfried Schenner. Smart expansion techniques for ASP-based interactive configuration. *TPLP*, 25(4):473–488, 2025. doi:10.1017/S147106842510029X.
- 5 Alexander Beiser, Markus Hecher, Nysret Musliu, and Stefan Woltran. Joint air traffic flow and capacity management via answer set programming. In *18th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR26)*, 2026.
- 6 Dimitris Bertsimas and Shubham Gupta. Fairness and collaboration in network air traffic flow management: An optimization approach. *Transp. Sci.*, 50(1):57–76, 2016. doi:10.1287/TRSC.2014.0567.
- 7 Dimitris Bertsimas, Guglielmo Lulli, and Amedeo Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59:211–227, 2011. doi:10.1287/OPRE.1100.0899.
- 8 Dimitris Bertsimas and Sarah Patterson. The Air Traffic Flow Management Problem with Enroute Capacities. *Oper. Res.*, 46(3):406–422, 1998. doi:10.1287/OPRE.46.3.406.
- 9 Tatjana Bolić, Lorenzo Castelli, Luca Corolli, and Desirée Rigonat. Reducing ATFM delays through strategic flight planning. *Transp. Res. Part E: Logist. Transp. Rev.*, 98:42–59, 2017.
- 10 Razvan Bucuroiu and Stephanie Vincent. European network operations plan 2025/2026 - 2029. EUROCONTROL, 2025.
- 11 Aitichya Chandra, Sayan Hazra, and Ashish Verma. The integration of En route flow optimization, complex network clustering, and rule-based approach to airspace sub-sectorization for enhanced air traffic monitoring. *J. Air Transp. Res. Soc.*, 3:100036, 2024.
- 12 Richard Comploi-Taupe, Andreas Falkner, Susana Hahn, Torsten Schaub, and Gottfried Schenner. Interactive Configuration with ASP Multi-Shot Solving. In *ConfWS-23*, 2023.
- 13 Andrew Cook, editor. *European Air Traffic Management: Principles, Practice and Research*. Ashgate, 2007.
- 14 Veronica Dal Sasso, Franklin Djeumou Fomeni, Guglielmo Lulli, and Konstantinos G. Zografos. Planning efficient 4d trajectories in air traffic flow management. *EJOR*, 276(2):676–687, 2019. doi:10.1016/J.EJOR.2019.01.039.
- 15 Ramon Dalmau, Gilles Gawinowski, and Jacek Kopec. Multi-objective air traffic flow management through lexicographic optimisation. In *SID24*, 2024.
- 16 D. Delahaye, J.-M. Alliot, M. Schoenauer, and J.-L. Farges. Genetic algorithms for partitioning air space. In *CAIA-94*, pages 291–297, 1994.
- 17 Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer. In *Reasoning Web*, volume 5689, pages 40–110, 2009. doi:10.1007/978-3-642-03754-2_2.

- 18 Mohammed M. S. El-Kholany, Martin Gebser, and Konstantin Schekotihin. Problem Decomposition and Multi-shot ASP Solving for Job-shop Scheduling. *TPLP*, 22(4):623–639, 2022. doi:10.1017/S1471068422000217.
- 19 EUROCONTROL Experimental Centre. Innovative Slot Allocation: an Overview. EUROCONTROL, 2001.
- 20 Ke Feng, Dahai Liu, Yongxin Liu, Hong Liu, and Houbing Song. Graphdac: A graph-analytic approach to dynamic airspace configuration. In *IRI23*, pages 235–241, 2023. doi:10.1109/IRI58017.2023.00048.
- 21 David García-Heredia, Antonio Alonso-Ayuso, and Elisenda Molina. A Combinatorial model to optimize air traffic flow management problems. *Comput. Oper. Res.*, 112:104768, 2019. doi:10.1016/J.COR.2019.104768.
- 22 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, Philipp Wanko, and Max Ostrowski. Theory solving made easy with clingo 5 (extended version), 2016.
- 23 Martin Gebser, Philipp Wanko, Roland Kaminski, Benjamin Kaufmann, Marius Lindauer, Max Ostrowski, Javier Romero, Torsten Schaub, and Sven Thiele. *Clingo Guide*. Univ. of Potsdam, 2019.
- 24 Michael Gelfond and Nicola Leone. Logic programming and knowledge representation—The A-Prolog perspective. *Artif. Intell.*, 138(1-2):3–38, 2002. doi:10.1016/S0004-3702(02)00207-2.
- 25 Jeongseok Kim and Kangjin Kim. Enhancing Urban Air Mobility Scheduling Through Declarative Reasoning and Stakeholder Modeling. *Aerospace*, 12(7):605, 2025.
- 26 Go Nam Lui, Guglielmo Lulli, Luigi De Giovanni, Martina Galeazzo, Iciar Garcia-Ovies Carro, and Rebeca Llorente Martinez. A robust optimization approach for dynamic airspace configuration. In *ATRS25*, 2025.
- 27 Go Nam Lui, Guglielmo Lulli, M Florencia Lema-Esposto, and Rebeca Llorente Martinez. Airspace sector design: An optimization approach. In *SID24*, 2024.
- 28 Anh-Duong Nguyen, Loc Pham, Nathan Lindsay, Liang Sun, and Son C. Tran. Optimized eVTOL Aircraft Scheduling - An Answer Set Programming Based Approach. In *AIAA AVIATION/ASCEND 2024*, 2024.
- 29 Veronika Semmelrock and Gerhard Friedrich. Investigating the grounding bottleneck for a large-scale configuration problem: Existing tools and constraint-aware guessing. In Martin Gebser, Daniela Inclezan, Francesco Ricca, Manuel Carro, and Mirosław Trzuszczynski, editors, *Proceedings 41st International Conference on Logic Programming*, Rende, Italy, 12–19th September 2025, volume 439 of *Electronic Proceedings in Theoretical Computer Science*, pages 482–495. Open Publishing Association, 2026. doi:10.4204/EPTCS.439.33.
- 30 Marina Sergeeva, Daniel Delahaye, and Catherine Mancel. 3d airspace sector design by genetic algorithm. In *MT-ITS15*, pages 499–506, 2015. doi:10.1109/MTITS.2015.7223300.
- 31 Qihang Xu, Yutian Pang, and Yongming Liu. Dynamic airspace sectorization with machine learning enhanced workload prediction and clustering. *J. Air Transp. Manag.*, 121:102683, 2024.
- 32 Min Xue. Airspace sector redesign based on Voronoi diagrams. *J. Aerosp. Inf. Syst.*, 6(12):624–634, 2009. doi:10.2514/1.41159.
- 33 Cheryl Wong Sze Yin, T.K. Venugopalan, and S. Suresh. A multi-objective approach for 3D airspace sectorization: A study on Singapore regional airspace. In *IEEE SSCI 2016*, pages 1–8, 2016.

A Appendix

A.1 AI Usage Statement

To improve spelling and wording we used ChatGPT (various ChatGPT-5 versions) and Gemini 3.1 Pro. Further, we used the same AI tools as coding assistants, for example to query software libraries, during debugging, and for documentation, however, substantial

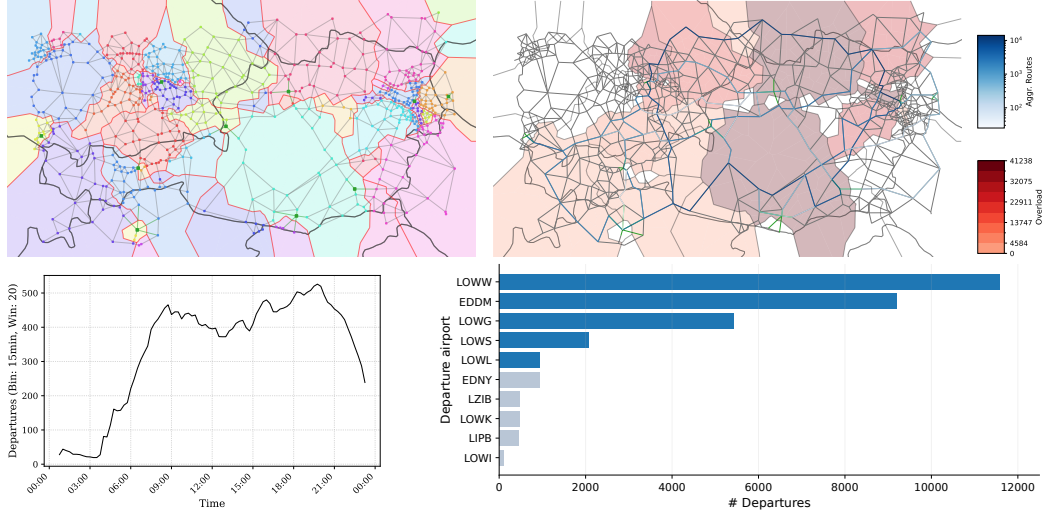


Figure 5 Figure displaying an instance of CE (Central Europe), from the data generator. Left top: Initial navpoint-sector figure. Right top: Initial total overload exceedances (K_o) and aggregated trajectory-counts figure. Left bottom: Filed flights departure times. Right bottom: Histogram of departure airports. Instance: 31622 flights, CE.

parts of the prototype were manually constructed (e.g., the entire ASP encoding). Lastly, we used AI tools to help in the data analysis part to generate Python scripts which are able to generate $\text{\LaTeX}$ tables and matplotlib figures.

A.2 Limitations

(1) *Heuristic nature.* ASPaeroFlow is a heuristic without global optimality guarantees; solution quality can depend on decomposition bounds and the overload resolution order. (2) *Operational fidelity.* Reroutes and sector partitions are generated on a navpoint graph and may violate unmodeled constraints; we also omit tactical interventions and weather uncertainty. Time is discretized for discrete optimization, which may induce errors. (3) *Capacity and occupancy abstraction.* Capacity assumes partitioning enables parallel staffing. (4) *Network scope and objectives.* We partially model airport/terminal constraints, so mitigating en-route overload may shift congestion elsewhere; efficiency is proxied mainly by arrival delay and stability terms, without explicit fuel/emissions, heterogeneous airline costs, or stakeholder fairness constraints.

A.3 Additional Experimental Results

In Table 6 we show the detailed results for the small and SOTA scenarios. Further, in Figure 5 we show an example of a generated instance produced by the data generator and Figure 6 shows the detailed significance results.

■ **Table 6** Small and SOTA scenarios (selected) results. We show overload (O) and arrival delay (D) of the considered variants.

Instance	EA-3x3 (Seed 150699)										CE-5x5 (Seed 150699)												
	Initial		ASPaeroFlow _{r,d_p}		CASA		MIP		ASP-P		Initial		ASPaeroFlow _{r,d_p}		CASA		MIP		ASP-P				
	O	D	O	D	O	D	O	D	O	D	O	D	O	D	O	D	O	D	O	D			
10	10	0	0		56	0	20	0	0	1	9	1	0	0		0	0	2	0	2	0	0	0
20	22	0	0		311	0	156	0	15	7	33	5	0	0		30	0	11	0	5	0	0	1
30	53	0	0		687	0	919	0	65	24	54	13	0	0		102	0	79	0	11	0	4	0
40	80	0	0		1370	0	1247	0	230	54	102	26	0	0		213	0	231	0	90	0	21	0
50	117	0	0		2115	0	2740	3	422	88	96	37	0	0		342	0	506	0	187	0	25	0
60	133	0	0		3285	0	3351	12	482	95	127	38	0	0		401	0	562	0	193	0	49	0
70	178	0	0		4504	0	5366	35	382	134	160	59	0	0		748	0	903	0	305	0	57	0
80	204	0	0		6106	0	6302	59	250	157	178	76	0	0		1005	0	1288	18	271	3	77	0
90	241	0	0		7651	0	9517	88	186	192	230	94	0	0		1356	0	2013	38	208	9	148	0
100	287	0	0		9520	0	11034	120	111	241	199	111	0	0		1578	0	2365	56	170	17	138	0

Instance	IND-4x10 (Seed 150699)										USA-7x7 (Seed 150699)												
	Initial		ASPaeroFlow _{r,d_p}		CASA		MIP		ASP-P		Initial		ASPaeroFlow _{r,d_p}		CASA		MIP		ASP-P				
	O	D	O	D	O	D	O	D	O	D	O	D	O	D	O	D	O	D	O	D			
10	15	0	0		69	0	36	0	7	0	3	1	0	0		10	0	1	0	1	0	1	0
20	21	0	0		192	0	152	0	34	0	5	12	0	0		66	0	27	0	10	0	3	0
30	45	0	0		577	0	503	0	166	0	28	26	0	0		312	0	251	0	63	0	7	0
40	70	0	0		904	0	1030	9	169	2	59	49	0	0		689	0	468	9	117	4	27	0
50	92	0	0		1395	0	1689	23	255	10	97	61	0	0		1031	0	968	15	146	5	43	0
60	136	0	0		2420	0	3115	77	133	38	125	80	0	0		1368	0	1513	26	129	10	50	0
70	135	0	0		3190	0	3241	72	177	35	147	89	0	0		1721	0	1791	30	168	9	81	0
80	186	0	0		4418	0	5481	131	85	54	177	117	0	0		2472	0	2871	47	161	14	104	0
90	193	0	0		5145	0	6016	138	111	77	186	140	0	0		2988	0	3168	65	175	18	139	0
100	233	0	0		6453	0	9523	181	87	100	197	180	0	0		4083	0	5584	94	186	38	173	0

Instance	EUR-10x10 (Seed 150699)										Instance	SOTA											
	Initial		ASPaeroFlow _{r,d_p}		CASA		MIP		ASP-P			Initial		ASPaeroFlow _{r,d}		ASPaeroFlow _{seq}		ATFM _{r,d}		ATFM _{r,d_p}			
	O	D	O	D	O	D	O	D	O	D		O	D	O	D	O	D	O	D	O	D		
10	8	0	0		2	0	15	0	4	0	0	BLO1	466	0	0	423	0	431	0	292	0	608	0
20	14	0	0		87	0	69	0	11	0	3	BLO2	294	0	0	1897	0	1843	0	1897	0	1921	0
30	25	0	0		93	0	128	0	33	0	1	BLO3	1884	0	0	27074	0	27074	0	27074	0	26845	0
40	49	0	0		199	0	503	0	158	0	5	IC	3368	0	0	85	0	222	0	26328	2744	6314	0
50	62	0	0		305	0	1045	7	172	0	13	c01	466	0	0	423	0	431	0	292	0	608	0
60	88	0	0		408	0	1621	23	208	0	41	c02	91	0	0	56	0	89	0	56	0	75	0
70	110	0	0		621	0	2370	42	194	0	68	c03	171	0	0	83	0	39	0	-86	0	-108	0
80	148	0	0		943	0	3637	62	210	0	117	mBLO1	14	0	0	-4	0	0	0	-18	0	-18	0
90	156	0	0		884	0	4009	69	194	0	99	mBLO3	1884	0	0	27074	0	27074	0	27074	84	26409	0
100	172	0	0		1181	0	5115	105	148	5	140												

A.4 ASP Encoding Details

Generated Facts: Flight Trajectories

$\text{trj}(f, p)$	Defines the available trajectories (route-delay options) p for flight f .
$\text{fpl}(f, n, t)$	Specifies the presence for a filed flight f at navpoint n at time t .
$\text{actd}(f, t, p)$	Indicates the actual departure time t for flight f using path p .
$\text{plana}(f, t)$	Defines the originally planned arrival time t for flight f .
$\text{acta}(f, t, p)$	Denotes the actual arrival time t for flight f given the chosen trj . p .
$\text{npx}(\dots)$	Encodes sequential trajectory segments: $(f, p, n_{i-1}, t_{i-1}, n_i, t_i)$.
$\text{chp}(\dots)$	Propagates trajectory selection: $(f_{\text{aff}}, p) \leftarrow (f, p)$.

Generated Facts: Sector Configurations

$\text{config}(c, k_{\text{conf}})$	Sector configuration c associated with k_{conf} underlying conflicts.
$\text{sec_cap_conf}(\dots)$	Sets the capacity k_{cap} of sector s at time t under configuration c – assuming all flights fixed except to-be-optimized flights.
$\text{nav_assign_conf}(\dots)$	Maps navpoint n to sector s at time t within configuration c .

The following part of the encoding shows how guesses are performed. Line (1) shows how guesses for trajectories, and Line (2) shows guesses for sector configurations.

```

1 if{chtrj(ID,P):trj(ID,P)}1 ← flightID(ID).
2 if{chc(CONFIG):config(CONFIG,_)}1.

```

Next, we map the chosen trajectory to the navpoint trajectory that is actually flown, which is needed to compute the updated converted instance $\mathbf{M}'$ in Line 5 in Algorithm 4.1. Normal flights are shown in Lines (1) and (2), which map for each flown edge (v_0, v_1) the navpoints to the trajectory. Line (3) is a special case for handling atomic flights.

```

1 nvpt_f(ID,NAV0,T) ← actd(ID,TSTART,P),chtrj(ID,P),npx(ID,P,NAV0,T0,_,_),T=T0+TSTART.
2 nvpt_f(ID,NAV1,T) ← actd(ID,TSTART,P),chtrj(ID,P),npx(ID,P,_,_,NAV1,T1),T=T1+TSTART.
3 nvpt_f(ID,NAV0,T) ← actd(ID,TSTART,P),chtrj(ID,P),single_pos(ID,P,NAV0,T0),T=T0+TSTART.

```

Line (1) maps a chosen sector configuration to the effects it has on the navpoint sector

10:20 Decomposition Heuristics for Joint Air Traffic Flow & Capacity Management

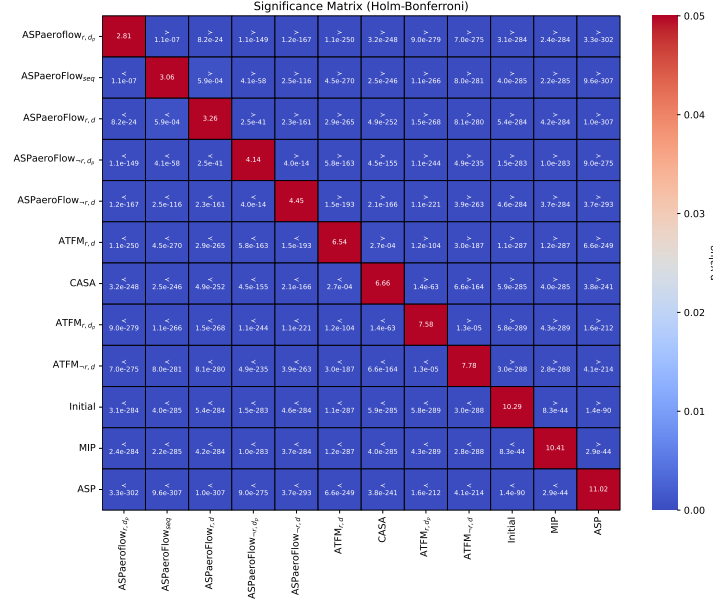


Figure 6 Matrix showing pairwise statistical significance results.

assignments and Line (2) maps the capacities of the chosen sector configuration to the *sec_cap* predicate.

```
1 nav_ass(NAV, SEC, TIME) ← chc(CONFIG), nav_assign_conf(NAV, SEC, TIME, CONFIG).
2 sec_cap(SEC, CAPACITY, TIME) ← chc(CONFIG), sec_cap_conf(SEC, CAPACITY, TIME, CONFIG).
```

Combining the chosen trajectory and the chosen config, we are left to map the flown navpoint trajectory to the actual effects on the sectors. Lines (1) and (2) handle the standard behavior for a trajectory; they encode the standard behavior for a flight $f \in \mathbf{F}$ reaching navpoint v_a at t_a , followed by arriving at v_b at t_b . Then f is in sector $sec^{-1}(v_a, t)$ in $t \in [t_a, t_a + \lfloor \frac{\Delta t}{2} \rfloor]$ and in sector $sec^{-1}(v_b, t')$ for $t' \in [t_a + \lfloor \frac{\Delta t}{2} \rfloor + 1, t_b]$. Additionally, we consider two special cases: first atomic flights (Line (3)) and then flights staying at a navpoint (Line (4)).

```
1 sec_f(ID, SECO, T) ← actd(ID, TSTART, P), chtrj(ID, P), time(T), nxp(ID, P, NAV0, T0, NAV1, T1), NAV0
  ≠ NAV1, T >= T0 + TSTART, T <= T1 + TSTART, DT = (T1 - T0) / 2, T <= DT + T0 + TSTART, nav_ass(NAV0, SECO, T).
2 sec_f(ID, SEC1, T) ← actd(ID, TSTART, P), chtrj(ID, P), time(T), nxp(ID, P, NAV0, T0, NAV1, T1), NAV0
  ≠ NAV1, T >= T0 + TSTART, T <= T1 + TSTART, DT = (T1 - T0) / 2, T > DT + T0 + TSTART, nav_ass(NAV1, SEC1, T).
3 sec_f(ID, SECO, T) ← actd(ID, TSTART, P), chtrj(ID, P), single_pos(ID, P, NAV0, T0), nav_ass(NAV0,
  SECO, T), T = T0 + TSTART.
4 sec_f(ID, SECO, T) ← actd(ID, TSTART, P), chtrj(ID, P), time(T), nxp(ID, P, NAV0, T0, NAV0, T1),
  T >= T0 + TSTART, T <= T1 + TSTART, nav_ass(NAV0, SECO, T).
```

If an overload occurs, we store this in the predicate *overload* (Line (1)), an arrival delay is stored in *arrival_delay* (Line (2)), and if a flight is rerouted, it is stored in *reroute* (Line (3)).

```
1 overload(SEC, T, LOAD - CAPACITY) ← sec_cap(SEC, CAPACITY, T), #count{ID: sec_f(ID, SEC, T)} =
  LOAD, LOAD > CAPACITY.
2 arrival_delay(ID, Y - T) ← chtrj(ID, P), plana(ID, T), acta(ID, Y, P).
3 reroute(ID) ← flightID(ID), not chtrj(ID, 0).
```

Integrated Line Planning and Strategic Ridepooling: Minimizing the Passengers' Travel Time

Lena Dittrich¹  

Department of Mathematics, RPTU University Kaiserslautern-Landau, Germany

Sven Jäger  

Institute of Mathematics, Technische Universität Berlin, Germany

Anita Schöbel  

Department of Mathematics, RPTU University Kaiserslautern-Landau, Germany

Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Abstract

Public transport systems increasingly combine fixed-line services with demand-responsive transportation such as ridepooling. In this paper, we present an integrated optimization model for line planning and strategic ridepooling that explicitly incorporates passenger routing into the planning process. Building on previous work on integrated line planning and strategic ridepooling as well as Change&Go networks, we develop a model for the integration of line planning and strategic ridepooling that minimizes the total travel time of all passengers while respecting a given budget. The resulting mixed-integer optimization model simultaneously determines line frequencies, ridepooling capacities, and passenger routes, capturing interactions between the two different transportation modes – one line-based mode and one mode without fixed lines and timetables – and still accounts for travel times, transfer times, and capacity decisions within a single optimization model. Focusing on an applicable model, we show how the data structure of an existing line planning model with integrated passenger routing in a Change&Go network can be improved, rendering the problem solvable for medium-sized instances. We consider the optimization model as a bi-criteria problem with the objective functions of travel time and operating cost, and compute pareto-optimal solutions with a budget constraint method. Our computational experiments on benchmark instances demonstrate the applicability of the model and show that integrating passenger routing into the strategic planning process yields multimodal transport systems that dominate solutions from approaches based on fixed routings.

2012 ACM Subject Classification Applied computing → Operations research; Applied computing → Transportation; Mathematics of computing → Mathematical optimization; Theory of computation → Network flows

Keywords and phrases Multi-modal planning, Line planning, Ridepooling, Integrated models

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.11

Funding *Lena Dittrich*: Funded by BMBF Project number 05M22UKB (SynphOnie).

1 Motivation

Line Planning is a well-known problem in public transport optimization, with many papers published on the topic, including the over 100 years old [12] and more recent surveys such as [14] or [15]. The task is to choose lines and their frequencies such that everybody can travel and a budget restricting the costs is respected. In this paper we aim to minimize the overall travel time of the passengers. One model which minimizes the passengers' travel times, including their transfer times between different lines, has been presented in [16].

¹ corresponding author



© Lena Dittrich, Sven Jäger, and Anita Schöbel;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 11; pp. 11:1–11:18



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In addition to line-based public transport systems, demand responsive services such as ridepooling are becoming more and more popular. Typically, a ridepooling service provider has a fleet of vehicles available to transport passengers, who can request to be taken from their origin to a destination. A passenger who requested a trip is then picked up by a vehicle, which drives them to their destination. The trips can be *pooled*, meaning that different trips with similar routes can be covered by the same vehicle, which is shared by the passengers. This pooling aspect leads to a reduction of the overall number of vehicles and vehicle driving time and distance, compared to a traditional taxi service or private transportation in individual cars. In [4], the role of ridepooling in multimodal public transport settings is discussed. Some example of these services include the company *MOIA* [11] which offers ridepooling in the city of Hamburg, Germany, with an all-electric fleet of vehicles, complementing the transportation infrastructure of line-based public transport and private vehicles. In the city of Birmingham, United Kingdom, *Max On-Demand* [10] offers on-demand transportation in different zones in and around the Birmingham city center. The transport association *Verkehrsbund Rhein-Neckar (VRN)* covers parts of the German federal states Baden-Württemberg, Rhineland-Palatinate and Hesse, including the entire Palatinate Forest and both cities of Kaiserslautern and Landau. As an addition to an extensive system of line-based public transportation, VRN also offers a demand-responsive transport option *VRNflexline* [20] in many different areas, especially in rural areas such as villages which are often not well-connected to the line systems.

Known as the *dial-a-ride-problem (DARP)*, the operative aspects of ridepooling services of assigning passenger requests to vehicles and scheduling their journeys have been studied extensively, both in online and in offline settings. The operative business is important, but not topic of our research in this paper. For a literature review we refer to [17, 19], see also [3] for some references.

In [3], *strategic* ridepooling with public transport line planning is integrated into one optimization model that jointly determines line plans and ridepooling services, thereby explicitly accounting for the interaction between fixed-line public transport and strategically planned ridepooling operations. However, in [3], the passengers' routes have been fixed in advance to receive traffic loads that then have to be covered either by a bus line or a ridepooling area. This is common in older models also for line planning, but is, in particular when planning multiple modes with different velocities, a simplification that may yield undesirable results. In this paper we hence extend [3] and include routing of passengers into the design of a multimodal public transport system, i.e., we integrate line-based transport with demand-responsive ridepooling services taking passengers' routes into account. Our contribution hence is two-fold:

- We present a new model for integrating strategic ridepooling with line planning *and* passengers' routing. We illustrate that integrating the routes of the passengers is crucial in particular when different modalities are planned. The numerical study shows the usefulness of the approach.
- We improve the underlying data structures for the Change&Go network making our approach applicable to medium-sized instances. Line planning with integrated passenger routing is a special case and hence benefits from this algorithmic improvement making the model presented in [16] eventually applicable for real-world instances.

The remainder of this paper is structured as follows: In Section 2 we give an introduction to strategic ridepooling and its integration with line planning, and highlight the benefits of including the routing of passengers into the model. In Section 3 we repeat the model for line planning from [16] which aims at minimizing the overall travel time of the passengers. We

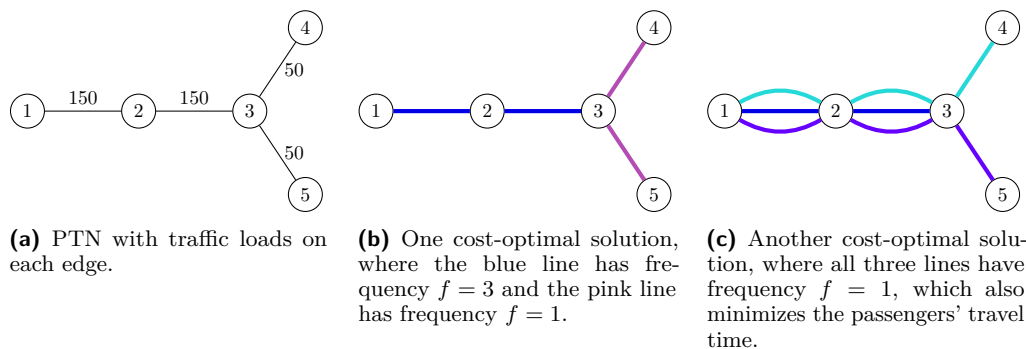
demonstrate that two model reduction techniques compared to the original model presented in [16] lead to significant improvements in performance. In Section 4 a new model for strategic ridepooling is presented, which has the objective of minimizing the passengers' travel time. Analogous to the lines and frequencies in line planning, this strategic ridepooling model chooses ridepooling areas and determines for each a number of vehicles to be available to the passengers. Both models are combined in Section 5 to form an integrated model for planning multiple modalities. Numerical experiments are presented in Section 6. We conclude in Section 7.

2 Importance of Passengers' Routes in Multimodal Settings

Ridepooling services can be a valuable addition to a line-based public transport system. Line-based systems are typically very good at transporting many passengers quickly and efficiently, but in situations with only a low passenger demand, the large vehicles used in these systems often contribute to a huge waste of resources, both economical and ecological. Contrarily, ridepooling services are more useful in lower demand settings, where ridepooling vehicles offer more flexibility since they can react spontaneously to passenger requests and hence avoid driving around unnecessarily without passengers. Hence, it is useful to have both modes of public transport as part of a public transport system, each deployed in areas where they are most beneficial to the overall system. Multi-modal transport is hence beneficial.

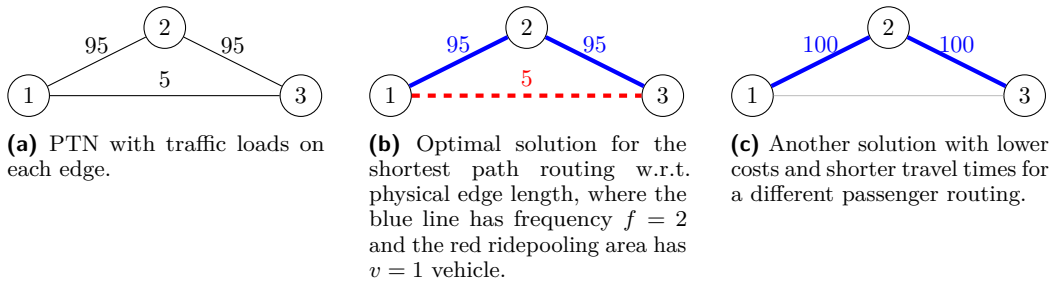
In order to model multi-modal transport, [3] combine line planning and strategic ridepooling in an integrated model minimizing the cost of the line concept and the ridepooling service. However, the model assumes that passengers are pre-routed such that only traffic loads are given. The number of transfers and the travel time of the passengers are hence not considered. We now improve this model by taking the passenger routing explicitly into account and instead of the cost we minimize their travel times.

Taking the routing into account is already relevant for only one mode of transport, as can be seen in the following example: Suppose we have a PTN as in Figure 1a, and a line modality with vehicle capacity 50. The traffic loads on the edges originate from three OD-pairs $(1, 3)$, $(1, 4)$ and $(1, 5)$, each with 50 passengers. Suppose the cost of operating a line with frequency f is f times the number of edges in the line. Then, from a cost perspective, both solutions in Figures 1b and 1c are optimal, but the latter would be preferable for the passengers since it does not require any transfers. In this example, the difference between the two solutions is only in the transfers.



■ **Figure 1** Example: Transfers should be considered in the optimization.

For multimodal planning, taking the routing into account is even more important, since the modes might travel at vastly different speeds, which has a big impact on the passengers' preferences and route choices. The following example shows this effect: Take the PTN as depicted in Figure 2a, with the OD-pairs $(1, 2)$, $(2, 3)$ with 95 passengers and $(1, 3)$ with 5 passengers. For the traffic loads in the PTN, the passengers have been routed along shortest paths according to the physical lengths of the edges. This leads to the solution in Figure 2b with one line and one ridepooling area. However, if, for instance, the line vehicles are high-speed trains which can travel fast and are not affected by traffic congestion, unlike a ridepooling vehicle which travels on roads, then, the passengers traveling between stations 1 and 3 would prefer the train over the ridepooling service, rendering the latter unnecessary. The new solution in Figure 1c is even cheaper than the solution from Figure 1b since it uses the same line with the same frequency but no ridepooling vehicles.



■ **Figure 2** Example: A pre-determined routing does not always lead to the best possible solution, even if passengers are routed on shortest paths.

Hence, in this paper, we propose a new model for strategic ridepooling which incorporates the passenger routing in the optimization step. The sum of all passengers' travel times can be minimized, while the overall cost is bounded by a budget constraint, or vice versa. We show that solutions of the integrated model from [3] yield feasible solutions for the new model, provided the budget constraint is chosen appropriately.

3 Line Planning with Minimal Travel Time

We briefly repeat the model for line planning from [16] and emphasize two techniques for reducing the model size compared to the original formulation, in order to be able to solve large instances without a specialized algorithm like column generation. We start with some notation also needed in Section 4.

Let $\text{PTN} = (V, E)$ be a public transport network represented as an undirected graph with a set of stops or stations V as its nodes and a set of direct connections E as edges. Let $\mathcal{OD} = (\text{OD}_{o,d})_{o,d \in V} \in \mathbb{N}^{V \times V}$ be the OD-matrix describing the expected passenger demand within a given planning period T . A line is a path in the PTN. Let $\mathcal{L}$ be a set of potential lines, the so-called *linepool*. We assume that the cost of a line depends linearly on its frequency $f_l \in \mathbb{N}$, i.e., the number of times it is operated within the given planning period T . For a line $l \in \mathcal{L}$ let $C_l \geq 0$ be the cost per frequency of this line. The task in the line planning problem is to assign a service frequency $f_l \in \mathbb{N}$ to each line $l \in \mathcal{L}$. Note that frequency $f_l = 0$ means that line l is not selected. A vector of frequencies $f \in \mathbb{N}^{\mathcal{L}}$ for a given linepool $\mathcal{L}$ is called a *line concept*.

The goal is to find a line concept that minimizes the sum of travel times. Since the exact travel times depend on the timetable, which is usually determined after the lines and their frequencies have been chosen, we assume constant speed driving on the lines and add a penalty $t^{\text{trans-}\mathcal{L}}$ for each transfer.

The passenger routing is often modeled using a Change&Go network, introduced by [16], which is a graph that has a node for each possible line at each served station and an additional node for each station. The arcs of the Change&Go network represent the driving along a line and the transfers between lines at a station. This is integrated into the line planning model, which then simultaneously determines the line frequencies and the passengers' routes as an (integral) multi-commodity flow.

The first reduction technique concerns the modelling of transfers in the Change&Go network. Instead of adding a transfer arc for each pair of lines at each station, transfers are modeled via a platform node with an arc from and to each line node at the station. This technique is described in [7, 5] and also used in the recent computational comparison of line planning models [18]. It is applicable when the transfer penalty is independent of the ordered pair of lines but depends only on making a transfer.

Formally, for a given PTN = (V, E) and line pool $\mathcal{L}$, the *Change&Go network* is the graph CGN = (V, E) defined as follows: The nodes are $\mathcal{V} = \mathcal{V}^p \cup \mathcal{V}^d$, where $\mathcal{V}^p = \{(s, 0, \mathcal{L}) : s \in V\}$ are the platform nodes and $\mathcal{V}^d = \{(s, l, \mathcal{L}) : l \in \mathcal{L}, s \in l\}$ are the driving nodes. The arcs are $\mathcal{E} = \mathcal{E}^p \cup \mathcal{E}^d$, where

$$\mathcal{E}^p = \{((s, 0, \mathcal{L}), (s, l, \mathcal{L})) \in \mathcal{V}^p \times \mathcal{V}^d, ((s, l, \mathcal{L}), (s, 0, \mathcal{L})) \in \mathcal{V}^d \times \mathcal{V}^p : s \in V, l \in \mathcal{L}, s \in l\}$$

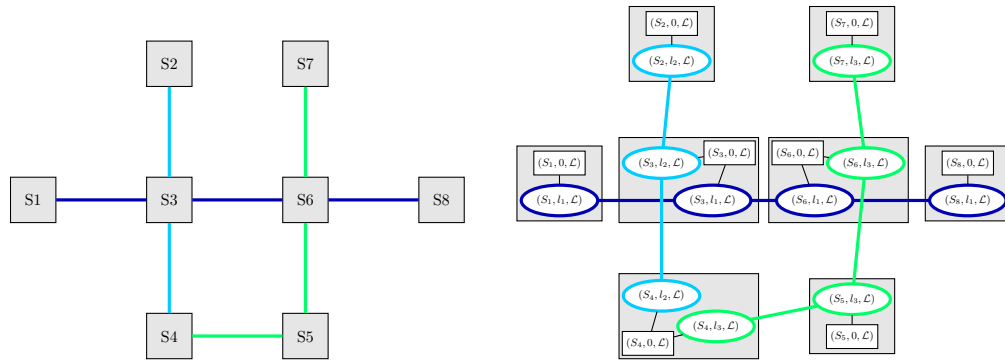
are the platform arcs and $\mathcal{E}^d = \bigcup_{l \in \mathcal{L}} \mathcal{E}^l$ with

$$\mathcal{E}^l = \{((s, l, \mathcal{L}), (t, l, \mathcal{L})), ((t, l, \mathcal{L}), (s, l, \mathcal{L})) \in \mathcal{V}^d \times \mathcal{V}^d : e = \{s, t\} \in l\}, \quad l \in \mathcal{L},$$

are the driving arcs. Note that the Change&Go network is a directed graph, where for each driving and platform arc, its backward arc also exists. Figure 3a shows a PTN with three lines, and Figure 3b shows the corresponding Change&Go network.

For a driving arc $e \in \mathcal{E}^d$ denote by $E(e)$ its underlying edge in the PTN. We define the arc weights to determine the length of a passenger's path:

$$t_e = \begin{cases} t_{E(e)}^{\text{drive-}\mathcal{L}} & \text{if } e \in \mathcal{E}^d, \\ \frac{1}{2} t_{E(e)}^{\text{trans-}\mathcal{L}} & \text{if } e \in \mathcal{E}^p. \end{cases}$$



(a) PTN with three lines.

(b) Change&Go network, note that each undirected arc in the picture represents a forward and a backward arc in the Change&Go network.

■ **Figure 3** Networks for Line Planning.

If a passenger wishes to travel from PTN-node $o \in V$ to $d \in V$, then in the CGN they are routed from platform node $(o, 0, \mathcal{L}) \in \mathcal{V}$ to platform node $(d, 0, \mathcal{L}) \in \mathcal{V}$. Their travel time can be computed using the knowledge of the travel times on the arcs along their path. Suppose

11:6 Line Planning and Strategic Ridepooling with Minimal Travel Time

p is a passenger's path from their origin to their destination in the CGN. Then the travel time on the path p is the sum of the travel times on the edges in the path:

$$t_p = \sum_{e \in p} t_e. \quad (1)$$

Note that each transfer arc in the CGN has travel time $\frac{1}{2}t^{\text{trans-}\mathcal{L}}$, but since each transfer consists of two transfer arcs (from driving node to platform node to another driving node), the time needed for each transfer is $t^{\text{trans-}\mathcal{L}}$. Furthermore, the initial entry onto and final exit from the line system each contribute $\frac{1}{2}t^{\text{trans-}\mathcal{L}}$ the passenger's travel time. Hence, in (1), it is assumed that the passengers also spend $t^{\text{trans-}\mathcal{L}}$ waiting to enter their first line vehicle.

For given line frequencies f any feasible routing of the passengers corresponds to a feasible (integral) multi-commodity flow in the Change&Go network, where each arc's capacity is determined by the line frequency. Most formulations model this either with path variables or with arc flow variables for every origin-destination pair. The second model reduction technique is to aggregate all passengers with the same origin, which reduces the number of variables and constraints. This is a standard technique for multi-commodity flow problems, see, e.g., [8], which was only recently applied to line planning with passenger routing [2, 1]. This formulation is integrated into the problem of choosing a frequency for each line from the linepool, resulting in an integer programming formulation with the constraints that all passengers must be transported and a given cost budget B is not exceeded and the objective of minimizing the overall sum of the passengers' travel times. To this end, let Θ be the node-arc-incidence-matrix of the CGN:

$$\Theta_{v,e} = \begin{cases} 1 & \text{if } e \text{ enters } v, \\ -1 & \text{if } e \text{ leaves } v, \\ 0 & \text{otherwise.} \end{cases}$$

For each origin $o \in V$ and CGN arc $e \in \mathcal{E}$, let $x_e^{(o)} \geq 0$ be the number of passengers with origin o who travel along arc e . For each origin $o \in V$ define $b^{(o)} = (b_v^{(o)})_{v \in V}$ with:

$$b_v^{(o)} = \begin{cases} -\sum_{d \in V} \text{OD}_{o,d} & \text{if } v = (o, 0, \mathcal{L}), \\ \text{OD}_{o,d} & \text{if } v = (d, 0, \mathcal{L}) \text{ for some } d \in V \setminus \{o\}, \\ 0 & \text{else.} \end{cases}$$

Let $\text{cap}^{(\mathcal{L})} \in \mathbb{N}$ be the capacity of a line vehicle. The frequencies $f_l \in \mathbb{N}$ of the lines $l \in \mathcal{L}$ need to be chosen large enough such that all passengers on their driving arcs can be transported, while overall, the cost of the line system is not above the given budget B .

This leads to the following integer linear program:

$$\begin{aligned} \min \quad & \sum_{o \in V} \sum_{e \in \mathcal{E}} t_e \cdot x_e^{(o)} & (\text{CG-L}) \\ \text{s.t.} \quad & \sum_{o \in V} x_e^{(o)} \leq \text{cap}^{(\mathcal{L})} \cdot f_l & \forall l \in \mathcal{L}, e \in \mathcal{E}_l \\ & \Theta x^{(o)} = b^{(o)} & \forall o \in V \\ & \sum_{l \in \mathcal{L}} C_l \cdot f_l \leq B \\ & f_l \in \mathbb{N} & \forall l \in \mathcal{L} \\ & x_e^{(o)} \in \mathbb{N} & \forall o \in V, e \in \mathcal{E} \end{aligned}$$

Any flow $x^{(o)} \in \mathbb{N}^{\mathcal{E}}$ of passengers starting at o that satisfies the flow conservation constraints $\Theta x^{(o)} = b^{(o)}$ can be decomposed into flows for all OD-pairs. After removing cycles, this can be decomposed into integral path flows, corresponding to paths used by passengers with origin o and destination d .

The two presented model reduction techniques have a significant effect on the performance on practical instances: To illustrate this, we applied Gurobi Version 13.0.2 [6] on an AMD Ryzen 7 5825U laptop with 16 GiB memory RAM to two instances from LINTIM [13]. For the Mandl instance [9], depicted in Figure 6 using the original formulation from [16] took ≈ 14 seconds. When replacing pairwise transfer arcs by platform-mediated transfers, the computation time was reduced to ≈ 10 seconds, with origin aggregation ≈ 2 seconds, and, when both techniques are combined, only ≈ 1 second.² For larger models such as the Athens metro instance, without origin aggregation, the solver exceeded the memory, whereas for aggregated flows, the optimum could be computed in ≈ 6 minutes, and with additional reduction of the transfer arcs, in ≈ 1 minute.

Line planning problems are known to be NP-complete, see e.g. [16].

4 Strategic Ridepooling with Minimal Travel Time

Ridepooling services are becoming more and more common. They typically function as follows: A provider has a fleet of vehicles which are available to the passengers, who request trips, i.e., to be taken by a ridepooling vehicle from their origin to their destination. The ridepooling service provider then assigns vehicles to service passenger requests, similar to a taxi service. But the passengers' trips can be pooled, meaning that it is allowed for the vehicles to be shared by multiple passengers, who might travel along small detours to allow for more efficient pooling of trips, such that the overall necessary number of vehicles and distance driven by vehicles can be reduced.

The operative aspects of ridepooling services have been very well researched. Contrarily, strategic ridepooling is a rather new problem in public transport optimization. Disregarding operational details such as specific vehicle routes, passenger requests and their assignments to available ridepooling vehicles, strategic ridepooling is the problem of deciding where a ridepooling service should be established and how much capacity (i.e. how many ridepooling vehicles) is needed to service the expected passenger demand. This is similar to the line planning problem, where the specific timetable is disregarded and only the layout and capacity of the line system is determined. Summarizing, strategic ridepooling considers ridepooling from a planning perspective rather than an operative one.

To model strategic ridepooling with minimal traveling time, take, again, a public transport network $\text{PTN} = (V, E)$, a planning period $T \in \mathbb{N}$ and a passenger demand OD-matrix $\text{OD} \in \mathbb{N}^{V \times V}$. Define a *ridepooling area* $r \subseteq E$ to be a connected set of edges in the PTN. Let $\mathcal{R}$ be a *ridepooling pool*, i.e., a set of ridepooling areas. Suppose that the cost of operating a ridepooling vehicle for the duration of the planning period T is $C_{\mathcal{R}} \geq 0$. The task in strategic ridepooling is to assign a number of vehicles $v_r \in \mathbb{N}$ to each ridepooling area $r \in \mathcal{R}$ to be available to the passengers in the planning period. Note that $v_r = 0$ is allowed, i.e., not every ridepooling area needs to be chosen. The number of vehicles for each area should be chosen such that all passengers can be transported and the expected overall travel time is minimized while a given cost budget B is not exceeded.

² These times include the whole execution of the program, including reading the input data and building the model.

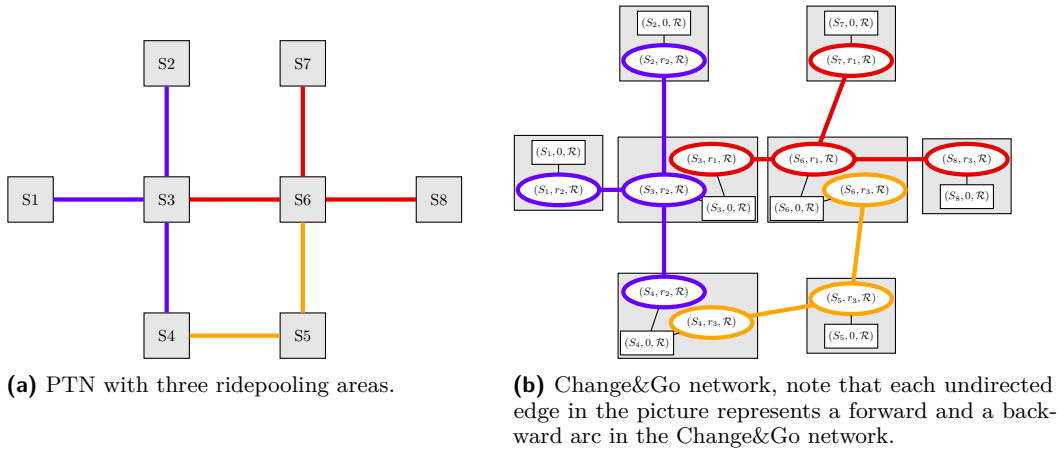
Similarly to the line planning case, define a Change&Go Network $\text{CGN} = (\mathcal{V}, \mathcal{E})$ for a given PTN (V, E) and ridepooling pool $\mathcal{R}$:

- $\mathcal{V} = \mathcal{V}^p \cup \mathcal{V}^d$ where:
 - Platform nodes: $\mathcal{V}^p = \{(s, 0, \mathcal{R}) : s \in V\}$
 - Driving nodes: $\mathcal{V}^d = \{(s, r, \mathcal{R}) \in V \times \mathcal{R} : s \in r\}$
- $\mathcal{E} = \mathcal{E}^p \cup \mathcal{E}^d$ where:
 - Platform arcs: $\mathcal{E}^p = \{((s, 0, \mathcal{R}), (s, r, \mathcal{R})) \in \mathcal{V}^p \times \mathcal{V}^d \text{ and } ((s, r, \mathcal{R}), (s, 0, \mathcal{R})) \in \mathcal{V}^d \times \mathcal{V}^p\}$
 - Driving arcs of ridepooling area $r \in \mathcal{R}$:
 $\mathcal{E}^r = \{((s, r, \mathcal{R}), (t, r, \mathcal{R})), ((t, r, \mathcal{R}), (s, r, \mathcal{R})) \in \mathcal{V}^d \times \mathcal{V}^d : \{s, t\} \in r\}$
 - Driving arcs: $\mathcal{E}^d = \bigcup_{r \in \mathcal{R}} \mathcal{E}^r$

Note the slight abuse of notation: A station $s \in V$ is said to be contained in a ridepooling area $r \in \mathcal{R}$ (i.e. $s \in r$), if there exists an edge $e = \{s, v\} \in E$ incident to s with $e \in r$.

The exact vehicle routes, and therefore also the exact travel times of the passengers, are not yet known, since the vehicles react spontaneously to the passenger requests that arise in practice. However, in strategic ridepooling, these operative details are disregarded. Instead, assume that for each edge $e \in E$ in the PTN, the time needed $t_e^{\text{drive-}\mathcal{R}} \geq 0$ for a ridepooling vehicle to traverse the edge is known. Furthermore, let $t^{\text{trans-}\mathcal{R}} \geq 0$ be an estimation for the time needed to transfer between two ridepooling vehicles from two different ridepooling areas. Then, the driving times of the arcs in the Change&Go network $\text{CGN} = (\mathcal{V}, \mathcal{E})$ with ridepooling are defined as follows:

$$t_e = \begin{cases} t_{E(e)}^{\text{drive-}\mathcal{R}} & \text{if } e \in \mathcal{E}^d, \\ \frac{1}{2}t^{\text{trans-}\mathcal{R}} & \text{if } e \in \mathcal{E}^p. \end{cases}$$



■ **Figure 4** Networks for Strategic Ridepooling.

Again, passengers' routes between PTN nodes $o \in V$ and $d \in V$ through the ridepooling public transport system correspond to routes in the CGN between platform nodes $(o, 0, \mathcal{R}) \in \mathcal{V}$ and $(d, 0, \mathcal{R}) \in \mathcal{V}$. The travel time along a path p is computed analogously:

$$\text{travel time} = \sum_{e \in p} t_e. \quad (2)$$

Each transfer contributes $t^{\text{trans-}\mathcal{R}}$ to the overall travel time along the path. The initial entry into the first ridepooling vehicle of the journey also comes with a waiting time $t^{\text{trans-}\mathcal{R}}$.

It should be noted that the actual routes of the vehicles are not known at this stage of the planning process. Some passengers might make small detours to allow the ridepooling vehicle to pick up other passengers whose journeys have a similar direction. This assignment of vehicles to passenger requests is part of the optimization of the operative aspects of the ridepooling service, which is not considered in strategic ridepooling. The travel times as computed in (2) are therefore only estimations, not unlike they are in the line planning case, where the timetable is not yet known.

Instead of finding a frequency for each line in a given linepool, in strategic ridepooling the task is to determine a number of ridepooling vehicles for each ridepooling area $r \in \mathcal{R}$ from a given ridepooling pool. The number of vehicles should be chosen such that it suffices to cover the expected passenger demand without leaving many vehicles unused, since the cost for each vehicle $C_{\mathcal{R}} \geq 0$ needs to be paid regardless of their utilization. Let $r \in \mathcal{R}$ be a ridepooling area and for each arc $e \in \mathcal{E}$ let $p_e \geq 0$ be the number of passengers who wish to travel along this arc. The overall required passenger travel time in the ridepooling area r can then be computed:

$$\text{travel time in area } r = \sum_{e \in \mathcal{E}^d: E(e) \in r} t_e \cdot p_e = \sum_{e \in \mathcal{E}_r \cap \mathcal{E}^d} t_{E(e)}^{\text{drive-}\mathcal{R}} \cdot p_e. \quad (3)$$

Let $\text{cap}^{(\mathcal{R})} \in \mathbb{N}$ be the capacity of a ridepooling vehicle. A vehicle assigned to a ridepooling area for the duration of the planning period T comes with T units of available driving time. Hence, for the number of vehicles v_r in the area $r \in \mathcal{R}$, it is required that:

$$\frac{1}{\text{cap}^{(\mathcal{R})}} \cdot \sum_{e \in \mathcal{E}^d: E(e) \in r} t_e \cdot p_e \leq T \cdot v_r. \quad (4)$$

With this observation, the number of vehicles can be described as linear constraints in a strategic ridepooling model analogous to the line planning model CG-L. It should be noted that the number of vehicles as given by 3 and 4 is only an estimation. In 3, the overall passenger time in the area $r \in \mathcal{R}$ consists only of the passengers' direct travel time. It is possible to include a parameter to incorporate expected detours or add time penalties for passenger pick-ups and drop-offs. Similarly, 4 assumes that vehicles can operate at full capacity at all times, although, in practice, this is rarely possible. Again, additional parameters could be introduced to incorporate this. For the remainder of this work, we use 3 and 4, but it is possible to include more parameters to take the aforementioned aspects into consideration.

Let Θ be the node-arc-incidence matrix of the Change&Go network for $\mathcal{R}$ and $b = (b^{(o)})_{o \in V}$ defined according to the OD-matrix as in CG-L. Again, the passengers are modeled as a multi-commodity flow aggregated by the origins. The travel time is minimized, while a given cost budget B must not be exceeded. The variables $\beta_{r,e} \geq 0$ for $r \in \mathcal{R}$ and $e \in \mathcal{E}_r$ represent the number of passengers who travel on this arc in the Change&Go network.

$$\begin{aligned}
\min \quad & \sum_{o \in V} \sum_{e \in \mathcal{E}} t_e \cdot x_e^{(o)} & (\text{CG-R}) \\
\text{s.t.} \quad & \sum_{o \in V} x_e^{(o)} \leq \beta_{r,e} & \forall r \in \mathcal{R}, e \in \mathcal{E}_r \cap \mathcal{E}^d \\
& \frac{1}{\text{cap}(\mathcal{R})} \sum_{e \in \mathcal{E}_r \cap \mathcal{E}^d} \beta_{r,e} \cdot t_e \leq T \cdot v_r & \forall r \in \mathcal{R} \\
& \Theta x^{(o)} = b^{(o)} & \forall o \in V \\
& \sum_{r \in \mathcal{R}} C_{\mathcal{R}} \cdot v_r \leq B \\
& x_e^{(o)}, v_r \in \mathbb{N} & \forall o \in V, e \in \mathcal{E}, r \in \mathcal{R} \\
& \beta_{r,e} \geq 0 & \forall r \in \mathcal{R}, e \in \mathcal{E}
\end{aligned}$$

The NP-completeness of CG-R can be derived from the known NP-completeness of CG-L.

5 Integrated Planning with Minimal Travel Time

For the integrated planning of lines and ridepooling with minimal travel time, we define a joint Change&Go network. To this end, let $\text{CGN}^{(\mathcal{L})} = (\mathcal{V}^{(\mathcal{L})}, \mathcal{E}^{(\mathcal{L})})$ be the Change&Go network for an instance of CG-L with underlying linepool $\mathcal{L}$ and $\text{PTN}^{(\mathcal{L})} = (V^{(\mathcal{L})}, E^{(\mathcal{L})})$. Analogously, let $\text{CGN}^{(\mathcal{R})} = (\mathcal{V}^{(\mathcal{R})}, \mathcal{E}^{(\mathcal{R})})$ be the Change&Go network for an instance of CG-R with underlying ridepooling pool $\mathcal{R}$ and $\text{PTN}^{(\mathcal{R})} = (V^{(\mathcal{R})}, E^{(\mathcal{R})})$. The public transport networks need not be disjoint, i.e., $V^{(\mathcal{L})} \cap V^{(\mathcal{R})}$ and $E^{(\mathcal{L})} \cap E^{(\mathcal{R})}$ need not be empty. Define a joint network $\text{PTN} = (V, E)$ with $V = V^{(\mathcal{L})} \cup V^{(\mathcal{R})}$ and $E = E^{(\mathcal{L})} \cup E^{(\mathcal{R})}$.

For multimodal planning problems, the joint Change&Go network should also capture the fact that intermodal transfers contribute differently to the travel time than transfers within one mode. In order to model this, define the joint Change&Go network as a two-stage network which has for every station a station node, a modality node for every modality at that station, and a line or ridepooling area node for every possible line or ridepooling area, respectively, at that station.

- $\mathcal{V} = \mathcal{V}^{(\mathcal{L})} \cup \mathcal{V}^{(\mathcal{R})} \cup \mathcal{V}^{(\mathcal{S})}$ with station nodes $\mathcal{V}^{(\mathcal{S})} = \{(s, 0, 0) : s \in V\}$,
- $\mathcal{E} = \mathcal{E}^{(\mathcal{L})} \cup \mathcal{E}^{(\mathcal{R})} \cup \mathcal{E}^{(\mathcal{S})}$ with station arcs

$$\begin{aligned}
\mathcal{E}^{(\mathcal{S})} = & \{((s, 0, 0), (s, 0, \mathcal{L})), ((s, 0, \mathcal{L}), (s, 0, 0)) : s \in V^{(\mathcal{L})}\} \\
& \{((s, 0, 0), (s, 0, \mathcal{R})), ((s, 0, \mathcal{R}), (s, 0, 0)) : s \in V^{(\mathcal{R})}\}.
\end{aligned}$$

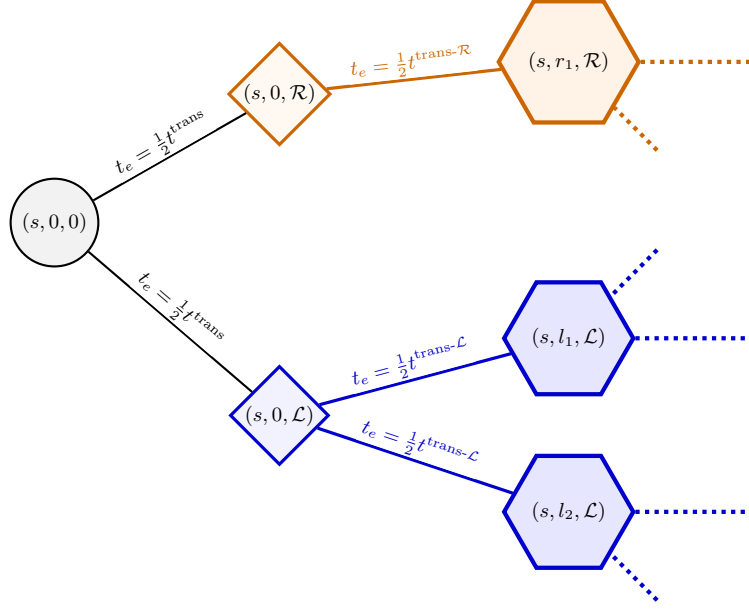
Figure 5 shows a section of joint Change&Go network with all nodes at one station $s \in V$. For the traveling times on the added station arcs, let $t^{\text{trans}} > 0$ be the penalty time for an intermodal transfer. Then, for $e \in \mathcal{E}^{(\mathcal{S})}$ define

$$t_e = \frac{1}{2} t^{\text{trans}}.$$

A passenger's route between two stations can then be represented by a path in the CGN between the two respective station nodes. A transfer between modes consists of one platform arcs from each mode and two station arcs, which contributes

$$\frac{1}{2} t^{\text{trans-}\mathcal{L}} + \frac{1}{2} t^{\text{trans-}\mathcal{R}} + 2 \cdot \frac{1}{2} t^{\text{trans}}$$

to the traveling time on the path.



■ **Figure 5** Multimodal Change&Go Network within one station $s \in V$ with one station node $(s, 0, 0)$, for the line modality a platform node $(s, 0, \mathcal{L})$ and two driving nodes $(s, l_1, \mathcal{L})$ and $(s, l_2, \mathcal{L})$, for the ridepooling modality a platform node $(s, 0, \mathcal{R})$ and a driving node $(s, r_1, \mathcal{R})$. The station node is connected to the platform nodes via station arcs, the platform nodes are connected to their respective driving nodes via platform arcs, the driving nodes are connected to driving nodes of the same line or ridepooling area at different stations. The station and platform arcs are labelled with their travel times. Note that each undirected arc in the picture represents two directed arcs, specifically a forward and a backward arc, in the CGN.

For the integrated planning of a line system and a ridepooling service in a joint Change&Go network, the two models CG-L and CG-R can be combined. The passengers form a multi-commodity flow between the station nodes are again aggregated by the origins. Line frequencies and ridepooling vehicles have to be assigned in a way that offers enough capacity on the driving arcs to accommodate the passenger flow. The overall travel time is minimized, while the cost cannot exceed a given budget B .

$$\begin{aligned}
 \min \quad & \sum_{o \in V} \sum_{e \in \mathcal{E}} t_e \cdot x_e^{(o)} & (\text{CG-L-R}) \\
 \text{s.t.} \quad & \sum_{o \in V} x_e^{(o)} \leq \text{cap}^{(\mathcal{L})} \cdot f_l & \forall l \in \mathcal{L}, e \in \mathcal{E}_l^{(\mathcal{L})} \\
 & \sum_{o \in V} x_e^{(o)} \leq \beta_{r,e} & \forall r \in \mathcal{R}, e \in \mathcal{E}_r^{(\mathcal{R})} \cap \mathcal{E}^{(\mathcal{R})d} \\
 & \frac{1}{\text{cap}^{(\mathcal{R})}} \sum_{e \in \mathcal{E}_r^{(\mathcal{R})} \cap \mathcal{E}^{(\mathcal{R})d}} \beta_{r,e} \cdot t_e \leq T \cdot v_r & \forall r \in \mathcal{R} \\
 & \Theta x^{(o)} = b^{(o)} & \forall o \in V \\
 & \sum_{l \in \mathcal{L}} C_l f_l + \sum_{r \in \mathcal{R}} C_r v_r \leq B \\
 & x_e^{(o)}, f_l, v_r \in \mathbb{N} & \forall o \in V, e \in \mathcal{E}, l \in \mathcal{L}, r \in \mathcal{R} \\
 & \beta_{r,e} \geq 0 & \forall r \in \mathcal{R}, e \in r
 \end{aligned}$$

The NP-completeness of CG-L-R follows from the NP-completeness of CG-L and CG-R.

11:12 Line Planning and Strategic Ridepooling with Minimal Travel Time

There are possible variations of this model CG-L-R: For instance, there could be separate budgets $B^{(\mathcal{L})}$ or $B^{(\mathcal{R})}$ for each of the modalities. The (intra- and intermodal) transfer times could vary between different stations, since often the size and layout of a station can impact the amount of time needed to transfer. Furthermore, while the basic model only has one line mode and one ridepooling mode, multiple modes of each kind are also possible. The joint Change&Go network can be extended to consist of not only two but more mode-specific sub-networks, each connected to the station nodes via station arcs at their platform nodes.

In [3], a model for strategic ridepooling and its integration with line planning was introduced. For a fixed passenger routing in the PTN (V, E) , given as edge loads $p_e \in \mathbb{N}$ for all $e \in E$. The so-called *vehicle frequencies* $\alpha_{r,e} \geq 0$ for $r \in \mathcal{R}$ and $e \in r$ are an estimation for the number of times a vehicle assigned to the ridepooling area r traverses the edge e within the planning period T . For more detailed explanation of their meaning and a procedure for determining these constants, we refer the reader to [3].

$$\begin{aligned}
 \min \quad & \sum_{l \in \mathcal{L}} C_l \cdot f_l + \sum_{r \in \mathcal{R}} C_{\mathcal{R}} \cdot v_r & (\text{Cost-L-R}) \\
 \text{s.t.} \quad & p_e \leq \sum_{\substack{l \in \mathcal{L}: \\ e \in l}} \text{cap}^{(\mathcal{L})} \cdot f_l + \sum_{\substack{r \in \mathcal{R}: \\ e \in r}} \text{cap}^{(\mathcal{R})} \cdot \alpha_{r,e} \cdot v_r & \forall e \in E \\
 & f_l, v_r \in \mathbb{N} & \forall l \in \mathcal{L}, r \in \mathcal{R}.
 \end{aligned}$$

We assume that the edge loads have been obtained from an OD-matrix $\mathcal{OD}$ by routing the passengers through the PTN (V, E) , and that the vehicle frequencies $\alpha_{r,e}$ for $r \in \mathcal{R}$ and $e \in E$ have been chosen as proposed in [3], i.e.,

$$\alpha_{r,e} = N_e \cdot \frac{T}{\sum_{e' \in E} N_{e'} t_{e'}}$$

where $L_e = \left\lceil \frac{p_e}{\text{cap}^{(\mathcal{R})}} \right\rceil$ and

$$N_e = \begin{cases} L_e & \text{if } L_e \text{ is even,} \\ L_e + 1 & \text{if } L_e \text{ is odd.} \end{cases}$$

We can show that, for suitable budget B , feasible solutions of Cost-L-R yield feasible solutions for CG-L-R.

► **Theorem 1** (Feasible solutions of Cost-L-R and CG-L-R). *Let (f^*, v^*) be a feasible solution of Cost-L-R and $B \geq \sum_{l \in \mathcal{L}} C_l f_l^* + \sum_{r \in \mathcal{R}} C_{\mathcal{R}} v_r^*$. Then there exists a feasible solution for CG-L-R with the same frequencies f^* and numbers of ridepooling vehicles v^* .*

For a formal proof, see Appendix A.

6 Numerical Experiments

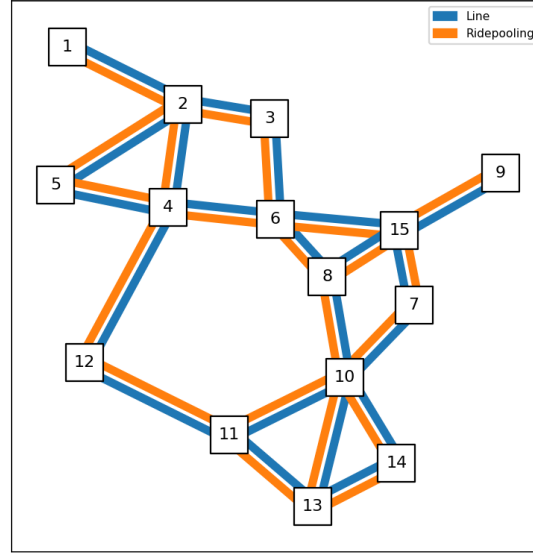
The model CG-L-R has been implemented in the open-source software toolbox LINTIM [13]. In the following, we present results from computational experiments. All IP formulations were solved in Gurobi 12.0.0 [6].

In this example, we use the small academic *mandl* graph [9], depicted in Figure 6, with a line modality and ridepooling allowed at all stations and on all edges. The linepool $\mathcal{L}$ is constructed as follows: First, for each pair of nodes a shortest path is computed. Then, all shortest paths which are contained in another shortest path are removed from the set

of paths. The remaining paths form the linepool. We assume that the cost of a line are computed as follows:

$$C_l = C_{\mathcal{L}}^{\text{fixed}} + \sum_{e \in l} C_{\mathcal{L}}^{\text{length}} \cdot t_e^{\text{drive-}\mathcal{L}}.$$

The ridepooling pool $\mathcal{R}$ contains one ridepooling area for each node, which consists of all edges incident to the node. Properties of the pools as well as cost parameters can be found in Table 1.



■ **Figure 6** PTN of the *mandl* graph.

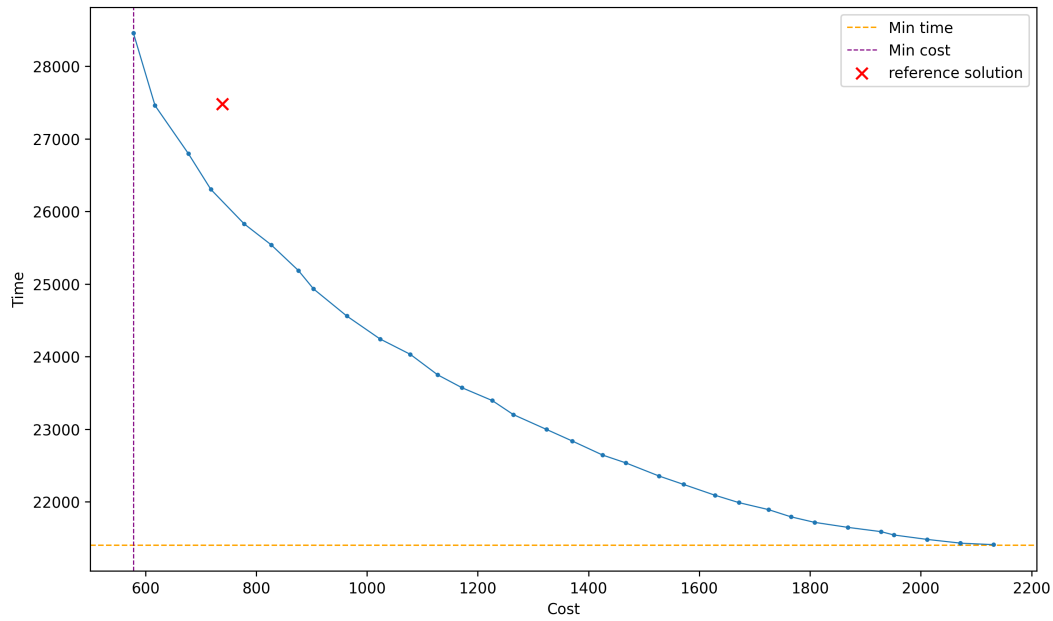
■ **Table 1** Input parameters and properties of the line- and ridepooling pools.

$C_{\mathcal{L}}^{\text{fixed}}$	$C_{\mathcal{L}}^{\text{length}}$	$C_{\mathcal{R}}$	$\text{cap}^{(\mathcal{L})}$	$\text{cap}^{(\mathcal{R})}$	$ \mathcal{L} $	$ \mathcal{R} $
8	4	60	50	6	22	15

■ **Table 2** Transfer times.

$t^{\text{trans-}\mathcal{L}}$	$t^{\text{trans-}\mathcal{R}}$	t^{trans}
8	4	2

For the passenger demand for each pair of nodes $v_1, v_2 \in V$ with $v_1 \neq v_2$ we have 5 passengers who wish to travel from v_1 to v_2 . The driving times on the edges differ between the two modalities and have been chosen such that the line modality takes about 1.2 times as long as the ridepooling modality. The platform and station times for each modality can be seen in Table 2. This means that while the line vehicles are cheaper and therefore preferable in terms of minimizing the operating cost, the ridepooling vehicles come with better traveling times for the passengers.

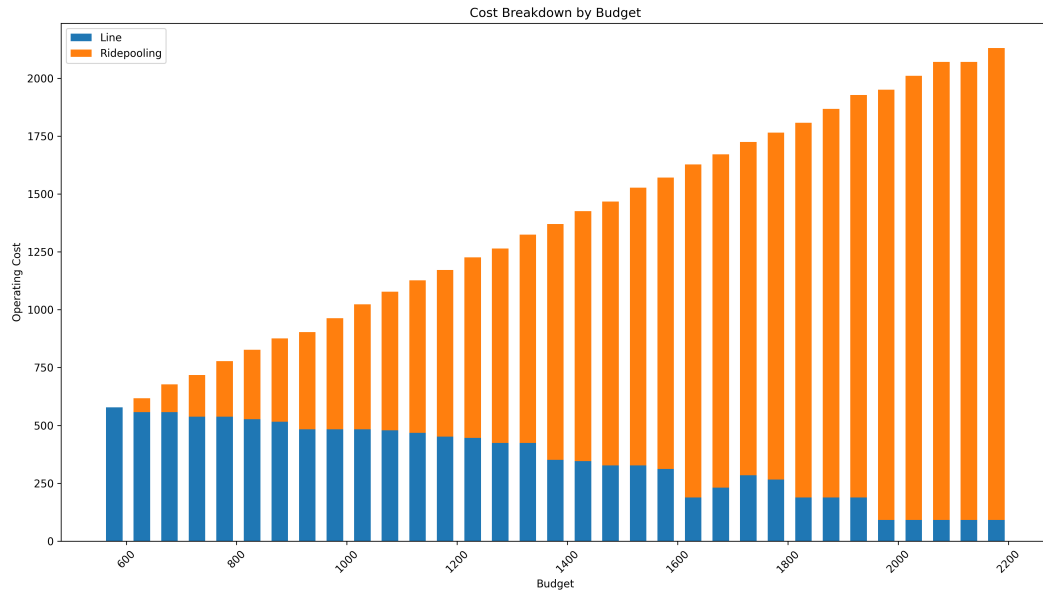


■ **Figure 7** Pareto front of CG-L-R with the optimal solution of Cost-L-R as the reference solution.

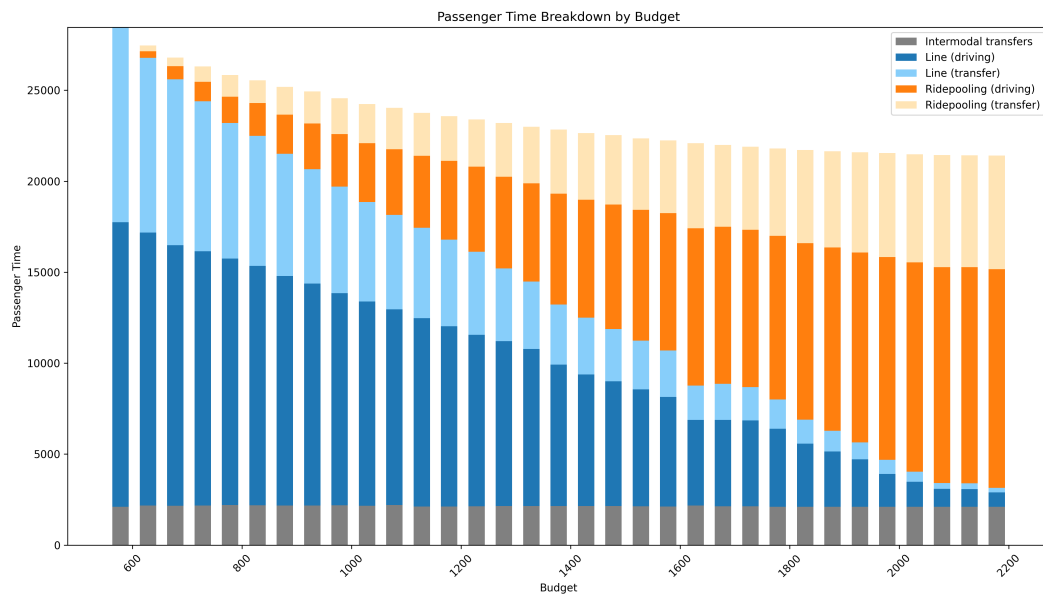
Both the cost and the travel time can be used as an objective function of CG-L-R. It can even be rewritten as a multi-objective optimization problem with the two objective functions of operating cost and travel time. By varying the budget in CG-L-R, we can compute a set of pareto-optimal solutions. All solutions have an optimality gap of at most 0.1%. The resulting pareto front is depicted in Figure 7. The reference solution is the optimal solution of Cost-L-R, where the passengers have been routed on shortest paths in the PTN, which yields a feasible solution for CG-L-R by Theorem 1, which is dominated by many of the optimal solution of Cost-L-R. The solution of Cost-L-R is even dominated in terms of cost. Since the routing is determined beforehand, without taking the cost into account, it is possible that for some passengers there exist alternative paths with equal travel time. Redistributing passengers between different paths of equal travel time can lead to a reduction in operating cost. Since ridepooling offers faster travel times for the passengers, but the line modality has lower costs, the low-budget solutions tend to have a higher portion of line-based transportation, while the size of the ridepooling portion of solutions increases for higher budgets, leading to lower travel times. Figure 8 shows how the two modalities contribute to the objective functions.

7 Conclusion and Further Research

In this paper, we presented a new optimization model for the integrated planning of line-based public transport and strategic ridepooling with the objective of minimizing passengers' travel times. Building on existing approaches to line planning and strategic ridepooling, we introduced new formulations for both planning problems that explicitly incorporate passenger routing through Change&Go networks. These formulations were then combined into a unified multimodal optimization model that simultaneously determines line frequencies, ridepooling capacities, and passenger routes while respecting a given operating budget. By integrating passenger routing directly into the planning process, the proposed model is able to account



(a) Cost of solutions for CG-L-R.



(b) Travel time of solutions for CG-L-R.

Figure 8 Objective function values of the pareto solutions of CG-L-R from Figure 7 split by modality with the corresponding budget at the x-axis.

for transfer times, different travel speeds of the transportation modes, and the interaction between line-based services and demand-responsive ridepooling. This leads to solutions that are better aligned with the passengers' perspective than approaches relying on predetermined routings or minimizing operating costs alone.

The proposed model provides a first step towards a comprehensive strategic planning model for multimodal public transportation systems that jointly optimizes fixed-line and demand-responsive services from a travel-time perspective. The computational results

demonstrate the applicability of the approach and illustrate how the integrated optimization can exploit the strengths of both transportation modes to improve service quality under limited resources.

More experiments on larger and even real-life instances are part of current ongoing research. On the one hand, more model reduction techniques could be investigated: for example, the flow of passengers starting at origin o is limited to the arcs in the Change&Go network that are reachable from o . Precomputing these subnetworks reduces the number of variables further. However, at some point, these reduction techniques alone will not suffice any more. For such instances, decomposition approaches often offer promising alternatives to common IP solvers. Heuristic approaches could also be developed, providing feasible, and ideally also good solutions which could also be used as warm-starts for other algorithms aiming to solve instances to optimality, which can often improve both runtime and solution quality.

Of course, modes of transportation such as walking or cycling, which do not require any planning or operation by a public transport provider, are also an important part of a public transportation system, since passengers can also use them for parts of their journeys. Hence, they should also be considered in the planning process. We are working on also incorporating such modes into our multimodal planning models. Furthermore, the planning process of a public transport system does not end after the line planning stage: a timetable must be determined in a next step. It is an ongoing research question how to include ridepooling from a strategic perspective into the timetabling stage of the planning process.

References

- 1 S. Bull, J. Larsen, R. M. Lusby, and N. J. Rezanova. Optimising the travel time of a line plan. *JOR*, 17(3):225–259, October 2018. doi:10.1007/s10288-018-0391-5.
- 2 S. H. Bull, R. M. Lusby, and J. Larsen. An optimization based method for line planning to minimize travel time. In *Proceedings of the 13th Conference on Advanced Systems in Public Transport (CASPT) 2015*. Erasmus University, 2015.
- 3 L. Dittrich, M. Rihlmann, S. Roth, and A. Schöbel. A model for strategic ridepooling and its integration with line planning. In Jonas Sauer and Marie Schmidt, editors, *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2025)*, volume 137 of *Open Access Series in Informatics (OASIs)*, pages 16:1–16:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIs.ATMOS.2025.16.
- 4 M. Friedrich, M. Hartl, and C. Mägg. A modeling approach for matching ridesharing trips within macroscopic travel demand models. *Transportation*, 45(6):1639–1653, 2018. doi:10.1007/s11116-018-9957-5.
- 5 M. Goerigk and M. Schmidt. Line planning with user-optimal route choice. *European Journal of Operational Research*, 259(2):424–436, June 2017. doi:10.1016/j.ejor.2016.10.034.
- 6 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL: <https://www.gurobi.com>.
- 7 M. Karbstein. *Line Planning and Connectivity*. PhD thesis, Technische Universität Berlin, 2013.
- 8 A. Kazemi, P. Le Bodic, A. T. Ernst, and M. Krishnamoorthy. New partial aggregations for multicommodity network flow problems: An application to the fixed-charge network design problem. *Computers & Operations Research*, 136:105505, December 2021. doi:10.1016/j.cor.2021.105505.
- 9 C. E. Mandl. *Applied Network Optimization*. Academic Press, London, UK, 1979.
- 10 Max On-Demand. On-demand rideshare in Birmingham and Homewood, AL. URL: <https://city.ridewithvia.com/birmingham> [cited 2026-07-01].

- 11 Moia. Ridepooling for Hamburg. URL: <https://www.moia.io/en/passengers> [cited 2026-07-01].
- 12 A. Patz. Die richtige Auswahl von Verkehrslinien bei großen Strassenbahnnetzen. *Verkehrstechnik*, 50/51, 1925. (in German).
- 13 P. Schiewe, A. Schöbel, O. Herrala, M. Rihlmann, S. Roth, S. Albert, C. Biedinger, T. Dahlheimer, L. Dittrich, K. Hoffmann, S. Jäger, P. Pattanaik, A. Schiewe, M. Stinzendorf, and R. Urban. Documentation for LinTim 2025.11, 2025. URL: <https://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-93118>.
- 14 M. Schmidt and A. Schöbel. Modeling and optimizing transit lines. In S. Parragh and T. V. Woensel, editors, *Research Handbook on Transport Modeling*, Research Handbooks in Transportation Studies, chapter 16. Edward Elgar Publishing, 2025.
- 15 M. Schmidt and A. Schöbel. Passenger-oriented and robust transit line planning. In S. Parragh and T. V. Woensel, editors, *Research Handbook on Transport Modeling*, Research Handbooks in Transportation Studies, chapter 17. Edward Elgar Publishing, 2025.
- 16 A. Schöbel and S. Scholl. Line planning with minimal travel time. In *ATMOS 2005*, number 06901 in Dagstuhl Seminar Proceedings, 2006. doi:10.4230/OASICS.ATMOS.2005.660.
- 17 M. M. Solomon and J. Desrosiers. Survey paper—time window constrained routing and scheduling problems. *Transportation Science*, 22(1):1–13, 1988. doi:10.1287/TRSC.22.1.1.
- 18 R. van Lieshout and B. van Rossum. Line planning at scale: Models, methods, and insights, 2026. doi:10.48550/arXiv.2606.24386.
- 19 P. Vansteenwegen, L. Melis, D. Aktaş, B. D. G. Montenegro, F. S. Vieira, and K. Sörensen. A survey on demand-responsive public bus systems. *Transportation Research Part C: Emerging Technologies*, 137:103573, 2022. doi:10.1016/j.trc.2022.103573.
- 20 Verkehrsverbund Rhein-Neckar. VRNflexline. URL: <https://www.vrn.de/mobilitaet/bedarfsverkehre/flexline/> [cited 2026-07-01].

A Bonus Material

Proof of Theorem 1. The budget constraint is satisfied by the assumptions about the budget.

Define for $r \in \mathcal{R}$ and $e \in \mathcal{E}_r^{(\mathcal{R})}$ define $\beta_{r,e} = \alpha_{r,E(e)} \cdot \text{cap}^{(\mathcal{R})} \cdot v_r$. Then:

$$\begin{aligned}
 \sum_{e \in \mathcal{E}_r^{(\mathcal{R})}} \beta_{r,e} t_e &= \sum_{e \in \mathcal{E}_r^{(\mathcal{R})}} \alpha_{r,E(e)} \text{cap}^{(\mathcal{R})} v_r t_{E(e)} \\
 &= \text{cap}^{(\mathcal{R})} v_r \sum_{e \in \mathcal{E}_r^{(\mathcal{R})}} N_{E(e)} \frac{T}{\sum_{e' \in \mathcal{E}_r^{(\mathcal{R})}} N_{E(e')} t_{E(e')}} t_{E(e)} \\
 &= \text{cap}^{(\mathcal{R})} v_r T \frac{\sum_{e \in \mathcal{E}_r^{(\mathcal{R})}} N_{E(e)} t_{E(e)}}{\sum_{e' \in \mathcal{E}_r^{(\mathcal{R})}} N_{E(e')} t_{E(e')}} \\
 &= \text{cap}^{(\mathcal{R})} v_r T
 \end{aligned}$$

This shows the ridepooling vehicle constraint.

It remains to show that there exists a feasible passenger flow $x = (x_e^{(o)}) \in \mathbb{N}^{\mathcal{E} \times V}$. For $\bar{e} \in E$ a PTN-edge, define $\mathcal{E}(\bar{e}) = \{e \in \mathcal{E} : E(e) = \bar{e}\}$ the set of corresponding driving arcs in the Change&Go network. Since the traffic loads have been obtained by routing the demand given by $\mathcal{OD}$ in the PTN, we know that there exists a passenger flow in the CGN with:

$$\forall o \in V : \Theta x^{(o)} = b^{(o)} \quad (\text{OD-constraint})$$

11:18 Line Planning and Strategic Ridepooling with Minimal Travel Time

and for all $\bar{e} \in E$

$$p_{\bar{e}} = \sum_{o \in V} \sum_{e \in \mathcal{E}(\bar{e})} x_e^{(o)} \leq \sum_{\substack{l \in \mathcal{L}: \\ \bar{e} \in l}} \text{cap}^{(\mathcal{L})} f_l + \sum_{\substack{r \in \mathcal{R}: \\ \bar{e} \in r}} \text{cap}^{(\mathcal{R})} \alpha_{r, \bar{e}} v_r \quad (\text{Cap})$$

Since all transfer arcs in the CGN have unlimited capacity, the flow can be redistributed between parallel driving arcs $\mathcal{E}(\bar{e})$ for the same PTN edge $\bar{e} \in E$ without violating the flow constraints OD-constraint. It follows that there exists a passenger flow $x \in \mathbb{N}^{\mathcal{E} \times V}$ which satisfies OD-constraint, Cap, and, by redistributing between parallel driving arcs, also the following additional capacity constraints:

$$\forall l \in \mathcal{L}, e \in \mathcal{E}_l^{(\mathcal{L})} : \sum_{o \in V} x_e^{(o)} \leq \text{cap}^{(\mathcal{L})} \cdot f_l \quad (\text{Cap-}\mathcal{L})$$

$$\forall r \in \mathcal{R}, e \in \mathcal{E}_r^{(\mathcal{R})} : \sum_{o \in V} x_e^{(o)} \leq \alpha_{r, E(e)} \text{cap}^{(\mathcal{R})} v_r = \beta_{r, e} \quad (\text{Cap-}\mathcal{R})$$

This implies the claim. ◀

A Multimodal Framework for Transport Planning

Lena Dittrich  

Department of Mathematics, RPTU University Kaiserslautern-Landau, Germany

Michael Rihlmann¹  

Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Philine Schiewe  

Department of Mathematics and Systems Analysis, Aalto University, Finland

Anita Schöbel  

Department of Mathematics, RPTU University Kaiserslautern-Landau, Germany

Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Abstract

Planning high-quality public transport systems requires taking multiple modalities of transportation into account simultaneously. This includes public transport such as trains or buses, but also ridepooling, walking or biking. While planning of public transport lines and timetables has been studied extensively for unimodal systems, research on settings which include more than one modality is just at its beginning. In this paper, we categorize the modalities as line-based, ridepooling, and nonscheduled for which we develop a framework for multimodal planning. This can be applied for different line-based modalities, e.g., for planning bus and tram lines simultaneously, but also for modalities from different categories, e.g., for planning ridepooling together with bus lines while taking footpaths into account. Technically, we extend classic graph structures from unimodal public transport optimization to the multimodal case and show how established solution methods for the unimodal case carry over to the multimodal setting. In particular, we formulate multimodal line planning and periodic timetabling problems. We demonstrate the practical applicability of the framework through a case study on real-world data from Helsinki, Finland, considering five line-based modalities and walking. Our experiments show that incorporating multimodal aspects into the planning process consistently reduces average passenger travel time.

2012 ACM Subject Classification Applied computing → Operations research; Applied computing → Transportation; Mathematics of computing → Mathematical optimization; Software and its engineering → Software libraries and repositories

Keywords and phrases multimodal planning, line planning, timetabling

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.12

Funding *Lena Dittrich*: Funded by BMBF Project number 05M22UKB (SynphOnie).

Philine Schiewe: This work was supported by the Research Council of Finland (Flagship of Advanced Mathematics for Sensing Imaging and Modelling 359181).

Acknowledgements We want to thank Olli Herrala for implementing multimodal timetabling methods and generating the Helsinki dataset. We also want to thank Prof. Dr.-Ing. Markus Friedrich and Julian Zimmer from Institut für Straßen- und Verkehrswesen at University of Stuttgart for providing us with estimations for realistic input parameters of our models.

1 Motivation

High-quality public transport systems are of great significance in a world that is striving towards carbon neutrality and improved access within and across regions. To enable passengers to replace journeys usually undertaken with individual transport in private vehicles,

¹ corresponding author



© Lena Dittrich, Michael Rihlmann, Philine Schiewe, and Anita Schöbel;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 12; pp. 12:1–12:20

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

convenient and practical alternatives are necessary. One single modality, e.g., regional trains, is not enough to motivate passengers to use public transport since the first and last mile is not covered. In real-world scenarios, this is taken into account: public transport typically does not consist of only one modality. Besides line-based modalities, e.g., buses and trams or underground trains within a city, other means of transport such as walking, cycling or taxi rides are common. A ridepooling service can also be part of a passenger's journey.

However, so far, the planning process in public transport has mostly been focused on one single modality, typically, on buses or on trains. It is usually decomposed into its stages that are followed sequentially: For a given demand for this modality, first the stops are located, then the lines and their frequencies are determined, followed by timetabling, vehicle and crew scheduling, see, e.g., [5] for an early definition of the stages and [22, 24] for the sequential process.

Multimodal planning is hard since the different modalities are linked by two major aspects: First, passengers are not restricted when choosing which mode of transportation to use, i.e., a single passenger's journey can consist of different legs, each with a different modality. The resulting modal split thus depends on the attractiveness of each modality and how well they are coordinated. Second, different modalities may partially share the same infrastructure like streets or tracks, imposing further capacity restrictions on their combined traffic volume. Detecting shared infrastructure capacity is a subtle issue: two modalities may appear independent, while in fact competing for the same underlying infrastructure. We refer to Figure 1, discussed in more detail later, for an illustration.

While literature on public transport planning is at least 50 years old [4], multimodal planning has just recently become of interest. In [3], frequency setting in the context of multimodal line planning has been considered. Intermodal timetabling including flight services has been studied in [13]. Models for integrating line planning and strategic ridepooling have been provided in [8]. A multi-objective analysis for different transport modalities on one single edge is given in [27].

In freight transport, multimodal planning is more common, see the survey [9]. Note that for *evaluating* the passengers' travel times in an existing multimodal transport supply, *multimodal routing* is needed. This has been extensively researched. We refer to [1, 2, 6, 10] and references therein. Multimodal equilibria for public transport including ridepooling have been tackled in [16]. The authors of [21] consider a fixed multimodal system and plan the operation of an on-demand service optimizing different justice metrics.

In this paper, we propose a general framework to extend the traditional unimodal sequential approach to multimodal planning, where we can decide for each planning step if it is performed for each modality in a unimodal way, or if it is performed for multiple modalities at the same time. The data structures are developed so that known algorithms for unimodal planning can be applied to the multimodal case, in particular for line planning and timetabling. Our contribution is the following:

- We propose data structures and a framework for multimodal planning following the sequential approach.
- We show that the framework covers many problems with practical relevance as special cases.
- In our case study we show that multimodal planning improves public transport supplies.

The remainder of this paper is structured as follows. First, we introduce different categories of transport modalities in Section 2, grouping modalities with similar properties together, and differentiate the categories by outlining the key aspects that set them apart.

In Section 3 we introduce some known graph-structures commonly used in unimodal public transport optimization and extend them to the multimodal case. Section 4 gives some examples of possible applications for the multimodal models proposed in this work. We demonstrate in Section 4.1 by a case study using data from the city of Helsinki, Finland, that multimodal planning methods significantly enhance the quality of the public transport system. Finally, we conclude in Section 5.

2 Categories of Transport Modalities

Clearly, not all modes of transportation function in the same way: Bus or train systems have fixed timetables, while walking, cycling or ridepooling are not scheduled beforehand. Walking or cycling do not require action of a public transport provider, while line-based or ridepooling systems need to be planned, set in place and operated. To model different types of transport modalities, we propose three different *modality categories*, grouping modalities with similar properties together.

- **Line-based:** In a line-based transport system such as buses, trains, trams, or metros, vehicles travel along fixed lines. The departure and arrival times are determined by a timetable, which is planned in advance. A line-based system is operated according to the timetable, regardless of the actual demand.
- **Ridepooling:** Ridepooling works as follows: A provider has a fleet of vehicles. The passengers request trips, i.e., to be taken from an origin to a destination at a specified time (window). If the request is accepted, the ridepooling service provider sends a vehicle to pick up the passenger and drop them off at their destination. Passengers' trips can be pooled, meaning that different trips with a similar direction can be combined and served by the same vehicle. This can lead to small detours in the passengers' journeys, but also reductions in the overall necessary number of vehicles and driving distance. The vehicle routes depend on the passenger demand that takes place at a specific day. The planning decisions for ridepooling services are the areas where it is offered and the number of vehicles operating in each area. It is on the same level as the line planning step for line-based modalities, as it determines the rough layout of the system. This is known as *Strategic Ridepooling* and was introduced in [8]. The operative aspects, such as the routes of vehicles and their arrival and departure times, depend on the actual demand.
- **Nonscheduled:** Means of transport such as walking, cycling or the use of private cars are not scheduled and do not require any action by a provider during the operational phase. Passengers can travel using a nonscheduled modality, wherever the underlying road system allows it. The planning problem determines where footpaths or bike lanes should be built, see, e.g., [12] for bicycle network design. The availability of such modalities is independent of passenger demand or utilization and there are no capacity constraints.

The networks to be described in the next section can cover all three categories of modalities. There are two ways these modalities can be included in multimodal planning:

- (a) The modality can be assumed to be given and fixed, and hence (only) used for passenger routing.
- (b) The modality can be designed/planned within the optimization.

Planning a modality has mostly been researched for line-based modalities (e.g., bus line planning, train timetabling), and is at its start for ridepooling. It is more common to assume different types of modalities as given and fixed and include them when routing the passengers. In the case study in Section 4.1 we plan five different line-based modalities and use one fixed nonscheduled modality (walking) for determining optimal passenger routes.

3 Networks and Planning Problems

In this section, we develop the networks needed for the different planning stages of a multimodal public transport system. While theoretically all planning stages can be included in our framework, we concentrate on line planning and timetabling here. We start by introducing the *infrastructure network*, which contains information on the underlying infrastructure, its capacity, and the modalities that can use it. Additionally, we assume that the *public transport networks* for each of the single modalities are given. They are the basis for unimodal public transport planning in the literature. To detect and account for (partially) shared infrastructure between different modalities or between edges of the same modality, a relation between the infrastructure network and the public transport networks must be established. We also define one common *multimodal public transport network*, allowing passengers to transfer from one modality to another at common stops.

We proceed with the *change&go network* for line planning (see [25]) and the *event-activity-network* for timetabling (see [26]). Both are defined in their unimodal version per modality and in their multimodal version for joint planning of all modalities. As research on strategic ridepooling planning is still in its early stages, these two graph structures are presented for only line-based and nonscheduled modalities in this work, though a general extension of the change&go network to ridepooling modalities is possible and demonstrated in [7].

3.1 Infrastructure Network

To capture shared infrastructure capacity between different modalities, we first define the *infrastructure network* as the most basic underlying graph structure. It contains information on the capacity of the infrastructure segments which all modalities must respect jointly.

► **Definition 3.1** (Infrastructure Network). Let $\text{ISN} = (N, \Lambda)$ be an undirected (multi-)graph representing the infrastructure. The vertices of the graph $\nu \in N$ are called *nodes*. They can represent crossings and stops or stations. The edges $\lambda \in \Lambda$ are called *links* and are given with

- $\text{length}(\lambda)$: the physical length of the link,
- $\text{capacity}(\lambda)$: the maximal capacity of the link, i.e., the maximal amount of traffic that can be sent over it during one period.

Let $\mathcal{M}$ be a set of (line-based, ridepooling, and nonscheduled) modalities operating in a given infrastructure network $\text{ISN} = (N, \Lambda)$. We denote by $\mathcal{M}(\nu)$ the set of modalities, for which the node ν is a *stop*, i.e., an entry point for passengers to this modality. For a line-based or ridepooling modality, these are the stops or stations where the vehicles halt and passengers can board and alight, and for a nonscheduled modality, every infrastructure node reachable with this modality is also called a stop. Similarly, $\mathcal{M}(\lambda)$ denotes the modalities that can use the infrastructure link λ .

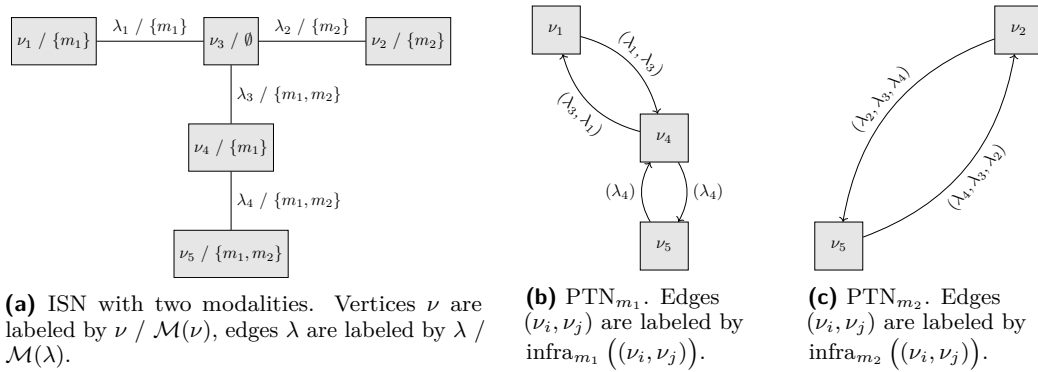
Suppose that we want to plan a public transport system in ISN with modalities $\mathcal{M}$ for a time period $T > 0$, e.g., T could represent one hour. The passenger demand in the network is represented by an *OD matrix* $\mathcal{OD} \in \mathbb{N}^{|N| \times |N|}$ where for a pair of nodes $\nu_i, \nu_j \in N$ the entry $\mathcal{OD}_{\nu_i, \nu_j}$ represents the number of passengers who wish to travel from ν_i to ν_j with any of the modalities from $\mathcal{M}$ in the planning period T . For every OD pair (ν_i, ν_j) with $\mathcal{OD}_{\nu_i, \nu_j} > 0$ we assume that $\mathcal{M}(\nu_i) \neq \emptyset$ and $\mathcal{M}(\nu_j) \neq \emptyset$, otherwise the passengers have no possibility to start or end their journey with any modality at the desired start or end node.

3.2 Public Transport Network

For planning a single modality, not all information from the infrastructure network is relevant: All parts of the underlying infrastructure network that are inaccessible to this modality need not be considered, e.g., bike lanes cannot be used by a bus. For nonscheduled modalities, all parts of the infrastructure network that cannot be reached using this modality can be omitted. For instance, if some infrastructure links represent a track system, then they are not relevant for passengers traveling on foot. To obtain a more suitable abstraction, one commonly relies on the public transport network, as established in the literature.

► **Definition 3.2** (Public Transport Network). For a modality $m \in \mathcal{M}$ the *public transport network* (PTN) of modality m is the directed simple graph $\text{PTN}_m = (V_m, E_m)$, where V_m contains all stops of modality m and E_m are all direct connections between them.

To connect the information from the infrastructure network to the public transport network, we assume that $V_m = \{\nu \in N : m \in \mathcal{M}(\nu)\} \subseteq N$. Furthermore, an edge $e = (\nu_1, \nu_2) \in E_m$ in the public transport network represents, that there is a direct connection from ν_1 to ν_2 in the infrastructure network without passing another stop of this modality. Therefore, we assume that together with the $\text{PTN}_m = (V_m, E_m)$, we are given a map $\text{infra}_m : E_m \rightarrow \Pi_{\text{ISN}}$, mapping each PTN edge $e = (\nu_1, \nu_2)$ to a path P in the infrastructure network starting at ν_1 and ending at ν_2 while $m \in \mathcal{M}(\lambda)$ holds for all $\lambda \in P$.



■ **Figure 1** An infrastructure network and corresponding public transport networks.

While the infrastructure network provides much more detailed information about the specific layout of the underlying road or track network, the public transport network of a modality omits irrelevant details, capturing only the information that is relevant to this modality. However, these details are important to correctly model infrastructure with limited capacity shared between different modalities. As an example, consider the ISN shown in Figure 1a with the two modalities m_1, m_2 . Figures 1b and 1c show the PTNs of the two modalities together with their underlying infrastructure paths. When focusing on the PTN graphs, one could overlook that m_1 and m_2 both use the same infrastructure link λ_4 between ν_4 and ν_5 . The maximal amount of traffic along λ_4 during one period is not allowed to exceed $\text{capacity}(\lambda_4)$ which can be expressed as

$$w_{m_1} \cdot Z_{m_1} + w_{m_2} \cdot Z_{m_2} \leq \text{capacity}(\lambda_4), \quad (1)$$

where Z_m denotes the number of units (e.g. vehicles, trains, bikes or pedestrians) of modality m during one period and w_m is the *infrastructure capacity weight* of modality m , expressing

how much traffic one unit of this modality produces. For nonscheduled modalities like walking or biking, the infrastructure capacity weight can be set to zero, but for line-based modalities like trains, where a certain safety distance between consecutive vehicles is required, it is important to respect the shared capacity.

For each edge $e \in E_m$ in the PTN_m , the amount of time necessary to traverse the edge using modality $m \in \mathcal{M}$ can be determined using the edge length as given by the lengths of the ISN links of the underlying infrastructure path $\text{infra}_m(e)$ and the speed of the modality.

► **Definition 3.3 (PTN Travel Times).** For each modality $m \in \mathcal{M}$ let $\text{speed}(m)$ be the speed at which passengers travel with this modality. Then, for $e \in E_m$ define its travel time by

$$t_e := \frac{1}{\text{speed}(m)} \sum_{\lambda \in \text{infra}_m(e)} \text{length}(\lambda).$$

If m is a ridepooling modality, the speed should be adjusted by the average detour factor, reflecting that vehicles rarely travel the shortest path when multiple trips are pooled. Note that one could also use edge-specific speed values or even travel times derived from empirical data rather than physical length.

For the multimodal case on the PTN-level, we define the multimodal public transport network by stacking the unimodal public transport networks.

► **Definition 3.4 (Multimodal Public Transport Network).** For an infrastructure network $\text{ISN} = (N, \Lambda)$ with modalities $\mathcal{M}$ define the *multimodal public transport network* as a directed simple graph $\text{MPTN} = (V, E)$ with

- $V := \{(\nu, 0) : \nu \in N\} \cup \left(\bigcup_{m \in \mathcal{M}} \{(\nu, m) : \nu \in V_m\}\right)$, and
- $E := E^{\text{trans}} \cup \left(\bigcup_{m \in \mathcal{M}} \{((\nu_1, m), (\nu_2, m)) : (\nu_1, \nu_2) \in E_m\}\right)$,
where $E^{\text{trans}} := \{((\nu, 0), (\nu, m)), ((\nu, m), (\nu, 0)) : \nu \in V_m\}$.

For each modality there are the nodes from its public transport network, and, additionally, there are transfer nodes at each infrastructure network node. Analogously, all modality-specific public transport edges are also in the MPTN, and all public transport network nodes are connected to the transfer node corresponding to the same infrastructure network nodes via transfer edges in both forward and backward directions.

The multimodal public transport network represents each modality through its public transport network, and, additionally, all possible transfer points between modalities. The travel times on the edges from the modality-specific public transport networks PTN_m are the same. For the transfer edges let t^{trans} be a time penalty for each transfer between different modalities and for $e \in E^{\text{trans}}$ define

$$t_e := \frac{1}{2} t^{\text{trans}},$$

since a passenger has to use two transfer edges to transfer between two different modalities. A path between two stations $\nu_1, \nu_2 \in N$ can be represented as a path between the two corresponding transfer nodes $(\nu_1, 0)$ and $(\nu_2, 0)$. The travel time of a passenger path can be computed as the sum of travel times of its edges. Note that the time of such a path then includes the transfer penalty t^{trans} one more time than there are transfers in the path, since the path begins and ends with a transfer edge. However, this is a constant which can be subtracted to get the actual passenger travel time.

3.3 Change&Go Network

For passengers traveling in a line-based transport system, not only transfers between different modalities are relevant, but also transfers within the same modality, since they are time consuming and often seen as inconvenient. In the following, denote by $\mathcal{M}^{\mathcal{L}} \subseteq \mathcal{M}$ the set of line-based modalities. A line l of a line-based modality $m \in \mathcal{M}^{\mathcal{L}}$ is a path in the modality's public transport network PTN_m . For each line-based modality $m \in \mathcal{M}^{\mathcal{L}}$ let $\mathcal{L}_m$ be a line pool, i.e., a set of possible lines of this modality where some of them will be chosen to be operated together with a frequency per period.

The *change&go network* was introduced by [25] and can be used to solve the line planning problem with the objective of minimizing the travel time or for routing passengers with respect to fixed line frequencies. Figures 2a and 2b visualize the construction process for a line pool with three lines.

► **Definition 3.5** (Change&Go Network). For a line-based modality $m \in \mathcal{M}^{\mathcal{L}}$ with $\text{PTN}_m = (V_m, E_m)$ and a line pool $\mathcal{L}_m$, define the directed simple graph $\text{CGN}_m = (\mathfrak{N}_m, \mathfrak{E}_m)$ as the *change&go network* of modality m with line pool $\mathcal{L}_m$.

Each stop from the PTN_m is represented by a platform node in the CGN_m . For each line $l \in \mathcal{L}_m$ and each stop $\nu \in V_m$ contained in the line, the CGN_m has a corresponding driving node, i.e., $\mathfrak{N}_m := \mathfrak{N}_m^p \cup \mathfrak{N}_m^d$ with

- platform nodes $\mathfrak{N}_m^p := \{(\nu, 0, m) : \nu \in V_m\}$ and
- driving nodes $\mathfrak{N}_m^d := \{(\nu, l, m) : l \in \mathcal{L}_m, \nu \in l\}$.

There are two different types of arcs in the CGN_m : Driving arcs connecting consecutive driving nodes of the same line and platform arcs connecting driving nodes to the platform node that corresponds to the same stop $\nu \in V_m$. Formally, let $\mathfrak{E}_m := \mathfrak{E}_m^p \cup \mathfrak{E}_m^d$ with

- platform arcs $\mathfrak{E}_m^p := \{((\nu, 0, m), (\nu, l, m)), ((\nu, l, m), (\nu, 0, m)) : l \in \mathcal{L}_m, \nu \in l\}$ and
- driving arcs $\mathfrak{E}_m^d := \bigcup_{l \in \mathcal{L}_m} \mathfrak{E}_m^l$, where
- $\mathfrak{E}_m^l := \{((\nu_1, l, m), (\nu_2, l, m)) : (\nu_1, \nu_2) \in l\}$ are the driving arcs of every line $l \in \mathcal{L}_m$.

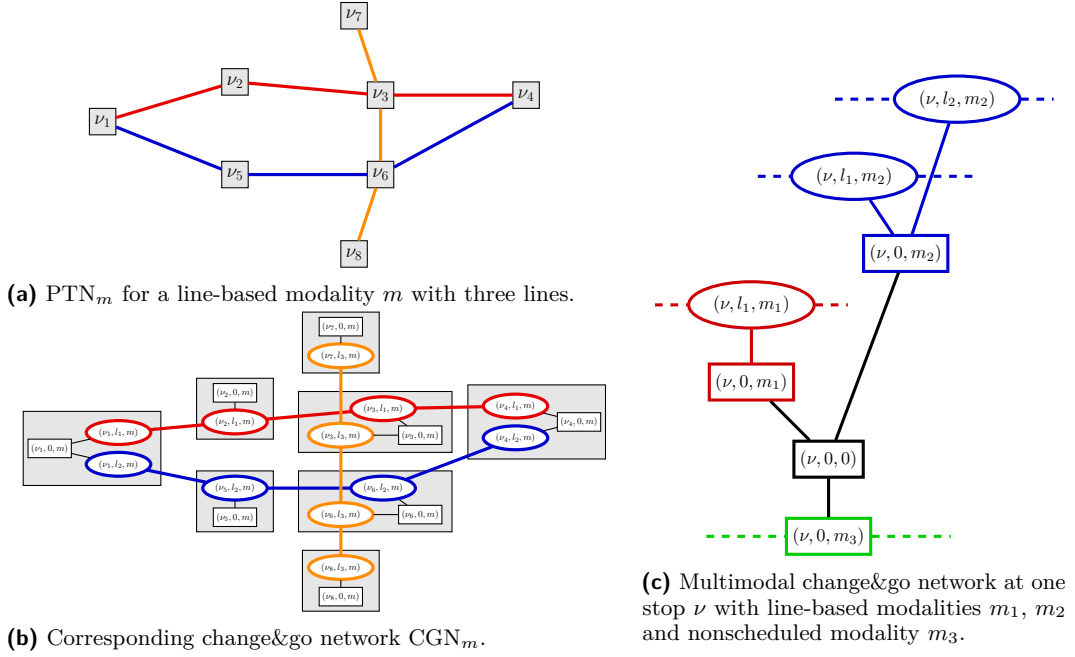
Multimodal Change&Go Network

In the multimodal case, transfers between modalities also need to be taken into account. Analogously to the multimodal public transport network for a set of unimodal public transport networks, we can define a multimodal change&go network for a set of unimodal change&go networks by stacking them as shown in Figure 2c. For the nonscheduled modalities we do not have a change&go network, since there are no transfers within a nonscheduled modality. Hence, they can still be represented by their public transport network instead. For ridepooling modalities we use the areas where it is offered instead of the lines in a line-based modality. In the following, we restrict ourselves to line-based and nonscheduled modalities, though the presented framework is capable of incorporating ridepooling as well. The multimodal change&go graph with ridepooling modalities is presented in [7].

► **Definition 3.6** (Multimodal Change&Go Network). For an infrastructure network $\text{ISN} = (N, \Lambda)$ with modalities $\mathcal{M}$, each with a $\text{PTN}_m = (V_m, E_m)$, and for each line-based modality $m \in \mathcal{M}^{\mathcal{L}}$ a line pool $\mathcal{L}_m$ and $\text{CGN}_m = (\mathfrak{N}_m, \mathfrak{E}_m)$, we define the *multimodal change&go network* $\text{MCGN} = (\mathfrak{N}, \mathfrak{E})$.

We use the nodes $\mathfrak{N}_m$ from the unimodal change&go networks for the line-based modalities. Additionally, for each infrastructure node $\nu \in N$, define a station node $(\nu, 0, 0)$. Each stop $\nu \in V_m$ of a nonscheduled modality $m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}$ is represented by a nonscheduled platform node $(\nu, 0, m)$. Formally, define $\mathfrak{N} := (\bigcup_{m \in \mathcal{M}^{\mathcal{L}}} \mathfrak{N}_m) \cup \mathfrak{N}^N \cup (\bigcup_{m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}} \mathfrak{N}_m)$ with

- station nodes $\mathfrak{N}^N := \{(\nu, 0, 0) : \nu \in N\}$ and
 - nonscheduled platform nodes $\mathfrak{N}_m = \{(\nu, 0, m) : \nu \in V_m\}$ for $m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}$.
- To allow for transfers between different modalities, we introduce station arcs, connecting each platform node (line-based or nonscheduled) to their respective station node. Furthermore, for each edge $e = (\nu_1, \nu_2) \in E_m$ of a nonscheduled modality $m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}$, we introduce an arc in the MCGN connecting the two nonscheduled platform nodes. Define $\mathfrak{E} := (\bigcup_{m \in \mathcal{M}^{\mathcal{L}}} \mathfrak{E}_m) \cup \mathfrak{E}^N \cup (\bigcup_{m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}} \mathfrak{E}_m)$ with
- station arcs $\mathfrak{E}^N := \{((\nu, 0, 0), (\nu, 0, m)), ((\nu, 0, m), (\nu, 0, 0)) : \nu \in N, m \in \mathcal{M}(\nu)\}$ and
 - nonscheduled arcs $\mathfrak{E}_m := \{((\nu_1, 0, m), (\nu_2, 0, m)) : (\nu_1, \nu_2) \in E_m\}$ for $m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}$.



■ **Figure 2** Construction of a unimodal change&go network (left) and a multimodal change&go network (right). Each edge is directed and exists in both orientations.

Travel Times

In a unimodal change&go network, a passenger's route between two stops $\nu_1, \nu_2 \in V_m$ is represented by a path connecting the corresponding platform nodes $(\nu_1, 0, m), (\nu_2, 0, m) \in \mathfrak{N}_m$. It consists of both driving and platform edges, providing information not only about the route in the underlying road or track network, but also about which lines are used and where the passenger transfers between lines.

In a multimodal change&go network, passenger paths connect the corresponding station nodes $(\nu_1, 0, 0), (\nu_2, 0, 0) \in \mathfrak{N}^N$ and provide the additional information of transfers between different modalities. The travel times on the driving edges equal those on the corresponding PTN_m edge. The transfer times depend on the timetable, which, at the line planning stage of the planning process, is not known yet, hence, we instead put a fixed time penalty for each transfer within a modality or between different modalities.

► **Definition 3.7** (CGN travel times). For each driving arc $e \in \mathfrak{E}_m^d$ denote its corresponding edge in the underlying PTN_{*m*} by $E_m(e)$. Let t_m^{trans} be the time penalty for a transfer between two lines of modality $m \in \mathcal{M}^{\mathcal{L}}$ and t^{trans} be the time penalty for a transfer between two different modalities. Let $e \in \mathfrak{E}$ and define its travel time as

$$\tau_e := \begin{cases} t_{E_m(e)} & , \text{ if } e \in \mathfrak{E}_m^d \text{ for some } m \in \mathcal{M}^{\mathcal{L}}, \\ \frac{1}{2}t_m^{\text{trans}} & , \text{ if } e \in \mathfrak{E}_m^p \text{ for some } m \in \mathcal{M}^{\mathcal{L}}, \\ t_{E_m(e)} & , \text{ if } e \in \mathfrak{E}_m \text{ for some } m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}, \\ \frac{1}{2}t^{\text{trans}} & , \text{ if } e \in \mathfrak{E}^N. \end{cases}$$

The travel time on a path p in a CGN_{*m*} of modality m or in the MCGN can be computed as the sum of the travel times of the arcs in the path $\tau(p) = \sum_{e \in p} \tau_e$. Note that, similar to the multimodal public transport network, the travel time of a passenger's path includes two "additional" platform and station edges for the initial boarding and the final alighting.

Line Planning in the Change&Go Network

The task in line planning is to choose lines from the line pool to be operated and determine their frequencies.

► **Definition 3.8** (Line Concept). Let $m \in \mathcal{M}^{\mathcal{L}}$ be a line-based modality, with line pool $\mathcal{L}_m$ and denote by $f_l \in \mathbb{N}$ for each line $l \in \mathcal{L}_m$ its *frequency*, i.e., the number of times the line is operated within a given planning period T . We call the tuple $(\mathcal{L}_m, f_m)$, with $f_m = (f_l)_{l \in \mathcal{L}_m} \in \mathbb{N}^{|\mathcal{L}_m|}$, a *line concept of modality m* .

There are many different models for line planning, with various objective functions and constraints for feasible solutions. In this work, we assume that the cost of operating a line $l \in \mathcal{L}_m$ with frequency $f_l \in \mathbb{N}$ is f_l times the cost of the line c_l , which is fixed and given. In the classic unimodal line planning problem, we need passenger demand for this modality, e.g., as an OD matrix $\mathcal{OD}^m \in \mathbb{N}^{|V_m| \times |V_m|}$ specifying the number of passengers traveling between two stops in PTN_{*m*} with the modality m . The passenger paths highly depend on the actual design of the system, which is not known in advance, hence the modal split can only be an estimation.

In the travel time model from [25], the passengers in $\mathcal{OD}^m$ form a multi-commodity flow in the CGN_{*m*} between the platform nodes. The frequencies have to be chosen in a way that accommodates the passenger flow on the driving arcs without exceeding a budget constraint on the overall cost of the line system, while the sum of passengers' travel times is minimized. In the multimodal line planning problem we want to determine line concepts for all line-based modalities simultaneously. The multimodal change&go network allows to apply the travel time model by routing the passengers of $\mathcal{OD}$ as a multi-commodity flow between the station nodes. Hence, the modal split is not fixed in advance but emerges endogenously from the line concepts determined by the optimization.

3.4 Event-Activity Network

In the timetabling step of the planning process, the task is to determine a time for each arrival and departure of a vehicle at a station. A vehicle's journey along a line can be represented by a sequence of departures and arrivals at the stops contained in the line.

► **Definition 3.9** (Event-Activity Network). For $m \in \mathcal{M}^{\mathcal{L}}$ with PTN_{*m*} = (V_m, E_m) and line concept $(\mathcal{L}_m, f_m)$, let

$$\mathcal{T}_m = \bigcup_{l \in \mathcal{L}_m} \{l^{(1)}, \dots, l^{(f_l)}\}$$

be the set of all trips of the line concept. A line l with frequency $f_l \in \mathbb{N}$ is operated f_l times within the planning period, and is therefore represented by f_l separate trips. Then, define the *event-activity network* as the directed simple graph $\text{EAN}_m = (\mathcal{E}_m, \mathcal{A}_m)$. The events $\mathcal{E}_m$ represent arrivals and departures of vehicles at stops or stations, i.e., $\mathcal{E}_m := \mathcal{E}_m^{\text{arr}} \cup \mathcal{E}_m^{\text{dep}}$ with

- $\mathcal{E}_m^{\text{arr}} := \{(v, t, \text{arr}, m) : t \in \mathcal{T}_m, v \in V_m \cap t\},$
- $\mathcal{E}_m^{\text{dep}} := \{(v, t, \text{dep}, m) : t \in \mathcal{T}_m, v \in V_m \cap t\}.$

The activities connect consecutive events and represent vehicles driving between or waiting at stations or possible transfers for passengers between trips at a station. Define $\mathcal{A}_m := \mathcal{A}_m^{\text{drive}} \cup \mathcal{A}_m^{\text{wait}} \cup \mathcal{A}_m^{\text{trans}}$, where

- $\mathcal{A}_m^{\text{drive}} := \{((v_1, t, \text{dep}, m), (v_2, t, \text{arr}, m)) : t \in \mathcal{T}_m, (v_1, v_2) \in E_m \cap t\},$
- $\mathcal{A}_m^{\text{wait}} := \{((v, t, \text{arr}, m), (v, t, \text{dep}, m)) : t \in \mathcal{T}_m, v \in V_m \cap t\},$
- $\mathcal{A}_m^{\text{trans}} := \{((v, t_1, \text{arr}, m), (v, t_2, \text{dep}, m)) : t_1, t_2 \in \mathcal{T}_m, v \in E_m \cap t_1 \cap t_2\}.$

Additionally, event-activity networks can be extended with further activity types beyond the basic ones presented here; we refer to [15, 18] for more advanced examples.

Multimodal Event-Activity Network

For the timetabling stage in the multimodal case, the set of events that need to be scheduled is the union of all sets of events from the unimodal line-based cases. But, additionally, we need to consider the possibility that passengers might walk between stops that are within acceptable distance of each other. Hence, we need to introduce additional activities that represent the use of nonscheduled modalities.

► **Definition 3.10** (Multimodal Event-Activity Network). Let $\mathcal{M}$ be a set of modalities with an event activity network $\text{EAN}_m = (\mathcal{E}_m, \mathcal{A}_m)$ for each line-based modality $m \in \mathcal{M}^{\mathcal{L}}$ and denote by $D_m > 0$ for each nonscheduled modality $m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}$ the maximal duration of a leg of this modality in a passenger's journey. Define for a nonscheduled modality $m \in \mathcal{M} \setminus \mathcal{M}^{\mathcal{L}}$ its set of activities

$$\mathcal{A}_m := \left\{ \left((v_1, t_1, \text{arr}, m_1), (v_2, t_2, \text{dep}, m_2) \right) : \right. \\ \left. (v_1, t_1, \text{arr}, m_1) \in \mathcal{E}_{m_1}, (v_2, t_2, \text{dep}, m_2) \in \mathcal{E}_{m_2} \text{ with } \text{dist}_m(v_1, v_2) \leq D_m \right\},$$

where $\text{dist}_m(v_1, v_2)$ denotes the shortest path distance in PTN_m with respect to the edge travel times t_e . Define the *multimodal event-activity network* $\text{MEAN} = (\mathcal{E}, \mathcal{A})$, with $\mathcal{E} := \bigcup_{m \in \mathcal{M}^{\mathcal{L}}} \mathcal{E}_m$ and $\mathcal{A} := \bigcup_{m \in \mathcal{M}} \mathcal{A}_m$.

To respect shared infrastructure capacities in railway applications, additional headway activities derived from the infrastructure network can be included.

Timetabling in the Event-Activity Network

Consecutive departures and arrivals cannot be scheduled arbitrarily, there are restrictions on the amount of time that needs to pass between them: vehicles need to wait for some time at each stop such that passengers can board and alight, and there is an amount of time needed for driving from one station to the next. These constraints need to be taken into account when determining a timetable. In this work we focus on periodic timetabling, but the following definitions can be easily extended to the aperiodic case.

► **Definition 3.11** (Timetable). Given an event-activity network $\text{EAN}_m = (\mathcal{E}_m, \mathcal{A}_m)$ of a line-based modality $m \in \mathcal{M}^{\mathcal{L}}$ and a planning period $T \in \mathbb{N}$, a *periodic timetable* $\pi^m = (\pi_e^m)_{e \in \mathcal{E}_m} \in \mathbb{N}^{|\mathcal{E}_m|}$ fixes a point in time $\pi_e^m \in \{0, \dots, T-1\}$ for each event $e \in \mathcal{E}_m$ which

occurs in every period repetition throughout the day.

Suppose that for each activity $a \in \mathcal{A}_m$ we have lower and upper bounds $L_m(a), U_m(a) \in \mathbb{N}$ on its duration. A periodic timetable π^m is called *feasible*, if for each activity $a = (e_1, e_2) \in \mathcal{A}_m$ the periodic difference in time between the two events is within the given bounds, i.e.

$$L_m(a) \leq ((\pi_{e_2}^m - \pi_{e_1}^m - L_m(a)) \bmod T) + L_m(a) \leq U_m(a).$$

The usual objective for the timetabling step is to minimize the overall passenger-weighted travel time. Since the passenger paths, and thus their travel times, depend on the timetable, which is not known beforehand, we need an estimate c_a of how many passengers will use activity $a \in \mathcal{A}_m$. To obtain these estimates, one can do a passenger prerouting in EAN_m with respect to some duration assumption $\tilde{t}_a$ for all $a \in \mathcal{A}_m$. Finding an optimal timetable is known as the *Periodic Event Scheduling Problem* (PESP) [26].

Timetabling in the Multimodal Event-Activity Network

For a nonscheduled activity $a = (e_1, e_2) \in \mathcal{A}_m, m \in \mathcal{M} \setminus \mathcal{M}^L$ connecting two events at stops v_1 and v_2 , we define the time bounds $L_m(a) := \text{dist}_m(v_1, v_2), U_m(a) := T + \text{dist}_m(v_1, v_2) - 1$. The task in the multimodal timetabling step is to determine a periodic timetable $\pi = (\pi^m)_{m \in \mathcal{M}^L} \in \mathbb{N}^{|\mathcal{E}|}$ respecting all activity bounds which minimizes the travel time for given passenger weights c_a on each activity $a \in \mathcal{A}$. Note that the bounds on the nonscheduled activities do not impose any restrictions on the times of the incident events, but they are needed to calculate the travel time of a passenger in the objective function correctly.

The resulting problem is still a PESP instance and all solution methods known for the unimodal case can be applied.

4 Applications

In the following we present multiple possible applications for our framework. To demonstrate its practical relevance, we show a case study with real-world data for the first application.

Multimodal Sequential Planning. As already mentioned above, the supplies of modalities mutually influence each other, as it determines the modal split. Therefore, it is reasonable to perform the sequential planning stages like line planning and timetabling not for a single line-based modality in isolation, but to consider all line-based modalities in an integrated manner. Furthermore, one can also fix individual modalities, plan the remaining modalities while using the fixed ones only for passenger routing, and measure its influence on the fixed modalities.

Multiple Vehicle Types. In most public transport systems there are different vehicle types operating, e.g., small and articulated buses or slower and faster trains. Especially in the line planning step, the passenger capacity of a vehicle and its speed are crucial. Furthermore, different vehicle types might not be allowed to operate on every PTN edge due to height limitations, narrow corners or weight limitations. One can model each different vehicle type as a single modality m having a specific passenger capacity A_m , a specific speed $\text{speed}(m)$ and an individual line pool $\mathcal{L}_m$. Multimodal planning methods can then be used to plan lines, timetables and vehicle schedules when vehicles of different types are present.

Strategic Ridepooling. Ridepooling systems are demand-responsive, i.e., they react spontaneously to the passenger demand that arises in each situation, while in line-based systems, the vehicles travel along fixed lines according to a pre-determined timetable. This makes line systems very good at transporting large numbers of passengers while ridepooling tends to be more efficient in low-demand situations. Hence, planning both modalities simultaneously leads to integrated solutions where each is utilized to make use of its strengths and advantages. First models for the strategic aspects of ridepooling, meaning the decisions of where to offer a ridepooling service and with which capacity, have been presented in [8]. Analogously to the line planning step, the task is to choose ridepooling areas and a number of ridepooling vehicles to be deployed in each area. The proposed framework allows to combine strategic ridepooling with line planning for the integrated planning of a multimodal system consisting of line-based modalities and ridepooling services, while nonscheduled modalities are also available for passengers.

Planning Infrastructure for Nonscheduled Modalities. Starting from an infrastructure network and the information for each infrastructure link, which modality can (in principle) be established on it and in which direction, our framework can be used to actually plan a PTN, i.e., to decide for stops to be opened and for each connection between them, which infrastructure path it uses. For nonscheduled modalities, this would mean to decide where bike lanes or footpaths should be built. Our framework allows to integrate the effect on other modalities into the decision or to plan infrastructure for multiple modalities at once.

Modelling Shared Infrastructure within one Modality. Especially in railway applications shared infrastructure capacities are an important problem, even within a single modality (see, e.g., [17]). If for example two trains leave a station in the same direction, a headway distance must be respected for security reasons, but if their paths split at some point before their next station, the headway is no longer needed on the remaining track segments. Therefore, the infrastructure network can be used to model and even plan track occupations.

Integration of Walking to Line-based Planning Problems. When planning a line-based modality in the classic unimodal framework, one usually considers an OD matrix, specifying the number of passengers that wish to travel between pairs of stops. This is done by assigning the actual demand points (homes, schools, workplaces etc.) to the closest stop. However, depending on the resulting system, it might shorten the travel time of passengers to walk to another stop to save travel time or transfers. Similarly, walking might also be reasonable between two trips with a line vehicle within a passenger's journey. Our framework allows to consistently include the possibility of walking into the planning of line-based systems.

4.1 Case study

We present a case study using real-world data of the city of Helsinki, Finland. We consider five line-based modalities and one nonscheduled modality and perform the sequential planning process starting with an infrastructure network and an OD matrix on infrastructure level. The goal is to plan line concepts and timetables for all line-based modalities by applying multimodal planning methods in the three steps of line planning, passenger-prerouting to get EAN weights, and timetabling.

All experiments are conducted within LinTim [23]; an open-source software toolbox for public transport planning that provides a collection of algorithms covering the full sequential planning pipeline. We have extended LinTim with a multimodal planning module, implementing all

graph structures described in Section 3 and making them available for use in each planning step. The calculations were performed on an Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz with 380 GB of memory and we used Gurobi 13.0.1 [11] as IP solver.

4.1.1 Dataset

We use real-world data from the city of Helsinki, Finland, for this numerical study. The Helsinki dataset from the LinTim toolbox [23] was cropped to include the city center of Helsinki as shown in Figure 6 in Appendix A. The resulting infrastructure network consists of 195 nodes and 2996 links. The OD matrix is based on Helsinki Region Transport (HSL) open data of passenger boarding in November 2016 and contains 49,544 passengers. We consider the five line-based modalities *bus*, *tram*, *rail*, *subway*, and *ferry* as well as the nonscheduled modality *walk*. Line pools for all line-based modalities are generated by computing a shortest path for each station pair of the corresponding modality and eliminating subpaths. The costs c_l of a line l are computed as $c_l = \text{cost}_{\text{fix}} + \sum_{e \in l} \sum_{\lambda \in \text{infram}(e)} \text{length}(\lambda) \cdot \text{cost}_{\text{length}}$. Table 1 in Appendix A gives an overview of the modalities' characteristics.

4.1.2 Methodology

In a traditional unimodal planning procedure we would consider every line-based modality independently of the other modalities. To apply unimodal planning, we need a preprocessing step to split the OD matrix on infrastructure level $\mathcal{OD}$ in modality-specific OD matrices $\mathcal{OD}^m \in \mathbb{N}^{|V_m| \times |V_m|}$ specifying the number of passengers traveling between two stops in PTN $_m$ with the modality m . To achieve an OD split, we route all passengers on shortest paths in the MPTN with respect to the edge travel times t_e and split up the resulting paths as described in Algorithm 1 in Appendix B. Note that the sum of passengers included in the OD matrices $\mathcal{OD}^m$ for all $m \in \mathcal{M}$ can be larger than the sum of passengers included in $\mathcal{OD}$, since a single passenger can appear in the OD matrix of multiple modalities or even multiple times within the same modality.

In the line planning step we use the travel time model as formulated by [25] and solve the IP formulation with Gurobi. This model minimizes the overall travel time with integrated passenger routing in the change&go network while respecting vehicle capacities and a total cost budget. To get a reasonable cost budget for each modality, we first solve the cost-model for line planning as introduced in [28] and increase its optimal objective value by 10%. Applying this procedure separately for each line-based modality m with CGN $_m$ and $\mathcal{OD}^m$ as input gives a line concept $(\mathcal{L}_m, f_m)$ for each modality.

The travel time model can also be applied in a multimodal setting using the MCGN as graph structure. We use the sum of the unimodal cost budgets as budget for the multimodal model and the original OD matrix $\mathcal{OD}$ as input. To model shared infrastructure capacities correctly, we apply linear constraints of the form of Equation (1). Again, we end up with line concepts for all modalities.

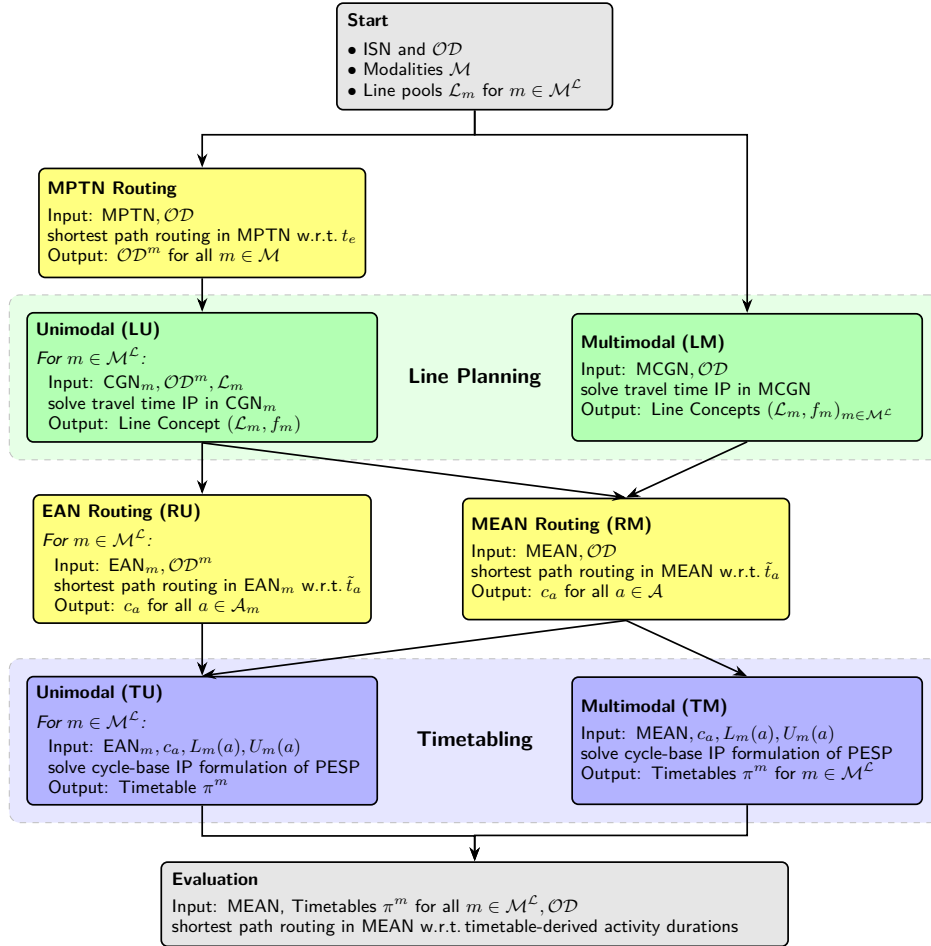
The timetabling step requires passenger weights for all activities and we consider two ways to construct them. In the traditional unimodal approach one takes the modality-specific OD matrix $\mathcal{OD}^m$ and routes all passengers in the EAN $_m$ on shortest paths with respect to estimated activity durations $\hat{t}_a$. In a multimodal approach we build the MEAN and route all passengers in the OD matrix $\mathcal{OD}$ on shortest paths with respect to estimated activity durations. This allows passengers to use different modalities compared to the first prerouting in the MPTN and leads to a different modal split. Since transfer times can now be estimated more accurately from the line frequencies, this new routing is more realistic.

In the timetabling step we minimize the overall travel time with respect to the fixed precomputed passenger routing while satisfying all activity time bounds. We use the cycle-base IP formulation of the resulting PESP (from [19], see also [14]). The traditional unimodal approach solves an IP for each line-based modality independently using the modality's EAN_m , the lower and upper activity time bounds $L_m(a), U_m(a)$ and the passenger weights c_a from the previous routing step as input. The multimodal approach solves one single IP with the MEAN as underlying graph structure.

This gives five possible ways to get from the infrastructure network and the OD matrix to timetables for all modalities. Figure 3 visualizes our procedure and the five described paths where we combine unimodal and multimodal planning methods in different steps.

To evaluate a solution, we do a final routing in the MEAN with respect to the actual timetable-derived activity durations, delivering final passenger paths and travel times.

Note that we use warm starts for the IP solver whenever possible, e.g., the solution of the line planning cost-model is used as starting solution for the line planning travel time model and already computed timetables are used as initial solutions in the timetabling step, whenever they are based on the same line concept.



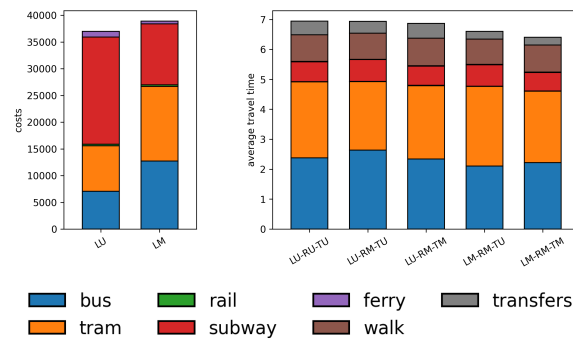
■ **Figure 3** Schematic overview of the sequential planning framework for the case study. Starting from an infrastructure network and OD matrix, five combinations of unimodal and multimodal planning steps in line planning, passenger prerouting, and timetabling yield timetables for all line-based modalities, evaluated via a final routing in the MEAN.

4.1.3 Results

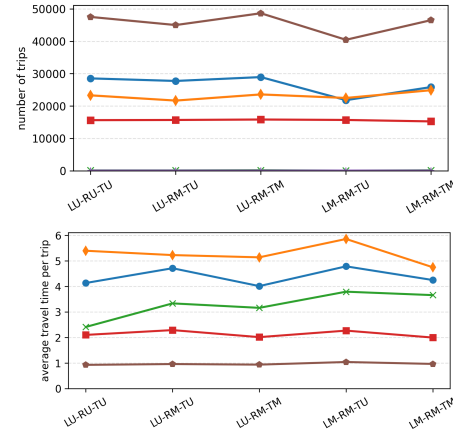
An overview of all numerical results can be found in Table 2 in Appendix C.

The left part of Figure 4 shows the total costs of the line concepts resulting from unimodal and multimodal planning and how much each modality contributes. The costs for the line concepts resulting from multimodal planning are 5.2% higher, since the travel time model for line planning allows to increase the costs up to the given budget when decreasing the (estimated) travel time. This is possible in the multimodal model since passengers can be re-assigned to use different modalities. Within the line concept resulting from multimodal planning, the cost shares of bus and tram increase relative to the unimodal baseline, while those of subway and ferry decrease. This redistribution of resources reflects the jointly optimized passenger flows across all modalities.

Regarding the average travel time of a passenger depicted in the right part of Figure 4, we can see a clear monotone trend: the overall travel time decreases from left to right, i.e., the more multimodal aspects are involved during the planning process, the better the average travel time gets in the end. Most strikingly, the time passengers spend on intermodal transfers drops by more than 30% as soon as multimodal line planning is applied, indicating that the integrated MCGN formulation produces line concepts that are inherently better coordinated across modalities.



■ **Figure 4** Modal split for costs (left) and average travel time (right).



■ **Figure 5** Trips per modality (top) and average travel time per trip (bottom).

However, the modality shares in the right part of Figure 4 show the time an average passenger spends within one modality during his entire journey. If this share decreases, it may either come from an improvement of service for this modality or reflect that less passengers use this modality and switched to another. To disentangle these effects, Figure 5 reports the number of trips per modality recorded in the final MEAN routing. Note that the sum of trips across all modalities exceeds the total number of passengers, since a single passenger journey may involve multiple modalities and is therefore counted once per leg. The upper part of Figure 5 reveals substantial shifts in the distribution of trips across experimental configurations: in particular, the number of trips for bus, tram and walk varies considerably, indicating that multimodal planning redistributes passengers across modalities rather than merely accelerating existing routes. Turning to the lower part of Figure 5, which shows the average travel time per trip and modality, the best results are achieved under multimodal timetabling, except for the rail modality.

Table 3 in Appendix C reports the optimization runtimes for line planning and timetabling, where entries without a gap value indicate that the solver reached optimality within the time limit of 6 hours. We can observe that the unimodal models are solved to optimality in negligible time (below 0.02 s) for the three “small” modalities rail, subway and ferry, while especially unimodal line planning was challenging for the modalities bus and tram, reaching only an optimality gap of 11.45% and 10.93%, respectively.

Notably, the multimodal models achieve optimality gaps comparable to their unimodal counterparts. We attribute this to the systematic use of warm starts throughout all experiments, which proves effective in practice, yielding substantially shorter runtimes for the multimodal models than a cold-start baseline would suggest. The increased runtime for unimodal timetabling after unimodal line planning in the bus modality reflects two factors: the resulting event-activity network contains twice as many events and five times as many activities as the tram event-activity network, and since the full unimodal planning served as the baseline, no warm starts could be exploited.

5 Conclusion and Further Research

We presented a general framework for multimodal public transport planning. We introduced the three categories line-based, ridepooling, and nonscheduled for transport modalities and proposed a hierarchy of graph structures that extend their well-known unimodal counterparts to the multimodal setting. These structures explicitly model interactions between modalities, including passenger transfers and shared infrastructure capacity. We showed that established optimization models for unimodal line planning and periodic timetabling extend naturally to the multimodal setting using these graph structures.

Our computational study on the Helsinki dataset demonstrates that incorporating multimodal aspects into the planning process leads to measurable improvements in passenger travel time. In particular, the results show that multimodal line planning significantly reduces the time passengers spend on intermodal transfers, and that a fully integrated multimodal planning pipeline consistently outperforms unimodal approaches. At the same time, the multimodal timetabling step can be solved as a single PESp instance.

Several directions for future research remain open. First, the integration of ridepooling into the timetabling stage is a natural next step: while strategic ridepooling has already been incorporated into line planning, the operative aspects of ridepooling, such as vehicle routing and assignment, are demand-responsive and do not fit directly into the periodic timetabling framework. Developing suitable models that bridge the strategic and operative levels of ridepooling within the sequential planning pipeline is therefore an important open problem.

Second, the multimodal IP formulations are substantially larger than their unimodal counterparts, and dedicated solution methods are needed to make them tractable at scale. One promising approach is to solve the multimodal problem iteratively by fixing the line concepts or timetables of individual modalities in turn and re-optimizing the remaining ones, exploiting the decomposable structure of the multimodal graphs.

Third, the planning of transport infrastructure has been studied for multiple line-based modalities in [20]. Our framework allows to extend the planning of infrastructure to other modality categories. In particular the planning problem for nonscheduled modalities, i.e., determining where bike lanes or footpaths should be built, has so far only been sketched as a potential application of the framework. Formulating this as a concrete optimization problem and studying its interaction with the planning of line-based and ridepooling modalities constitutes a promising research direction.

References

- 1 M.K. Anderson, O.A. Nielsen, and C.G. Prato. Multimodal route choice models of public transport passengers in the greater copenhagen area. *EURO Journal on Transportation and Logistics*, 6(3):221–245, 2017. doi:10.1007/s13676-014-0063-3.
- 2 H. Bast, D. Delling, A. V. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route Planning in Transportation Networks. In L. Kliemann and P. Sanders, editors, *Algorithm Engineering: Selected Results and Surveys*, volume 9220 of *LNCS State of the Art*. Springer, 2016. doi:10.1007/978-3-319-49487-6_2.
- 3 D. Bertsimas, Y. Sian Ng, and J. Yan. Joint frequency-setting and pricing optimization on multimodal transit networks at scale. *Transportation Science*, 54(3):839–853, 2020. doi:10.1287/TRSC.2019.0959.
- 4 O. Cats, K. Gkiotsalitis, and A. Schöbel. 50 years of Operations Research in public transport. *EJTL*, 14:100160, 2025.
- 5 A. Ceder and N.H.M. Wilson. Bus network design. *Transportation Research B*, 20(4):331–344, 1986.
- 6 J.M Dibbelt. *Engineering algorithms for route planning in multimodal transport networks*. PhD thesis, Karlsruhe Institute for Technology, 2016.
- 7 L. Dittrich, S. Jäger, and A. Schöbel. Integrated line planning and strategic ridepooling: Minimizing the passengers’ travel time. Working paper, 2026.
- 8 L. Dittrich, M. Rihlmann, S. Roth, and A. Schöbel. A Model for Strategic Ridepooling and Its Integration with Line Planning. In J. Sauer and M. Schmidt, editors, *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2025)*, volume 137 of *Open Access Series in Informatics (OASICS)*, pages 16:1–16:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.ATMOS.2025.16.
- 9 R. Elbert, J.P. Müller, and J. Rentschler. Tactical network planning and design in multimodal transportation – a systematic literature review. *Research in Transportation Business & Management*, 35:100462, 2020. Modal shift and logistics integration in intermodal transport networks’ in Research in Transportation, Business and Management. doi:10.1016/j.rtbm.2020.100462.
- 10 K. Giannakopoulou, A. Paraskevopoulos, and C. Zaroliagis. Multimodal dynamic journey-planning. *Algorithms*, 12(10), 2019. doi:10.3390/a12100213.
- 11 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL: <https://www.gurobi.com>.
- 12 M.C. Jurado, C. Canudas de Wit, G. De Nunzio, and M. Salazar. On transport justice and safety in bicycle network design optimization. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pages 4958–4963. IEEE, 2025.
- 13 Y. Ke, X. Wu, L. Nie, Z. Yao, and Y. Chen. Synchronizing train, aircraft, shuttle, and passenger flows in intermodal timetabling: A time–space network-based formulation and a decomposition algorithm using alternating direction method of multipliers. *Transportation Research Part C: Emerging Technologies*, 159:104464, 2024. doi:10.1016/j.trc.2023.104464.
- 14 C. Liebchen. *Periodic Timetable Optimization in Public Transport*. dissertation.de – Verlag im Internet, Berlin, 2006.
- 15 C. Liebchen and R. Möhring. The modeling power of the periodic event scheduling problem: railway timetables — and beyond. In *Algorithmic Methods for Railway Optimization*, number 4359 in *Lecture Notes in Computer Science*, pages 3–40. Springer, 2007.
- 16 B. Liu, Y. Ji, and O. Cats. Integrating ride-hailing services with public transport: a stochastic user equilibrium model for multimodal transport systems. *Transportmetrica A: Transport Science*, 21(1):329–357, 2023.
- 17 R. Lusby, J. Larsen, D. Ryan, and M. Ehrgott. Routing trains through railway junctions: A new set packing approach. *Technical report*, 21, 2006.

- 18 B. Masing, N. Lindner, and C. Liebchen. Integrating line planning for construction sites into periodic timetabling via track choice. In *23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023)*, volume 115, pages 5:1–5:15, 2023. doi:10.4230/OASIcs.ATMOS.2023.5.
- 19 K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. PhD thesis, University of Hildesheim, Hildesheim, Germany, 1998.
- 20 F. Perea, M.B.C. Menezes, J.A. Mesa, and F. Rubio-Del-Rey. Transportation infrastructure network design in the presence of modal competition: computational complexity classification and a genetic algorithm. *TOP*, 28(2):442–474, 2020.
- 21 M. Salazar, S. Betancur Giraldo, F. Paparella, L. Pedroso, and K. Martens. Mobilizing transport justice: a sufficientarian optimization framework for intermodal mobility systems. *NPJ Sustainable Mobility and Transport*, 2(1):43, 2025.
- 22 P. Schiewe and A. Schöbel. Integrated optimization of sequential processes: General analysis and application to public transport. *EURO Journal on Transportation and Logistics*, 11:100073, 2022. doi:10.1016/j.ejtl.2022.100073.
- 23 P. Schiewe, A. Schöbel, O. Herrala, M. Rihlmann, S. Roth, S. Albert, C. Biedinger, T. Dahlheimer, L. Dittrich, K. Hoffmann, S. Jäger, P. Pattanaik, A. Schiewe, M. Stinzenhöfer, and R. Urban. Documentation for lintim 2025.11, 2025. URL: <https://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-93118>.
- 24 A. Schöbel. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research C*, 74:348–365, 2017. doi:10.1016/j.trc.2016.11.018.
- 25 A. Schöbel and S. Scholl. Line planning with minimal travel time. In *ATMOS 2005*, number 06901 in Dagstuhl Seminar Proceedings, 2006.
- 26 P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2:550–581, 1989. doi:10.1137/0402049.
- 27 J. Zimmer, M. Friedrich, and A. Schöbel. Suitability of different transport means as a function of travel demand: Pareto-optimal solutions (in german). *Straßenverkehrstechnik*, pages 796–805, 2024. doi:10.53184/SVT10-2024-3.
- 28 P.J. Zwaneveld, M.T. Claessens, and N.M. van Dijk. A new method to determine the cost optimal allocation of passenger lines. In *Defence or Attack: Proceedings of 2nd TRAIL PhD Congress 1996, Part 2*, Delft/Rotterdam, 1996. TRAIL Research School.

A Helsinki Dataset

■ **Table 1** Overview of the modalities’ characteristics within the Helsinki dataset.

	$ V_m $	$ E_m $	$ \mathcal{L}_m $	vehicle capacity	cost _{fix}	cost _{length}
bus	147	347	1399	80	68	0.6
tram	93	207	436	150	160	4.5
rail	3	2	1	300	220	10
subway	9	14	3	200	300	14
ferry	4	4	4	225	250	6
walk	195	4842	-	-	-	-

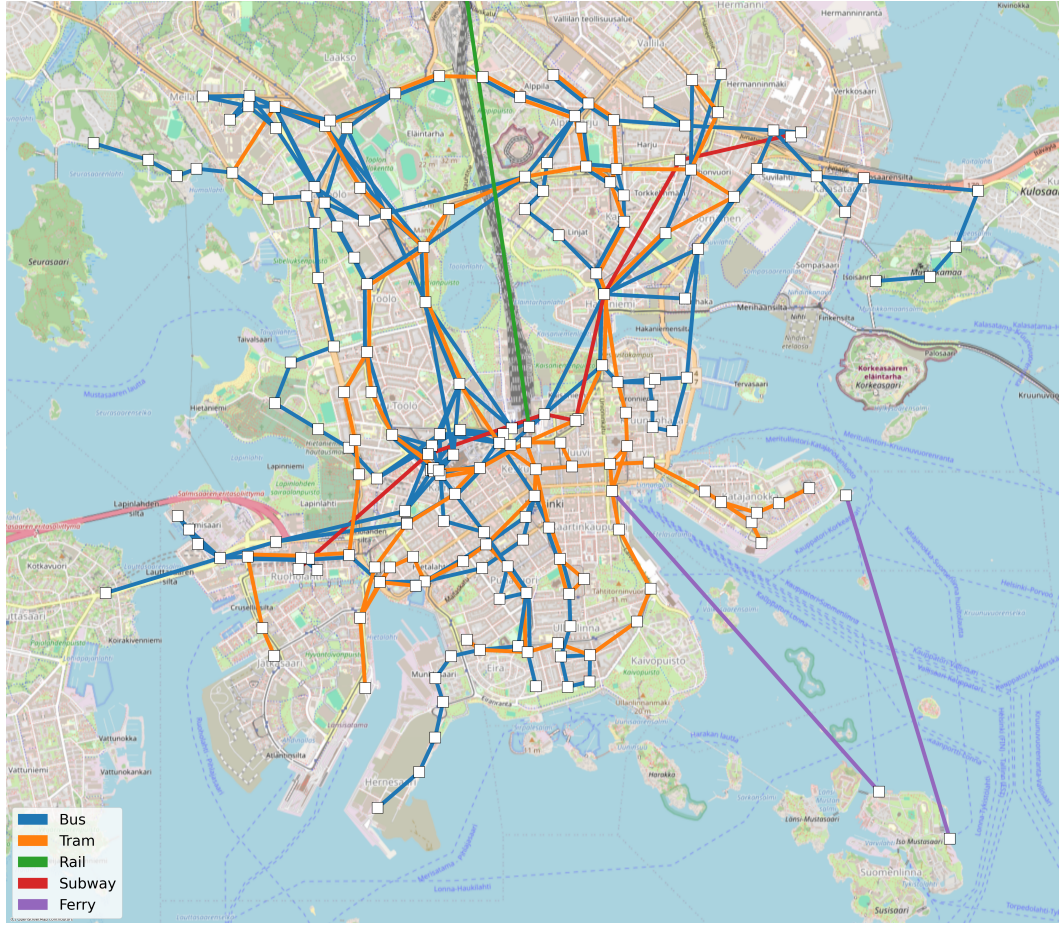


Figure 6 Visualization of the line-based modalities in the Helsinki dataset.

B OD Splitting

Algorithm 1 Splitting the OD matrix on infrastructure level into modality-specific OD matrices based on an MPTN routing.

Input: $\mathcal{OD}$, MPTN

Output: $\mathcal{OD}^m$ for all $m \in \mathcal{M}$

- 1: Initialize an empty OD matrix $\mathcal{OD}^m \leftarrow 0 \in \mathbb{N}^{|V_m| \times |V_m|}$ for all $m \in \mathcal{M}$
- 2: **for** all non-zero entries (ν_1, ν_2) in $\mathcal{OD}$ **do**
- 3: $p \leftarrow$ shortest path from ν_1 to ν_2 in MPTN with respect to t_e
- 4: split p in subpaths $p_1, \dots, p_k$ by removing transfer edges E^{trans}
- 5: **for** $i = 1, \dots, k$ **do**
- 6: Let m be the modality of subpath p_i starting at ν^s and ending at ν^t
- 7: $\mathcal{OD}_{\nu^s, \nu^t}^m \leftarrow \mathcal{OD}_{\nu^s, \nu^t}^m + \mathcal{OD}_{\nu_1, \nu_2}$
- 8: **return** $\mathcal{OD}^m$ for all $m \in \mathcal{M}$

C Numerical results

■ **Table 2** Computational results - solution characteristics.

Costs	bus	7,054.4329			12,722.4196	
	tram	8,528.4775			13,974.6190	
	rail	280.0000			280.0000	
	subway	20,048.8140			11,400.3060	
	ferry	1,059.3400			530.4200	
	total	36,971.0644			38,907.7646	
Average Travel Time	bus	2.3807	2.6389	2.3434	2.1065	2.2171
	tram	2.5375	2.2873	2.4460	2.6589	2.3862
	rail	0.0051	0.0070	0.0105	0.0022	0.0102
	subway	0.6627	0.7246	0.6431	0.7184	0.6142
	ferry	0.0116	0.0116	0.0116	0.0116	0.0116
	walk	0.8883	0.8708	0.9188	0.8484	0.9047
	intermodal transfers	0.4598	0.3932	0.4975	0.2564	0.2610
	overall	6.9458	6.9334	6.8709	6.6024	6.4051
Number of journeys	bus	28,534	27,745	28,941	21,808	25,851
	tram	23,297	21,681	23,581	22,488	24,892
	rail	105	104	164	29	138
	subway	15,634	15,688	15,827	15,694	15,246
	ferry	48	48	48	48	48
	walk	47,527	45,021	48,666	40,465	46,536
	total passengers	49,544	49,544	49,544	49,544	49,544

■ **Table 3** Computational results - runtime.

Runtime (gap)	Line planning	bus	6 h (11.45%)			6 h (0.02%)	
		tram	6 h (10.93%)				
		rail	0.01 s				
		subway	0.02 s				
		ferry	0.01 s				
	Timetabling	bus	6 h (2.44%)	9.64 s	6 h (1.13%)	0.01 s	1672.77 s
		tram	549.96 s	0.55 s		0.02 s	
		rail	0.01 s	0.01 s		0.01 s	
		subway	0.01 s	0.01 s		0.01 s	
		ferry	0.01 s	0.01 s		0.01 s	

Exploiting Treewidth to Solve the Pricing Problem in Column Generation for Non-Pool-Based Line Planning

Markus Niebisch ✉ 

Department of Data Analytics and Digitalisation, School of Business and Economics, Maastricht University, The Netherlands

Tom C. van der Zanden ✉ 

Department of Data Analytics and Digitalisation, School of Business and Economics, Maastricht University, The Netherlands

Abstract

The line planning problem (LPP) involves determining a set of lines (routes for vehicles to travel on) for a transportation network that meet travel demands while minimizing both travel time and the line operation costs. Unfortunately, this problem is NP-hard. Recently, Heinrich et al. [17] proposed using treewidth to solve (a variant of) this problem; however, the sizes of networks and values of treewidth for which it is practical are limited.

We propose a different way of exploiting bounded treewidth in solving the LPP. We combine a column generation approach due to Borndörfer et al. [5] with dynamic programming on tree decompositions to solve the pricing problem, which entails solving the NP-hard longest path problem.

We show that this approach is feasible. We are able to apply our approach to graphs with both more vertices and higher treewidth (up to 9) than Heinrich et al. We experimentally evaluate our approach using artificial benchmarks (connected concentric ring graphs) and two practical examples, the Munich train/subway network and the Japanese railroad system.

2012 ACM Subject Classification Mathematics of computing → Network flows; Mathematics of computing → Paths and connectivity problems; Applied computing → Transportation; Theory of computation → Fixed parameter tractability; Theory of computation → Integer programming

Keywords and phrases line planning, public transport, treewidth, integer programming, fixed parameter tractability, column generation

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.13

Supplementary Material *Software*: <https://doi.org/10.5281/zenodo.21797580>

1 Introduction

Public transport optimization is an important research topic in light of sustainable transport, rural exodus, and CO₂ emission goals [6, 12, 19]. One relevant aspect of public transportation optimization is the design of transportation routes and schedules. In this paper we study one such problem, which is called the line planning problem (LPP).

In the LPP it is asked to find a set of lines, i.e., routes for public transportation vehicles connecting origin-destination pairs, together with the frequency of the vehicles on those lines such that passenger transportation demands are met and the total cost is minimized, i.e., a weighted sum of passenger transportation time, fixed cost per line, and operating costs for running vehicles on lines. Unfortunately, the LPP is NP-hard [5]. One approach to solving the LPP is using column generation. We investigate the use of treewidth-based dynamic programming for solving the pricing problem, improving upon the approach using Color Coding due to Borndörfer et al. [5].



© Markus Niebisch and Tom C. van der Zanden;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 13; pp. 13:1–13:20

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The LPP is well-studied in the literature. Claessens et al. [7] solve a subset of the Dutch network with 28 stations in total to optimality. Fu et al. [13] solve the line planning problem for the full 783-station Chinese high-speed railway network of 2020, using a heuristic approach, while first determining the passenger paths and then the specific lines used, while at the same time first planning connections between major stations and after that connections with minor stations. Schöbel and Scholl [25] propose a method that can account for line transfers and evaluate it on the German rail network. For an extensive survey, see Schöbel [24].

One approach to tackling NP-hard problems is through parameterized complexity. A problem is *fixed-parameter tractable* (FPT) if it can be solved in time $f(k) n^{O(1)}$, where n is the size of the input and k is the *parameter*, see e.g., Cygan et al. [8]. Here, $f(k)$ can be an arbitrary computable function. Instances for which this parameter is small can then be solved efficiently. Both Borndörfer et al. [5] and Heinrich et al. [17] propose using FPT techniques to solve LPP.

Borndörfer et al. [5] formulate the LPP as a mixed integer linear program (MILP). Since the MILP contains variables corresponding to every possible line and every possible passenger path (of which there are exponentially many), it is infeasible to construct the full MILP. They therefore solve the linear relaxation of the MILP using column generation. They provide a greedy algorithm to round the solution of the LP relaxation into a solution of the original problem. They use column generation to generate variables corresponding to potential lines, i.e., origin-destination pairs and paths between them, and variables corresponding to passenger paths. The pricing problem for the lines is the weighted longest path problem. It is well known that this problem is strongly NP-hard [26].

To solve the pricing problem for the lines, Borndörfer et al. [5] employ a randomized dynamic programming algorithm, *color coding*, to find the longest weighted path. Color coding uses as a parameter k the maximum length of the path. They argue that this parameter is usually bounded in real-world transportation networks, which might be planar (in which case they argue “typical” lines have length $O(\sqrt{|V|})$ for a network with $|V|$ nodes), or might even be trees or tree-like, in which case one would get a bound of $O(\log |V|)$ on the maximum length of a path if the graph is preprocessed to remove nodes of degree 2. Therefore, they argue that color coding, an FPT algorithm parameterized by path length, should be a suitable method for solving the pricing problem.

The tree-likeness of networks can be exploited more directly by using the notion of treewidth. An obstacle to using treewidth in practice was the difficulty of finding tree decompositions. PACE 2016 and 2017 [9, 10] called for the development of practical treewidth solvers. Tamaki’s work (see e.g., [27, 28]) has been particularly influential in this area and has resulted in the availability of excellent solvers.

The first use of treewidth in solving the LPP is due to Heinrich et al. [17]. They observe that the treewidth of graphs underlying transportation networks is often small. Instead of using column generation, their algorithm, TW-LINES, constructs an integer linear program (ILP) that exploits the bounded treewidth of the network, allowing it to be solved in FPT time. However, the dependence on the parameter treewidth is bad, and they can only solve instances with a relatively small treewidth of at most 5. Moreover, they consider a more restricted version of the problem named “Line planning on all paths” (LPAL), in which the passenger paths are fixed in advance, and the only question is the routing and the integer frequency of the lines. Another simplification is their exclusion of a frequency-independent cost. This simplification is necessary since, otherwise, the problem is NP-hard even on paths and stars [16], thus excluding (unless $P=NP$) also the possibility of an FPT algorithm parameterized by treewidth. However, they can solve the ILP to optimality, whereas column generation only solves the LP relaxation to optimality and may yield a suboptimal solution after rounding.

We propose an alternative way to exploit treewidth to solve the LPP. We combine the column generation approach of Borndörfer et al. [5] with dynamic programming on tree decompositions to find the longest weighted path. Unlike the color coding approach used by Borndörfer et al. [5] to solve the (line) pricing problem, our approach can find lines of arbitrary length, which usually allows us to use fewer lines. This also allows us to generate columns for larger graphs without applying aggressive preprocessing. Borndörfer et al. [5] preprocess the network by iteratively removing leaves as well as degree-2 vertices, which is necessary to keep the length of the paths required reasonable. However, this simplification can make the model unrealistic. In the case of the Munich S-Bahn and subway network, this preprocessing would collapse the entire graph into only a few core vertices, making it difficult to apply the solution to the original graph.

To convert a solution to the LP relaxation to a solution for the original MILP, it is necessary to round the solution. Since the LP relaxation may contain many low-frequency lines, naively rounding it can result in a bad solution to the original MILP. We consider two complementary approaches: For small enough networks, it may be feasible to solve the original MILP restricted to the line set obtained by column generation for the LP relaxation. For larger networks, it may be necessary to use heuristics, as Borndörfer et al. [5] do.

We compare two different heuristic methods for rounding a solution of the LP relaxation to an MILP solution. Borndörfer et al. [5] used a method which we refer to as “Frequency-Based Greedy Rounding” (FB). We suggest an alternative method, “Generation-Order Greedy Rounding” (GO), which typically yields better results while reducing runtime by about 90%.

We similarly compare two methods to round the LP relaxation by solving an MILP. Following similar logic, the columns which are used in the final solution are usually generated early on during the column generation. Therefore, we compare an “early-stopping” variant of MILP rounding (that uses the same restricted line set as GO rounding) to MILP rounding using the full line set from the LP relaxation.

We experimentally evaluate our algorithm by solving the line-planning problem for concentric ring graphs, the Munich S-Bahn and subway network, and the Japanese national network, using both our method and color coding. We compare runtime, costs, and rounding methods. Furthermore, we adapt our method to the restricted problem of Heinrich et al. [17]. As Heinrich’s method solves LPAL to optimality, our result can only be worse. The main purpose of the comparison is to get insights into the quality of our algorithm with instances that TW-LINES can solve. This allows us to gauge the quality of our algorithm on larger instances that TW-LINES cannot solve.

Heinrich et al.’s [16, 17, 1] ILP-based approach, TW-LINES, is able to (within a time limit of 1 hour) solve instances with up to 90 vertices and treewidth 4, and graphs with up to 40 vertices and treewidth 5 (Intel Xeon Gold 6126 CPU). In some of their experiments, they stop the LP solver when it gets to within 2% of the optimal solution. In this setting, they are able to solve instances with up to 770 vertices and treewidth 3, up to 170 vertices and treewidth 4, and up to 41 vertices on graphs of treewidth 5. Compared to TW-LINES, our approach (with a more restrictive time limit of 30 minutes) can handle instances with up to ~ 700 vertices and treewidth 7, and instances with up to 82 vertices and treewidth 9 (Intel Core i7-11800H). Of course, our results are for solving an LP relaxation (albeit of a more complex problem) and cannot be directly compared to Heinrich et al.’s results, but they extend the set of networks for which *some* advantage of low treewidth can be taken.

This paper is organized as follows: we first discuss the Line Planning problem and Borndörfer et al.’s [5] column generation approach for it. We then discuss both the color coding and treewidth-based approaches for solving the longest path pricing problem. We

then discuss various approaches to rounding the LP relaxation to an integer solution. In our experiments section, we first compare against Heinrich et al.'s TW-LINES [17] (as a benchmark for another treewidth-based method) and against Borndörfer et al.'s [5] original method using color coding. Both comparisons are on artificial networks. We then investigate the performance of the different rounding methods. Finally, we present an evaluation of our method on two real-life networks: the train network of Japan and the public transport network of Munich.

2 Preliminaries

2.1 The Line Planning Problem

In this section, we introduce the variant of the line planning problem (LPP) that we consider, which is largely equivalent to that of Borndörfer et al. [5]. For simplicity, we do not consider multiple modes of transportation, though this can easily be added back into the model.

A public transportation network (PTN) is a graph $G = (V, E)$, where vertices represent stations and edges represent possible direct connections. Let $n := |V|$ denote the number of stations. In addition, we are given a demand matrix $D = (d_{s,t}) \in \mathbb{R}^{n \times n}$, where $d_{s,t}$ gives the number of travelers wanting to travel from station s to station t . Our objective is to find a set of lines (specifying routes and frequencies with which vehicles run on them) together with routes for passengers to travel (using the transportation capacity of the lines) so that the travel demand is met, while minimizing a weighted combination of operational cost and passenger travel time.

Formally, a line ℓ is a simple path $\ell = (v_1, \dots, v_k)$, where $v_i \in V$. The lines are undirected because we assume each transportation line will have vehicles running in both directions at the same frequency. We denote by κ_ℓ the maximum number of passengers that can be transported by a vehicle of a line in a single run. Each edge has a maximum frequency of vehicles that can run on it, which we denote by $f_e^{\max}$. We denote by F the maximum of the maximum edge frequencies of the entire network, i.e., $F = \max_{e \in E} f_e^{\max}$. We denote by $\mathcal{L}$ the set of all lines (i.e., the set of all simple paths in G).

In contrast, the passenger flows may be asymmetric. We define the passenger travel graph as $G_P = (V, A)$, with two antiparallel arcs $a_{s,t}$ and $a_{t,s}$ for every edge $e_{s,t}$ in E . For simplicity, we denote the two arcs of an edge e by $a(e)$ and $\bar{a}(e)$. The associated edge e (in the PTN) for arc a is represented by $e(a)$.

A travel path from v_s to v_t is a simple path $(v_s, \dots, v_t)$ using arcs in A . Each arc has a travel time τ_a , which leads to the travel time of a path p being $\tau_p = \sum_{a \in p} \tau_a$. The set of all travel paths connecting s and t is denoted as $\mathcal{P}_{s,t}$, and the set of all travel paths as $\mathcal{P}$.

The fixed operating cost of a line is c_ℓ^{fix} , and the variable operating cost of a line is the sum of the variable costs of its edges $c_\ell^{\text{var}} = \sum_{e \in \ell} c_e^{\text{var}}$.

Finally, to balance the total line operating cost with the travel cost, we multiply c_ℓ^{fix} and c_ℓ^{var} by $\lambda \in [0, 1]$. Similarly, we use $(1 - \lambda) \times \tau_a$ to adjust the travel cost. For notational convenience, we omit the scaling factor from our expressions and assume it has already been incorporated into the respective variables.

In the mixed-integer linear programming (MILP) formulation, we define a continuous variable y_p for each travel path $p \in \mathcal{P}$, giving the number of travelers using that path.

We define a binary variable x_ℓ , giving for each line ℓ whether that line is used in the solution, and a continuous variable f_ℓ , giving the frequency of vehicle runs on that line.

The total number of passengers using each arc for travel must be able to be transported by the lines that run along that arc. To this end, we define the line frequency of a single edge $f_e = \sum_{\ell \in \mathcal{L}(e)} f_\ell$, where the set of lines which share a single edge $e \in E$ is denoted as $\mathcal{L}(e) = \{\ell \in \mathcal{L} : e \in \ell\}$. Analogously, the passenger flow of a single arc is $y_a = \sum_{p \in \mathcal{P}(a)} y_p$, where $\mathcal{P}(a) = \{p \in \mathcal{P} : a \in p\}$ denotes the paths that use arc a .

Finally, to later be able to guarantee feasibility during the initial phases of column generation, we introduce continuous artificial variables $u_{s,t}$, denoting the unfulfilled travel demand between s, t , and introduce a big- M penalty M for unfulfilled demand. This then leads to the following MILP formulation:

$$\begin{aligned}
 (\text{MILP}) \quad & \min \sum_{p \in \mathcal{P}} \tau_p y_p + \sum_{\ell \in \mathcal{L}} c_\ell^{\text{fix}} x_\ell + \sum_{\ell \in \mathcal{L}} c_\ell^{\text{var}} f_\ell + \sum_{s,t \in D} M u_{s,t} \\
 \text{s.t.} \quad & \sum_{p \in \mathcal{P}_{s,t}} y_p + u_{s,t} = d_{s,t} & \forall (s,t) \in D & \quad (i) \\
 & y_a - \sum_{\ell \in \mathcal{L}(e(a))} \kappa_\ell f_\ell \leq 0 & \forall a \in A & \quad (ii) \\
 & f_e \leq f_e^{\max} & \forall e \in E & \quad (iii) \\
 & f_\ell \leq F x_\ell & \forall \ell \in \mathcal{L} & \quad (iv) \\
 & x_\ell \in \{0, 1\} & \forall \ell \in \mathcal{L} & \quad (v) \\
 & f_\ell \geq 0 & \forall \ell \in \mathcal{L} & \quad (vi) \\
 & y_p \geq 0 & \forall p \in \mathcal{P} & \quad (vii) \\
 & u_{s,t} \geq 0 & \forall (s,t) \in D & \quad (viii)
 \end{aligned}$$

The objective function contains four elements:

1. $\sum_{p \in \mathcal{P}} \tau_p y_p$: the total travel time across all passenger paths,
2. $\sum_{\ell \in \mathcal{L}} c_\ell^{\text{fix}} x_\ell$: the total fixed costs of all active lines,
3. $\sum_{\ell \in \mathcal{L}} c_\ell^{\text{var}} f_\ell$: the variable costs times the line frequencies,
4. $\sum_{s,t \in D} M u_{s,t}$: finally, a big- M penalty for unfulfilled travel demands.

The constraints are:

- (i) assigned passenger flow plus unfulfilled demand equals travel demand,
- (ii) the available capacity on each arc is at least the total travel demand y_a on that arc,
- (iii) the frequency of vehicles running on an edge does not exceed its maximum frequency,
- (iv,v) decide whether a line is part of the line plan or not,
- (vi, vii, viii) non-negativity of passenger flow, line frequencies, and unfulfilled travel demands.

2.2 Column generation for Line Planning

We use column generation to solve the LP relaxation of the MILP formulation introduced in the previous section. Borndörfer et al. [5] observe that the LP relaxation can be simplified because, in the LP relaxation, constraint (iv) will always be fulfilled with equality. We can therefore remove the x_ℓ variables and incorporate the fixed costs of the lines into the variable costs by setting $c_\ell^{\text{total}} = c_\ell^{\text{fix}}/F + c_\ell^{\text{var}}$. This leads to the following LP relaxation:

$$\begin{aligned}
 (\text{LP}) \quad & \min \sum_{p \in \mathcal{P}} \tau_p y_p + \sum_{\ell \in \mathcal{L}} c_\ell^{\text{total}} f_\ell + \sum_{s,t \in D} M u_{s,t} \\
 \text{s.t.} \quad & \sum_{p \in \mathcal{P}_{s,t}} y_p + u_{s,t} = d_{s,t} & \forall (s,t) \in D & (i) \\
 & y_a - \sum_{\ell: e(a) \in \ell} \kappa_\ell f_\ell \leq 0 & \forall a \in A & (ii) \\
 & f_e \leq f_e^{\max} & \forall e \in E & (iii) \\
 & f_\ell \geq 0 & \forall \ell \in \mathcal{L} & (vi) \\
 & y_p \geq 0 & \forall p \in \mathcal{P} & (vii) \\
 & u_{s,t} \geq 0 & \forall (s,t) \in D & (viii)
 \end{aligned}$$

To solve this LP, we use column generation to generate columns corresponding to the lines $\mathcal{L}$ and passenger paths $\mathcal{P}$. For an introduction to column generation, we refer to [11].

The dual of the above LP is:

$$\begin{aligned}
 (\text{dual}) \quad & \max \sum_{s,t \in D} d_{s,t} \pi_{s,t} - \sum_{e \in E} f_e^{\max} \eta_e \\
 \text{s.t.} \quad & \pi_{s,t} - \mu(p) \leq \tau_p & \forall p \in \mathcal{P}_{s,t}, (s,t) \in D & (a) \\
 & \kappa_\ell \mu(\ell) - \eta(\ell) \leq c_\ell^{\text{total}} & \forall \ell \in \mathcal{L} & (b) \\
 & \pi_{s,t} \leq M & \forall (s,t) \in D & (c) \\
 & \mu_a \geq 0 & \forall a \in A & (d) \\
 & \eta_e \geq 0 & \forall e \in E & (e)
 \end{aligned}$$

where

$$\mu(\ell) = \sum_{e \in \ell} (\mu_{a(e)} + \mu_{\bar{a}(e)})$$

For the derivation of the dual formulation, we refer to [5]. The only difference between our LP and that of [5] is that we use a big- M formulation to deal with unfulfilled demand, which leads to an additional constraint $\pi_{s,t} \leq M$ in the dual.

Again, following [5], the reduced cost of the variable relating to a passenger path p is

$$\bar{\tau}_p = \tau_p - \pi_{s,t} + \mu(p) = \tau_p - \pi_{s,t} + \sum_{a \in p} \mu_a = -\pi_{s,t} + \sum_{a \in p} (\mu_a + \tau_a)$$

Thus, to solve the pricing problem, we need to solve a weighted shortest path problem in the passenger travel graph, with arc weights $\mu_a + \tau_a$. This can easily be done using, e.g., Dijkstra's algorithm, since all weights are non-negative.

For the lines, the reduced cost is

$$\begin{aligned}
 \bar{\tau}_\ell &= c_\ell^{\text{total}} - \kappa_\ell \mu(\ell) + \eta(\ell) \\
 &= c_\ell^{\text{fix}}/F + c_\ell^{\text{var}} - \kappa_\ell \mu(\ell) + \eta(\ell) \\
 &= c_\ell^{\text{fix}}/F + \sum_{e \in \ell} (c_e^{\text{var}} - \kappa_\ell (\mu_{a(e)} + \mu_{\bar{a}(e)}) + \eta(e))
 \end{aligned}$$

Thus, to find a line with a negative reduced cost, we need to find a line ℓ with:

$$\sum_{e \in \ell} (\kappa_\ell(\mu_{a(e)} + \mu_{\bar{a}(e)}) - \eta(e) - c_e^{\text{var}}) > c_\ell^{\text{fix}}/F$$

As $\kappa_\ell(\mu_{a(e)} + \mu_{\bar{a}(e)}) - \eta(e) - c_e^{\text{var}}$ is not restricted in sign, solving the pricing problem for the lines is equivalent to solving a maximum weighted path problem, with edge weights $\kappa_\ell(\mu_{a(e)} + \mu_{\bar{a}(e)}) - \eta(e) - c_e^{\text{var}}$.

2.3 Color Coding

Borndörfer et al. [5] solve the longest weighted path problem with a randomized algorithm, color coding. In every iteration, a random color is assigned to every vertex, and, using dynamic programming, the longest weighted path without a color repeating is found for a given start vertex. By repeating this process with different random colorings, we can boost the probability of finding the longest path. A single iteration has a complexity of $m \times 2^{O(k)}$, where m is the number of edges and k is the number of colors. This algorithm would need $a \times e^k$ iterations to achieve a success rate of $1 - e^{-a}$. For example, this would mean that for $a = 2$ and $k = 10$, we would require $\sim 44,000$ iterations to have a success rate of $\sim 86\%$. Finally, this process must then be repeated for every vertex as a starting point.

This algorithm has several downsides; for one, we cannot find a path longer than the number of colors k . Furthermore, if a larger number of colors is selected, the runtime increases exponentially, as we need an exponentially larger number of iterations to achieve the same success rate, and the runtime of a single iteration also grows exponentially. Borndörfer et al. [5] use only 15 iterations in their implementation, suppress stations with degrees 1 or 2, and use a subset of V as terminal vertices for lines to make this approach feasible. In our experiments, we also increase the number of colors rather than the number of iterations, as suggested by [18].

In comparison, our treewidth-based dynamic programming can find paths of arbitrary length, provided the graph has bounded treewidth. Furthermore, we would generally not need to remove a node of degree 1 or 2, since subdividing edges does not increase treewidth; therefore, we can consider the original network.

3 Our approach

The main algorithm first constructs the initial linear-programming (LP) relaxation, as described in Section 2.2. Afterward, it conducts column generation, i.e., alternates between solving the pricing problem for the transport paths and for the transport lines, and modifies the LP using the edge weights as described in the same section. To solve the former, we use Dijkstra's algorithm; for the latter, we use dynamic programming on tree decompositions, as described in Section 3.1. This process of adding paths and lines is repeated until no improvement can be found after adding a path. After that, the solution is rounded to an MILP solution, and low-frequency lines are removed, as described in Section 3.2.

3.1 Treewidth-based dynamic programming for Longest Path

In this section, we briefly describe the algorithm for solving the weighted longest path problem by dynamic programming on tree decompositions and recall the basic definitions. For a thorough introduction to treewidth and dynamic programming, we refer to [8].

A *tree decomposition* of a graph G is a tuple $(T, \mathcal{B})$ where T is a tree and $\mathcal{B} = \{b_i \mid i \in V(T)\}$ is a set of bags. Each bag $b_i \in \mathcal{B}$ is a subset of V . Additionally, the following properties must be satisfied:

1. $\bigcup_{b \in \mathcal{B}} b = V$, i.e., every vertex is contained in some bag,
2. For all $(u, v) \in E$, there exists $b \in \mathcal{B}$ such that $\{u, v\} \subseteq b$, i.e., for each edge in E , there is a bag which contains both its endpoints,
3. For all vertices $u \in V$, the bags $b \in \mathcal{B}$ containing vertex u must form a connected subgraph.

The *width* of a tree decomposition is $\max_{b \in \mathcal{B}} |b| - 1$, and the *treewidth* of a graph is the minimum width of a tree decomposition of it.

An important property of tree decompositions is that the bags are separators of G (see e.g., [8, Lemma 7.3]). This property is what enables dynamic programming. If we view the tree decomposition as a rooted tree, then every bag $b_t \in \mathcal{B}$ separates the vertices in the subgraph of G induced by the vertices contained in (the bags of) the subtree of T rooted at t from the rest of G . We denote this subgraph by $G[t]$. For clarity, $G[t]$ can be viewed as b_t plus a union of connected components of $G \setminus b_t$; which components exactly depends on the choice of root.

We define a *partial solution* for $t \in V(T)$ as a union of vertex-disjoint (including endpoints) paths of $G[t]$, such that at most two endpoints of the union of paths are not contained in b_t and every path has at least one endpoint in $G[t]$, unless the partial solution consists of a single path.

Note that if p is a path in G , then the intersection of p with $G[t]$ is a partial solution for t . Thus, we can solve the longest path problem by enumerating all partial solutions. Dynamic programming on tree decompositions speeds up this enumeration by taking advantage of the fact that some partial solutions are in some sense “equivalent”, and we only need to keep a partial solution of maximum weight per equivalence class.

Two partial solutions for $t \in V(T)$ are equivalent if (1) all vertices in b_t have the same degree in the partial solution (which can be either 1 or 2, or, if the vertex is not contained in the partial solution, 0), (2) both partial solutions connect the same degree-1 vertices of b_t and (3) the number of path endpoints not contained in b_t is equal. For a graph of treewidth tw , the *characteristic* of a partial solution can thus be defined as a $(2|b_t| + 1)$ -tuple, with the first $|b_t|$ elements giving the degree of each vertex of b_t in relation to the partial solution, the second $|b_t|$ giving which vertex each degree-1 vertex is connected with (if any) and the last value giving the number of endpoints not contained in b_t .

Our algorithm is standard dynamic programming on tree decompositions: we process the tree in a bottom-up fashion, and for each node $t \in V(T)$ compute – using the values previously computed for the node’s children – a list of partial solutions, grouped by characteristic, storing only the maximum weight partial solution for each. To simplify our algorithm, we preprocess the tree decomposition into a *nice tree decomposition* [4]. We also implemented the rank-based approach [3] to further reduce the number of equivalence classes by recognizing that certain connectivity partitions of degree-1 vertices are equivalent.

This is very similar to the algorithm for Hamiltonian Cycle evaluated in [29] (to which we refer for more details), though modified slightly since we do not need to visit all vertices or connect the endpoints of a path into a cycle.

3.2 Rounding

As a final step, the LP solution needs to be rounded to a solution for the MILP. Since the binary decision variables x_ℓ for all $\ell \in \mathcal{L}$ were removed during relaxation, they must now be set to obtain a feasible solution to the MILP. Hereby, the solution frequencies for the lines f_ℓ and paths y_p of the LP are substituted into the objective function for the MILP. If $f_\ell > 0$ then x_ℓ is set to 1, otherwise x_ℓ is set to 0. As the cost per line changes from $\sum_{\ell \in \mathcal{L}} c_\ell^{\text{total}} f_\ell$ to $\sum_{\ell \in \mathcal{L}} c_\ell^{\text{fix}} x_\ell + \sum_{\ell \in \mathcal{L}} c_\ell^{\text{var}} f_\ell$, it is likely that the solution can be improved by omitting some low-frequency lines.

We use two approaches for heuristic rounding. First, we follow the greedy process of Borndörfer et al. [5], which starts with the full line pool and removes lines one by one to obtain a new solution. This is achieved by setting the capacity coefficient κ_ℓ of the line with the lowest non-zero frequency to 0, and solving the pricing problem for the passenger paths again. This is repeated as long as the MILP cost is not increasing. The last result is then the final solution. We will refer to this rounding method as “Frequency-Based Greedy Rounding” (FB).

The number of lines generated when solving the LP relaxation is usually far larger than the number of lines that end up in the final solution (generated by FB rounding). However, a lot of time is consumed by re-solving the LP each time a line is removed during FB rounding. We observed that, in practice, the lines which end up being used in the solutions are usually generated fairly early on during the column generation process.

This observation leads to an alternative approach. Here, we start with an empty line pool, i.e., set the capacity κ_ℓ of all lines ℓ to zero, and add them one by one in the order they were created during the column generation, i.e., set κ_ℓ to the original value, as long as the respective MILP cost is decreasing. We will refer to this rounding method as “Generation-Order Greedy Rounding” (GO). This method has the advantage that no additional runtime is needed when executed in parallel to the regular column generation.

Finally, we compare the results of the two heuristic rounding methods with those obtained by solving the MILP restricted to the line pool $\mathcal{L}$ produced by column generation for the LP relaxation. We distinguish between two different line pools: “Converged MILP” (CO MILP) uses all lines of the optimal solution to the LP relaxation. The second method, “Early Stop MILP” (ES MILP) uses a line pool containing only those lines that would have been used by “Generation-Order Greedy Rounding” (GO). For the biggest instances, running the column generation to completion is not possible within a reasonable amount of time. Similarly to the situation with GO vs. FB rounding, lines generated late are usually not used in the solution. Therefore, ES MILP rounding allows us to save significant computation time without a large decrease in solution quality.

4 Experiments

We run experiments on two types of graphs, concentric ring graphs and real-world examples, and investigate three different topics in our experiments. First, we compare the runtime and costs of our approach, which we refer to as TREEWIDTH COLUMN GENERATION (TWCG), against TW-LINES by Heinrich et al. [17, 1]. Second, we compare the runtime and solution quality of TWCG against COLOR CODING (CC) (as used by Borndörfer et al. [5]) for the pricing problem in line planning. Finally, we compare the quality of the four different rounding methods.

4.1 Experiment parameters

If not stated otherwise, we use the following parameter values: fixed costs per line $c_\ell^{\text{fix}} = 10,000$, operational costs for an edge $c_e^{\text{var}} = 100$, transport time $\tau_a = 1$, penalty value for unfulfilled transport demand $M = 10^6$, travel demand between $d_{s,t} = 50$ for all vertex pairs (s, t) . We use a time limit of 30 minutes wall time, after which rounding is started and we allow the MILP solver to run for an additional 15 minutes. The time limit was only reached for instances with treewidth 9 or instances with very many vertices. Since the final line used is typically found within the first few minutes, the time limit primarily affects how many ultimately disused columns were generated, without affecting the final solution quality.

The transport capacity κ_ℓ and the maximum edge frequency F are instance-dependent and are created similarly to [17] to help comparability: We set $\kappa_\ell = (|V| - 1)^2$ for all lines ℓ . To set F , we compute first the shortest paths between each pair of vertices s, t . We then count the number of passengers who would traverse each edge, assuming each passenger travels from their origin to their destination over the shortest path. The maximum edge frequency F is then the maximum resulting value divided by κ_ℓ , i.e., let $sp_{s,t}$ be the shortest path between v_s and v_t , and let $d_a = \sum_{\{(s,t): a \in sp_{s,t}\}} d_{s,t}$, then $F = \max_{a \in A, \ell: e(a) \in \ell} \frac{d_a}{\kappa_\ell}$. The line cost estimation in the LP relaxation will be closer to the MILP cost if the frequency of a line f_ℓ is close to the maximum frequency F , as in the LP relaxation, the line cost is estimated as $c_\ell^{\text{total}} = c_\ell^{\text{fix}}/F + c_\ell^{\text{var}}$. In practice, to solve the LPP using a relaxation-based approach, one would have to come up with reasonable estimates of the maximum frequency per line so that the integrality gap does not become too large. Our choice of F guarantees a demand-fulfilling solution while keeping the gap reasonably small.

To determine the value of λ for different instances, we let the experiment run twice: Once with $\lambda = 0.5$, after which we calculate the λ value that would equalize travel and operating costs of the LP solution and we run the experiment with that λ .

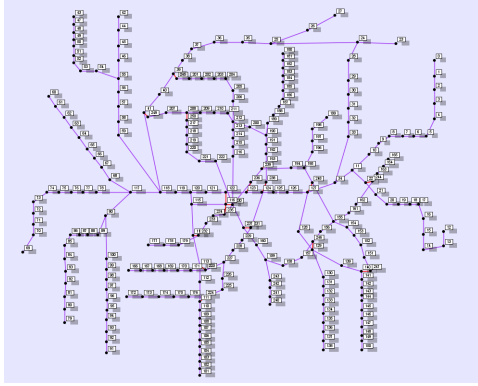
All experiments were conducted using an Intel Core i7-11800H processor at 2.30 GHz, 12 GiB RAM running Windows 11. Our implementation was written in Java and executed using OpenJDK Runtime Environment 22.0.1 [21]. We used OR-Tools version 9.8.3296 [22] together with Gurobi version 10.0.2 [15]. Although Gurobi 12.0.1 was available, OR-Tools did not yet support it when the experiments were conducted. For comparison, we used the authors' original C++ implementation of Heinrich's algorithm TW-LINES [1], which we ran with Gurobi 12.0.1. In order to speed up our experiments, we parallelized the COLOR CODING algorithm over the possible different starting points. Furthermore, we also parallelized the join operator of TWCG over the entries of one of the tables being joined. Every tree decomposition was created using Jdrasil 0.4dev [2], which is the same solver Heinrich et al. [17] use. We do not report the time of computing a tree decomposition separately, as even for the largest instances this only takes a few seconds. For this reason, we did not explore other solvers.

4.2 Instances

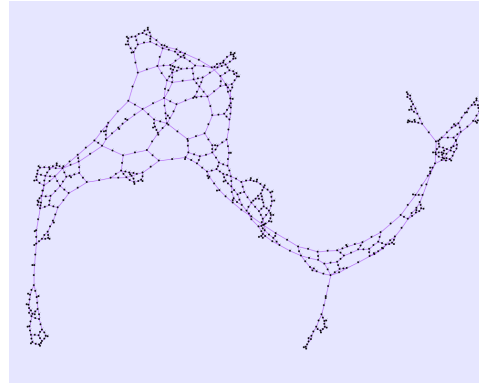
Next, we introduce the two types of graphs on which we conduct our experiments: concentric ring graphs and two real-world examples.

The first type of graph in our experiments is concentric ring graphs, as used in the experiments of Heinrich et al. [17]. The concentric ring graph consists of r concentric cycle graphs C_s , each with s vertices, and one extra "hub" vertex. All vertices of the first cycle graph are connected to the central hub and form a wheel graph with s spokes. Each vertex in each cycle graph is then connected to the corresponding vertex in the previous and next cycle graph. In total, we solve 32 instances with between 3 and 9 spokes and treewidth ranging from 3 to 9.

We also experiment on two graphs modeling real-world examples, namely the Munich S-Bahn and subway network and the Japanese national rail network.



■ **Figure 1** Munich S-Bahn and subway network. $|V| = 251$, $|E| = 249$, $tw = 4$. Transfer edges in red.



■ **Figure 2** Train network of Japan. $|V| = 694$, $|E| = 830$, $tw = 7$.

The Munich network consists of 251 vertices, 249 regular edges, 14 transfer edges, and has treewidth 4, as shown in Figure 1. We obtained the network by tracing publicly available maps [20]. As it utilizes two transportation modes, we employ a disconnected graph for line planning and additionally use 14 special transfer edges for path planning, an approach also used by [5]. These edges have infinite capacity and zero cost. We chose Munich as a medium-sized case to show that several transportation modes are possible. Furthermore, the Munich graph cannot be handled by the original simplification step proposed by Borndörfer et al. [5], which iteratively removes degree-2 vertices and leaves. In the case of the Munich S-Bahn without the subway, the resulting graph would no longer be a realistic model of the network, as the simplification would remove substantial portions of the network.

The final case is the Japanese rail network, which consists of 995 vertices and 1,131 edges [23]. We are using $d_{s,t} \in \{1, 2, 3, 4, 5\}$ chosen uniformly at random for all $(s, t) \in V \times V$. To reduce computation time, we contract the common edge between neighboring vertices of degree 2 and adjust the demand matrix accordingly (this is a slightly less aggressive preprocessing than the one used by Borndörfer et al. [5]). The contracted graph (see Figure 2) has 694 vertices and 830 edges, and treewidth 7.

For the Japanese network, the bottleneck is not solving the pricing problem but rather solving the LP relaxation. Using the full demand matrix results in a large number of constraints, since its size is quadratic in the number of vertices. Therefore, we reduce the size of the demand matrix and zero out all the columns except for 30. This could be interpreted as a rush hour where passengers commute into several key cities from all over the country, which may be a more realistic scenario than one in which we have equal demand between all pairs of stations. Additionally, we set $\kappa_\ell = 1,320$, the approximate transport capacity of a Shinkansen train, and $F = 18$. This happens to equal the actual maximum frequency of the Shinkansen, but the fact that this yields a feasible solution with our demand matrix (which does not include realistic demands) is a coincidence. To account for the larger instance size, we increase the running time before rounding starts to 60 minutes and use “Early Stop MILP”, as explained in Section 3.2, as the rounding method.

4.3 Comparison against TW-LINES

We compare TWCG against Heinrich et al.’s TW-LINES [17] treewidth-based algorithm on concentric ring graphs. The problem (LPAL) solved by TW-LINES is substantially different from the LPP problem: It does not concern itself with passenger paths (and instead, takes as input a travel demand per edge) and it does not incorporate a per-line fixed cost, which makes it a much simpler problem than LPP, but it does constrain line frequencies to be integer rather than continuous (which makes this aspect harder than LPP). This makes it difficult to compare TW-LINES and our TWCG, but since TW-LINES is the only other approach to a line planning problem using treewidth, we decided to include this benchmark.

Due to these differences, we have to adjust our LP formulation. Since passenger paths are no longer part of the problem, we add, for each OD-pair, the shortest path to the LP as a column and prohibit adding additional paths. Furthermore, we set $\lambda = 1$ (to take travel costs out of the equation). Instead of a per-line fixed cost, LPAL uses a frequency-dependent fixed cost. Since in the LP relaxation of LPP, the per-line fixed cost turns into a frequency-dependent cost, we can make these parameters equal.

After the LP relaxation is solved and no further line columns can be added, we solve an ILP version of the LP, i.e., all line variables are set to integer instead of continuous. Note that LPAL also features a maximum frequency per line, which would make solving this ILP infeasible, but our F is large enough to avoid this scenario.

The results of these experiments are shown in Table 1. As expected, their solution (as they solve to optimality instead of dealing with an ILP relaxation), is better in almost every case – with the exception of some cases (the two cases with treewidth 5 and 8 or 10 rings) where their ILP solver runs out of time before finding an optimal solution.

In the worst case, our solution was 23.6% more expensive for 5 spokes and 5 rings, while in the best case, we achieved 12.79% lower cost for 5 spokes and 8 rings. On average, our solutions were 8.3% worse, with a median of 8.0%.

When comparing runtimes, we observe that our algorithm runs faster on every ring graph instance except the smallest. The largest instances took TWCG 40 seconds, with many being below 10 seconds. For example, TWCG solves 3 spokes and 20 rings in 35.1 seconds, 4 spokes and 7 rings in 4.3 seconds, and 5 spokes and 5 rings in 2.6 seconds, while TW-LINES uses the full 30 minutes for each of these instances and finishes with IP gaps of 2%, 0.5%, and 0.6%, respectively.

The most complex instances where TW-LINES managed to provide a solution were for 5 spokes and 8 rings, and for 5 spokes and 10 rings. However, those had larger IP gaps of 12.8% and 16.6%, respectively. TW-LINES did not manage to solve the smallest treewidth 6 instance with 5 rings. In contrast, our approach can handle (within the 30-minute time limit) instances with treewidth up to 9.

Heinrich et al. [17] report average improvements of 36% over heuristic line pool generation in [14]. Considering that our solutions were only on average 8.3% worse than Heinrich et al.’s, TWCG appears to perform decently as a heuristic for LPAL (a problem it was not initially designed for).

4.4 Comparison against COLOR CODING on ring graphs

Next, we compare the solution quality of TWCG and COLOR CODING on concentric ring graphs. We solve instances with treewidth up to 9, up to 9 rings, and spokes ranging from 3 to 9. For COLOR CODING, we consider two parameter settings and report the one that yields better results for each instance: $k = 20$ with 50 iterations and $k = 25$ with 20 iterations.

■ **Table 1** Comparing TW-LINES by Heinrich et al. and TREewidth COLUMN GENERATION (TWCG) on ring graphs. Left: description of the instances where s , r , tw , and n denote the number of spokes, rings, treewidth, and number of stations. Middle: the TWCG solution – number of lines, total cost, average frequency, average line length, and runtime. Right: the same metrics for the TW-LINES solution, and the IP gap.

Input				TWCG					TW-LINES					
s	r	tw	n	no. l	cost	avg. freq	avg. len.	time (s)	no. l	cost	avg. freq	avg. len.	time (s)	IP gap
3	5	3	16	7	17,560	1	12.286	0.849	7	17,060	1	11.571	0.382	0
3	10	3	31	6	23,180	1	25.833	2.818	6	21,680	1	23.333	58.872	0
3	15	3	46	6	29,980	1	37.167	8.680	6	28,280	1	34.333	964.664	0
3	20	3	61	6	37,780	1	50.167	35.109	6	35,180	1	45.833	1,803.808	0.020
4	5	4	21	7	19,760	1	15.429	1.230	6	17,280	1	16	44.413	0
4	7	4	29	6	21,680	1	23.333	4.333	6	19,380	1	19.500	1,801.615	0.005
4	10	4	41	6	27,380	1	32.833	7.619	6	24,280	1	27.667	1,802.223	0.043
4	12	4	49	6	30,180	1	37.500	18.093	5	25,300	1	37.800	1,803.261	0.040
5	3	5	16	8	18,140	1	9.875	0.483	8	17,240	1	8.750	19.606	0
5	4	5	21	8	20,340	1	12.625	1.598	8	18,740	1	10.625	133.323	0
5	5	5	26	6	19,780	1	20.167	2.593	5	16,000	1	19.200	1,816.643	0.006
5	6	5	31	6	21,780	1	23.500	4.742	5	18,000	1	23.200	1,820.557	0.006
5	8	5	41	6	34,060	1	31.167	22.952	6	39,057	1	27	1,828.562	0.128
5	10	5	51	5	34,900	1	44.200	39.919	5	38,831	1	39.200	1,837.041	0.166
6	5	6	31	7	23,560	1	20.857	15.864	-	-	-	-	1,830	-

As shown in Table 2, TWCG outperforms the COLOR CODING pricing method on ring graphs in 17 of the 21 examples. This is primarily because longer lines are possible. For example, on an instance with 65 vertices and treewidth of 8, TWCG has an LP cost of 45,248 with 96 active lines and a MILP cost of 90,747 with 4 active lines, while COLOR CODING has an LP cost of 58,441 with 8 active lines and a MILP cost of 112,762 with 6 active lines. For the most complex instance with 9 rings and 9 spokes and treewidth 9, TWCG has an LP cost of 53,045 with 21 active lines and a MILP cost of 105,208 with 5 active lines, while COLOR CODING has an LP cost of 61,460 with 15 active lines and a MILP cost of 141,501 with 8 active lines.

As expected, the difference increases as the graph grows. Since the maximum line length for COLOR CODING is limited to 20 or 25, respectively, it incurs higher fixed costs because it must cover long paths with many short lines. Furthermore, in larger instances, relatively few lines are generated for COLOR CODING, as the runtime for generating them increases. COLOR CODING works better with more colors k on the smaller instances, but on the largest instances with 73 and 82 stations, the setting with $k = 20$ outperformed $k = 25$ because more lines could be generated.

4.5 Running times

Next, we investigate the runtime of TWCG and COLOR CODING on the same concentric ring graphs, with treewidth up to 9 and at most 82 stations. Table 3 shows the comparison of runtime between the different instances. In all instances, it took less than 1 second to price the transportation paths, which is negligible.

For treewidth up to 6, pricing the lines with TWCG is practically instantaneous: 11 seconds for treewidth 6 and 37 vertices. For instances with treewidth 7, the runtime still remained reasonable at below 2 minutes. For treewidth 8 and 57 stations, TWCG took 19 minutes to price all lines. For all instances with higher treewidth, it was no longer possible to generate all columns and solve the LP relaxation to optimality within our time limit.

■ **Table 2** Ring graph comparison between TWCG and COLOR CODING (CC). The left block describes the instances, where s , r , tw , n are the number of spokes, rings, treewidth, and number of vertices. The middle block compares the number of lines and cost between TWCG and COLOR CODING in the LP relaxation. The right block compares the MILP solution. Additionally, the number of colors k and the number of iterations $iter.$ are stated.

Input				LP						MILP						
s	r	tw	n	TWCG	no. l	cost	CC	no. l	cost	TWCG	no. l	cost	CC	no. l	cost	k $iter.$
3	1	3	4		2	1,160		2	1,160		2	1,351		2	1,351	25 20
3	2	3	7		8	4,418		5	4,418		3	6,219		2	6,195	25 20
3	3	3	10		8	9,612		7	9,612		2	15,450		2	15,702	25 20
4	2	4	9		6	7,906		6	7,906		3	12,166		2	12,206	20 50
4	3	4	13		11	13,646		14	13,646		3	22,625		2	22,302	25 20
4	4	4	17		23	19,487		26	19,344		2	31,389		2	31,821	20 50
5	3	5	16		25	16,390		20	16,425		4	29,733		2	28,323	25 20
5	4	5	21		33	22,395		34	22,355		2	37,964		2	38,493	25 20
5	5	5	26		37	27,003		35	27,061		2	46,290		2	49,874	25 20
6	4	6	25		40	23,874		41	24,117		2	44,152		3	48,651	25 20
6	5	6	31		49	27,778		43	28,403		3	53,758		3	59,476	25 20
6	6	6	37		66	34,174		40	34,625		4	65,071		4	70,300	25 20
7	5	7	36		54	30,619		38	31,100		3	55,827		4	66,072	25 20
7	6	7	43		72	34,354		30	35,671		3	63,850		4	78,186	25 20
7	7	7	50		84	39,559		19	43,685		4	75,337		4	88,973	25 20
8	6	8	49		90	36,820		19	38,058		3	75,883		5	89,969	25 20
8	7	8	57		97	42,094		12	46,469		4	85,276		5	101,027	25 20
8	8	8	65		96	45,248		8	58,441		4	90,747		6	112,762	25 20
9	7	9	64		62	44,084		8	53,577		5	90,570		6	111,923	25 20
9	8	9	73		20	46,995		19	51,361		3	92,354		7	124,643	20 50
9	9	9	82		21	53,045		15	61,460		5	105,208		8	141,501	20 50

COLOR CODING already reaches the 30-minute limit with treewidth 6 and 31 stations. All instances larger than that reached the threshold of 30 minutes, due to the time it takes to generate the lines. It would be possible to run the same experiments with fewer colors. We tried $k = 12$ colors as well, but then the solution quality is worse. It is better to have fewer high-quality lines in the pool than many small ones.

However, recall that COLOR CODING is not guaranteed to find the optimal solution for the longest paths, and the quality of the solution depends on the number of colors and number of iterations (relative to the instance size). However, since we keep these parameters fixed, this means that the quality of the solution found by COLOR CODING is decreasing – if we wanted to keep the quality of the solution comparable, we should increase both parameters of COLOR CODING with the size of the graph, but doing so is infeasible. As COLOR CODING generated fewer lines, the time for solving the MILP was less than 2 minutes, even in the worst instance with treewidth 6 and 31 stations.

A treewidth of 9 is the maximum treewidth the algorithm can solve within the 30-minute time limit. The join operator becomes the bottleneck, since it may have to combine every subsolution of the first solution set with many of the second solution set. Depending on the structure of the tree decomposition, the running time can change drastically (affected by, e.g., whether a join node is at the root of the tree decomposition). Unfortunately, the Jdrasil solver is nondeterministic and generates a new decomposition each run. For the treewidth-9 instances, we selected a decomposition for our experiments that gave a fast running time.

We observed that generating many lines is not necessarily advantageous for the MILP rounding due to time constraints. Furthermore, we often noticed that the final lines were generated early in the algorithm. If we stop early and use “Generation-Order Greedy Rounding” or “Early Stop MILP” as explained in Section 3.2, we can still solve the biggest instance within 6 minutes.

■ **Table 3** Comparison of the runtime on ring graphs of the different algorithmic components. The left block describes the instances, where s , r , tw , n are the number of spokes, rings, treewidth, and number of vertices. The middle block gives the results for TWCG, the right block those for CC. Shown are the number of iterations, the cumulative time for pricing the paths and lines, for solving the linear program (LP), and for solving the mixed integer linear program (MILP).

Input				TWCG						CC					
s	r	tw	n	iterations	t lines	t paths	t lp solve	t MILP solve	t total	iterations	t lines	t paths	t lp solve	t MILP solve	t total
3	1	3	4	14	0.037	0.009	0.048	0.005	0	16	0.119	0.003	0.046	0.006	0
3	2	3	7	26	0.022	0.007	0.093	0.024	0	30	0.203	0.004	0.093	0.033	0
3	3	3	10	40	0.071	0.012	0.139	0.162	0	38	0.763	0.005	0.222	0.177	1
4	2	4	9	20	0.016	0.003	0.106	0.032	0	16	0.219	0.002	0.089	0.022	0
4	3	4	13	36	0.129	0.008	0.275	0.338	1	48	2.859	0.007	0.274	0.494	4
4	4	4	17	72	0.147	0.012	0.523	1.492	2	108	16.699	0.009	0.539	1.607	19
5	3	5	16	82	0.345	0.011	0.492	1.102	2	140	20.327	0.007	0.657	1.198	22
5	4	5	21	86	1.134	0.019	0.734	1.756	4	118	86.846	0.020	0.963	2.303	90
5	5	5	26	116	1.334	0.030	1.438	18.922	22	282	676.806	0.090	3.552	10.600	691
6	4	6	25	98	4.914	0.027	1.329	9.308	16	224	463.672	0.064	2.663	76.640	543
6	5	6	31	130	5.827	0.058	2.761	18.910	28	252	1,808.483	0.122	5.158	105.383	1,920
6	6	6	37	180	11.212	0.138	4.556	257.973	274	120	1,805.300	0.100	4.689	10.331	1,821
7	5	7	36	146	44.331	0.093	3.999	64.125	113	124	1,815.915	0.092	4.871	27.963	1,849
7	6	7	43	208	100.625	0.189	7.839	403.585	513	64	1,843.678	0.084	4.449	5.225	1,854
7	7	7	50	252	110.525	0.349	19.354	900.100	1,031	38	1,852.437	0.099	8.412	40.648	1,902
8	6	8	49	294	1,023.991	0.396	11.819	267.337	1,304	38	1,821.465	0.078	4.385	14.908	1,841
8	7	8	57	368	1,142.769	0.719	22.448	900.098	2,067	24	1,883.243	0.094	6.436	4.662	1,895
8	8	8	65	324	1,788.235	0.888	36.869	900.150	2,728	16	1,843.733	0.096	3.007	0.827	1,848
9	7	9	64	184	1,790.911	0.550	27.985	115.887	1,936	16	1,856.004	0.091	5.362	1.607	1,863
9	8	9	73	40	1,824.380	0.227	18.971	2.308	1,849	38	1,841.109	0.223	21.888	7.818	1,872
9	9	9	82	42	1,882.879	0.419	12.164	3.809	1,901	30	1,929.489	0.273	17.502	13.083	1,961

4.6 Rounding analysis

Next, we compare the four rounding methods and investigate the effect of the line pool size on the quality of the MILP solution. We observed that the lines that are used in the final MILP solution are usually generated very early on in the solving process, and that most of the time towards the end is spent generating lines that do not end up being used.

In this comparison, we examine how often it is preferable to use Generation-Order Greedy Rounding rather than Frequency-Based Greedy Rounding. Furthermore, we compare the effect of a limited line pool of “Early Stop MILP” against the fully generated line pool of “Converged MILP”. In total, we run the problem set on the ring graphs three times: once for GO and ES MILP, once for FB, and once for CO MILP.

Table 4 compares the line selection criteria and the total runtime. When using TWCG, Generation-Order Greedy Rounding yields a better solution in 15 out of 21 instances. This provides evidence that, if processing time is a factor, it is possible to stop early. If the objective value of the MILP solution increases after adding a line and path set, we can remove the last line and stop.

Comparing GO against ES MILP, they appear to yield very similar solution values. The biggest difference has been observed with 9 spokes and 7 rings, resulting in a cost of 0.944 using GO, compared to 0.943 with ES MILP. This is surprising, given that GO naively rounds the LP relaxation, whereas ES MILP recalculates the MILP. However, since the runtime barely increases when moving from GO to ES MILP – at most 4 seconds on the instance with 9 spokes and 9 rings – it is always sensible to run ES MILP as well.

Using GO instead of FB, we have an average improvement of 13.7%, with the biggest improvement being 44% for 8 spokes and 7 rings, with a cost reduction from 1.501 to 0.838. The biggest worsening of 9.4% for 6 spokes and 4 rings would lead to a cost increase from 0.449 to 0.491. This, combined with a runtime speed-up by a factor of 10 to 20, makes GO the preferable method over FB.

The cost of the two MILP rounding methods is usually very close to each other. In some cases, the ES MILP actually finds a very slightly better solution. This is (again) due to nondeterminism in the tree decomposition generation. Even though the CO MILP solves the LP relaxation to optimality, it may end up with a different set of lines that behave differently during the rounding phase.

Furthermore, the larger line pool will sometimes cause the CO MILP to run out of time, as is the case with 8 spokes and 7 or 8 rings. If running CO MILP instead of ES MILP, the biggest improvement is 13.3%, with 9 spokes and 9 rings, from 1.021 to 0.886, while the lowest “improvement” is -20%, with an average improvement of 0.5%. For instances with more than 5 rings, we achieve, on average, about 20× faster runtime when using ES MILP. In the largest instances, this is no longer visible because the CO MILP runs out of time during column generation.

Overall, CO MILP gives the best solutions at the cost of a significantly increased runtime. As the average gap is only 0.5%, this experiment justifies the decision to use ES MILP for the Japan network and to stop early.

■ **Table 4** Comparison of heuristic and MILP rounding. The left block describes the instances in which s and r denote the number of spokes and rings of the circular city. The second block shows “Generation-Order Greedy Rounding” (GO) with its runtime in seconds, number of lines, and cost. The following blocks show the respective values for “Early Stop MILP” (ES MILP), “Frequency-Based Greedy Rounding” (FB), and “Converged MILP” (CO MILP).

Input		GO			ES MILP			FB			CO MILP		
s	r	t	l	cost $\times 10^5$	t	l	cost $\times 10^5$	t	l	cost $\times 10^5$	t	l	cost $\times 10^5$
3	1	0.1	2	0.014	0.1	2	0.014	0.8	2	0.014	1.5	2	0.014
3	2	0.1	2	0.065	0.2	2	0.065	0.2	3	0.076	0.2	3	0.075
3	3	0.1	2	0.166	0.2	2	0.166	0.3	3	0.195	0.5	3	0.186
4	2	0.1	2	0.125	0.1	2	0.125	0.2	2	0.123	0.2	3	0.150
4	3	0.1	2	0.250	0.2	2	0.249	0.5	2	0.244	0.5	2	0.231
4	4	0.1	2	0.358	0.3	2	0.358	0.8	2	0.345	0.9	2	0.325
5	3	0.1	2	0.318	0.3	2	0.318	0.9	3	0.339	1.5	2	0.301
5	4	0.4	2	0.373	0.7	2	0.373	2.0	2	0.426	1.5	2	0.378
5	5	0.3	2	0.514	0.7	2	0.514	2.9	3	0.525	2.8	2	0.448
6	4	0.6	2	0.491	0.7	2	0.491	6.4	2	0.449	8.2	2	0.440
6	5	0.4	2	0.502	0.8	2	0.502	8.8	4	0.654	12.5	3	0.550
6	6	1.0	3	0.639	1.7	3	0.639	16.2	4	0.737	8.0	3	0.633
7	5	4.4	3	0.617	5.0	3	0.617	48.6	4	0.713	47.4	3	0.613
7	6	5.8	4	0.747	6.3	4	0.746	109.0	3	0.708	126.7	3	0.708
7	7	8.5	3	0.779	9.2	3	0.779	130.7	7	1.093	1048.1	3	0.817
8	6	46.2	3	0.756	47.2	3	0.756	1037.0	8	1.168	1231.0	3	0.690
8	7	48.4	4	0.838	50.2	4	0.838	1166.9	11	1.501	2701.5	3	0.847
8	8	68.0	4	0.896	69.6	4	0.895	1827.5	8	1.340	2705.9	4	0.905
9	7	84.6	4	0.944	87.1	4	0.943	1820.3	9	1.390	1839.2	4	0.906
9	8	487.1	4	0.938	489.8	4	0.937	1846.3	7	1.426	1850.6	5	0.960
9	9	354.2	4	1.021	358.1	4	1.021	1897.3	4	1.053	1917.1	3	0.886

4.7 Comparison against COLOR CODING on Real-life networks

Next, we discuss the real-world examples: the Munich S-Bahn and subway network and the Japanese railway network. Table 5 shows details on the solution quality for both locations.

■ **Table 5** Comparisons of the solutions of Munich and Japan with CC and TWCG. Top-left: the network size ($|V|$, $|E|$) and the transport demand (Σd_{st}). Top-right: LP cost, MILP cost, travel cost, and operating cost. Bottom left: the number of active lines, the average frequency, and the average length of the lines. Bottom right: the cumulative time to price the lines and paths, the cumulative time to solve the LPs, and the time to solve the MILP.

Instance		Network			Costs			
Place	Method	$ V $	$ E $	Σd_{st}	LP	MILP	$cost_{travel}$	$cost_{var}$
Japan	CC	694	830	74,672	366,327	1,467,452	97,540	1,369,913
Japan	TWCG	694	830	74,672	338,468	776,061	96,705	679,356
Munich	CC	251	249	3,137,500	184,819	361,748	137,564	224,183
Munich	TWCG	251	249	3,137,500	184,840	333,373	139,013	194,360

Instance		Solution Structure			Time (s)			
Place	Method	act. Lines	$\bar{f}$	avg. len.	lines	paths	LP solve	MILP solve
Japan	CC	116	1.643	12.103	1,852	715	1,405	900
Japan	TWCG	46	0.349	121.370	16	206	2,331	18
Munich	CC	16	1.885	20.688	644	77	1,767	900
Munich	TWCG	13	1.543	27.231	1	61	1,310	197

In the Munich case, we use $k = 70$ colors and 20 iterations for COLOR CODING. In theory, using such a large number of colors should be infeasible; however, as our implementation is “positive-instance driven” (i.e., we do not enumerate infeasible entries of the DP table), it turns out to be feasible to use a large number of colors, as the very simple structure of the Munich network keeps the number of feasible solutions bounded.

The algorithm achieved an LP cost of 184,819; the travel cost is 137,564 and the operating cost is 224,183, yielding an MILP cost of 361,748. The solution has 16 active lines with an average frequency of 1.885 and an average length of 20.688. The time spent on finding the lines and paths, and on solving the LP relaxation and the MILP, is 644, 77, 1,767, and 900 seconds, respectively. The MILP timed out after 15 minutes; the MILP gap is 2.3%.

Using TWCG, we get an LP cost of 184,840; the travel cost is 139,013, and the operating cost is 194,360, yielding an MILP cost of 333,373. This results in a 7.8% reduction in MILP cost compared to COLOR CODING. The solution has 13 active lines with an average frequency of 1.543 and an average length of 27.231. The time spent on finding the lines and paths, and on solving the LP relaxation and the MILP, is 1, 61, 1,310, and 197 seconds, respectively.

Next, we compare the solutions for the Japanese network. Using COLOR CODING with $k = 20$ colors and 20 iterations, an LP cost of 366,327 is achieved. The travel cost is 97,540 and the operating cost is 1,369,913, yielding an MILP cost of 1,467,452. The solution has 116 active lines with average frequency 1.643 and average length 12.103. Time spent on pricing the lines, paths, and solving the LP relaxation and MILP is 1,852, 715, 1,405, and 900 seconds, respectively. The MILP timed out after 15 minutes; the MILP gap is 2.0%.

Using TWCG, the LP cost is 338,468; the travel cost is 96,705, and the operating cost is 679,356, yielding an MILP cost of 776,061. This results in a 47.1% reduction in MILP cost compared to COLOR CODING. The solution has 46 active lines with an average frequency of 0.349 and an average length of 121.370. The time spent on finding the lines and paths, and on solving the LP relaxation and the MILP, is 16, 206, 2,331, and 18 seconds, respectively. We observe that the bottleneck is the LP solver, as we spend only 16 seconds solving the pricing problem in the Japan case, but 39 minutes solving the LP relaxation with Gurobi.

The large difference between LP and MILP costs is because the average frequency is smaller than F , and consequently, the fixed costs are underestimated in the LP relaxation. In practice, this could be improved by tightening the estimate of F .

Using TWCG, we obtain average line lengths of 27.2 and 121.4 for Munich and Japan, respectively. Even if one were to use more colors for COLOR CODING, it is (due to its exponential behavior) impractical to use anywhere near that many colors. Munich allowed us to use many colors, but this is due to the very low average degree. We also see the advantage of TWCG in finding the lines. In the Japan case, with 694 stations, COLOR CODING accounts for over a third of the time spent searching for the lines, whereas using TWCG reduces the time to 16 seconds in total. Also, the solution quality showed the largest differences in real-world scenarios, with a 47.1% improvement in Japan and a 7.8% improvement in Munich. It is also notable that the LP costs in Munich are nearly identical; in the Japan case, however, there is a difference of 7.6%. The latter is due to early termination.

5 Conclusion

We have shown that column generation for the Line Planning Problem can be performed effectively by using treewidth-based dynamic programming for the pricing problem. We compared runtime, costs, and rounding methods for concentric ring graphs, the Munich S-Bahn and subway network, and the Japanese railway network. We improve on existing methods by solving bigger instances with higher treewidth and by improving solution quality. Furthermore, we improve on greedy rounding by introducing an alternative selection criterion, which also reduces runtime.

We can solve instances with treewidth up to 9. For treewidth up to 7, the time required to price lines is comparably negligible, as shown by the example of Japan. The computational bottleneck for networks with a large number of vertices is the LP solver. The bottleneck for solving instances with larger treewidth is the join operator. Switching to path decompositions, or using tree decompositions which have been optimized for a measure of DP “cost” rather than width, could be beneficial. However, unlike for treewidth, there is currently a lack of good pathwidth (or treecost) solvers.

Interestingly, Ziobro and Pilipczuk [29] report in their experimental evaluation of treewidth-based dynamic programming for Hamiltonian Cycle that they are able to solve some instances with treewidth up to 29 (even with the “naive” algorithm without the rank-based approach). This may seem like a large gap compared to our results, which are for a very similar algorithm. However, it is likely that these instances have a structure that limits the number of partial solutions considered to far below the worst-case bound. Even though Longest Path and Hamiltonian Cycle may seem like similar problems, in Hamiltonian Cycle, many partial solutions can likely be discarded early, since failing to visit even one vertex renders the solution infeasible immediately. In contrast, a partial solution for Longest Path cannot be excluded just because it skips one vertex.

TREewidth COLUMN GENERATION outperforms COLOR CODING because it selects longer, more effective lines while using fewer lines. This is particularly evident in real-world examples. TWCG is faster for Munich and Japan. Depending on the parameter choices, COLOR CODING can still be faster, but this results in shorter lines and a worse solution. Both heuristic and MILP-based rounding achieve significant time savings by terminating the column generation early. Our experiments show that this generally still results in good-quality solutions.

Compared to Heinrich et al.'s TW-LINES [17], we are able to handle larger networks with higher treewidth. However, it is difficult to fairly compare the two methods: Heinrich et al. solve a different problem without fixed costs per line and without passenger routing, but with integer frequencies, and solve to optimality rather than rounding an LP relaxation. Although directly using treewidth-based FPT algorithms to holistically solve line planning problems as Heinrich et al. do can provide superior (i.e., provably optimal) solutions, this is limited to cases which happen to admit FPT algorithms. Even then, the complexity of the resulting treewidth DP formulation can severely limit the values of treewidth for which such an approach is feasible. In contrast, our approach, which uses treewidth techniques combined with column generation, allows us to exploit bounded treewidth where it is most advantageous and use it to solve more complicated problems.

References

- 1 Oliver Bachtler, Irene Heinrich, Philine Schiewe, and Constantin Seebach. Non-pool-based line planning: a practical analysis of the FPT algorithm tw-lines. *Transportation Research Procedia*, 86:290–297, 2025. doi:10.1016/j.trpro.2025.04.037.
- 2 Max Bannach, Sebastian Berndt, and Thorsten Ehlers. Jdrasil: A Modular Library for Computing Tree Decompositions. In Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman, editors, *16th International Symposium on Experimental Algorithms (SEA 2017)*, volume 75 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SEA.2017.28.
- 3 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information & Computation*, 2015. doi:10.1016/j.ic.2014.12.008.
- 4 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996. doi:10.1006/jagm.1996.0049.
- 5 Ralf Borndörfer, Martin Grötschel, and Marc E. Pfetsch. A column-generation approach to line planning in public transport. *Transportation Science*, 2007. doi:10.1287/trsc.1060.0161.
- 6 Oded Cats, Roberto F. Abenzoza, Chengxi Liu, and Yusak O. Susilo. Evolution of satisfaction with public transport and its determinants in Sweden: Identifying priority areas. *Transportation Research Record*, 2538(1):86–95, 2015. doi:10.3141/2538-10.
- 7 M.T. Claessens, Nico M. van Dijk, and Peter J. Zwaneveld. Cost optimal allocation of rail passenger lines. *European Journal of Operational Research*, 110(3):474–489, 1998. doi:10.1016/S0377-2217(97)00271-3.
- 8 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 9 Holger Dell, Thore Husfeldt, Bart Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The first parameterized algorithms and computational experiments challenge. *International Symposium on Parameterized and Exact Computation*, 2017. doi:10.4230/LIPIcs.IPEC.2016.30.
- 10 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. *International Symposium on Parameterized and Exact Computation*, 2018. doi:10.4230/LIPIcs.IPEC.2017.30.
- 11 Jacques Desrosiers and Marco E. Lübbecke. *A primer in column generation*, pages 1–32. Springer US, Boston, MA, 2005. doi:10.1007/0-387-25486-2_1.
- 12 John Farrington and Conor Farrington. Rural accessibility, social inclusion and social justice: towards conceptualisation. *Journal of Transport Geography*, 13(1):1–12, 2005. Sustainability and the Interaction Between External Effects of Transport (Part Special Issue, pp. 23-99). doi:10.1016/j.jtrangeo.2004.10.002.

- 13 Huiling Fu, Lei Nie, Lingyun Meng, Benjamin R. Sperry, and Zhenhuan He. A hierarchical line planning approach for a large-scale high speed rail network: The China case. *Transportation Research Part A: Policy and Practice*, 75:61–83, 2015. doi:10.1016/j.tra.2015.03.013.
- 14 Philine Gattermann, Jonas Harbering, and Anita Schöbel. Line pool generation. *Public Transport*, 9(1):7–32, 2017. doi:10.1007/s12469-016-0127-x.
- 15 Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, 2024. URL: <https://www.gurobi.com>.
- 16 Irene Heinrich, Philine Schiewe, and Constantin Seebach. Algorithms and hardness for non-pool-based line planning. *Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, 2022. doi:10.4230/OASICS.ATMOS.2022.8.
- 17 Irene Heinrich, Philine Schiewe, and Constantin Seebach. Non-pool-based line planning on graphs of bounded treewidth. *Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, 2023. doi:10.4230/OASICS.ATMOS.2023.4.
- 18 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.
- 19 Jingwen Huo, Jing Meng, Heran Zheng, Priti Parikh, and Dabo Guan. Achieving decent living standards in emerging economies challenges national mitigation goals for CO₂ emissions. *Nature Communications*, 14(1):6342, 2023. doi:10.1038/s41467-023-42079-8.
- 20 Münchner Verkehrs- und Tarifverbund. S-Bahn, U-Bahn Netzplan München, 2024. Map. URL: <https://www.mvv-muenchen.de/>.
- 21 OpenJDK. JDK 22. <https://openjdk.org/projects/jdk/22/>, 2024. Accessed: 2026-08-03.
- 22 Laurent Perron and Vincent Furnon. OR-Tools. URL: <https://developers.google.com/optimization/>.
- 23 Toshiki Saitoh. Personal communication regarding the Japanese rail network. Personal communication, 2016.
- 24 Anita Schöbel. Line planning in public transportation: models and methods. *OR Spectrum*, 2012. doi:10.1007/s00291-011-0251-6.
- 25 Anita Schöbel and Susanne Scholl. Line planning with minimal traveling time. In Leo G. Kroon and Rolf H. Möhring, editors, *5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05)*, volume 2 of *Open Access Series in Informatics (OASICS)*, pages 1–16, Dagstuhl, Germany, 2006. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.ATMOS.2005.660.
- 26 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, Berlin, 2003.
- 27 Hisao Tamaki. Heuristic computation of exact treewidth. In *20th International Symposium on Experimental Algorithms (SEA 2022)*, pages 17:1–17:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SEA.2022.17.
- 28 Hisao Tamaki. A contraction-recursive algorithm for treewidth. *International Symposium on Parameterized and Exact Computation*, 2023. doi:10.48550/arXiv.2307.01318.
- 29 Michał Ziobro and Marcin Pilipczuk. Finding Hamiltonian Cycle in graphs of bounded treewidth: Experimental evaluation. *Journal of Experimental Algorithmics (JEA)*, 24:1–18, 2019. doi:10.1145/3368631.

Algorithmics for Safe Bicycle Network Design with Bounded Detours in Rural Areas

Till Fluschnik 

Department of Computer Science, Algorithm Engineering Group, Humboldt-Universität zu Berlin, Germany

Abstract

We introduce the *Safe Bicycle Network with Bounded Detours (SBNBD)* problem, a network-design problem motivated by the upgrade of rural road networks for bicycle traffic. Given an undirected graph whose edges have length and are classified as safe or unsafe with unsafe edges carrying upgrade costs, a set of terminal pairs, an upgrade budget, and a detour factor α , the task is to upgrade unsafe edges so that every terminal pair is connected by a safe path whose length is at most α times its shortest-path distance in the original network.

We study the computational complexity of SBNBD from a parameterized perspective. We prove strong NP-hardness even on highly restricted graph classes, including planar graphs of treewidth two, graphs with feedback vertex set number one, and graphs of maximum degree three. We complement these lower bounds with polynomial-time algorithms for trees and graphs of maximum degree two. For parameterized complexity, we show fixed-parameter tractability for the number of unsafe edges and prove matching lower bounds under SETH, a polynomial-kernel lower bound, and W-hardness for natural parameters. Our main positive structural result is an algorithm that maps any instance to an equivalent instance with $O(\text{fes} + p)$ vertices and edges, where fes is the feedback edge number and p is the number of terminal pairs; this yields fixed-parameter tractability for the combined parameter $\text{fes} + p$.

Finally, we construct and evaluate exact ILP-based algorithms on road networks derived from OpenStreetMap data for small German municipalities and their surrounding rural regions. The experiments show that these networks have small treewidth upper bounds and moderate feedback edge structure. We introduce a preprocessing routine based on the reduction algorithm for the parameter $\text{fes} + p$ and a heuristic cut generation based on tree decompositions. Both improve exact solving performance, in particular for harder instances. Our results for different detour-factor bounds show that a moderate increase in α can substantially reduce the total length of the upgraded network, revealing practical trade-offs between upgrade budget and the maximum allowed relative detour. Our results indicate that structural graph parameters provide a useful algorithmic lens for safe bicycle-network design in rural areas.

2012 ACM Subject Classification Mathematics of computing $\rightarrow$ Combinatorial optimization; Theory of computation $\rightarrow$ Design and analysis of algorithms; Social and professional topics $\rightarrow$ Sustainability

Keywords and phrases Parameterized Algorithms, NP-hardness, Pairwise weighted spanners, Pre-processing, Integer linear programming, Cut generation

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.14

Related Version *Full Version*: <https://arxiv.org/abs/2608.09472>

Supplementary Material *Software*: <https://github.com/buhtig-tf/Safe-Bicycle-Network-Design-with-Bounded-Detours>

archived at `swb:1:dir:13d402478fe99ff904d028497a6b10b5f5c53119`

Funding *Till Fluschnik*: Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project PACS (FL 1247/1-1, 522475669).



© Till Fluschnik;

licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 14; pp. 14:1–14:22

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Improving bicycle networks for urban and rural areas can help to reduce pollution [35], promote physical activity [37], and mitigate transport poverty [45]. Compared with urban areas, both public transport and bicycle infrastructure are often insufficient in rural regions [46]. Since improving public transport in rural areas is often economically challenging [17] and people use cars even for short-distance rides [2, 23], cost-effective improvement of bicycle networks becomes a central planning task. In Germany, 54% of respondents living in rural areas with less than 20,000 inhabitants report that they would like to use bicycles more frequently in the future [42]. High-quality, connected cycling infrastructure could roughly triple cycling use by 2035 (up to 45% modal share on short trips), with significant untapped potential also in rural areas [12]. Other mobility solutions like bike-sharing for rural areas [38] benefit from such infrastructure. Safety and bounded detour length [20, 40] are key criteria for a network's quality.

This paper treats the problem of cost-optimal infrastructure upgrades in a given network to enable safe bicycle trips for specified terminal pairs with only small detours. Herein, the network's edges are classified either as safe or unsafe, corresponding to low- and high-stress segments [21]. Our model is based on the bicycle network improvement problem due to Lim et al. [32]. They optimize an aggregate objective over weighted origin-destination pairs, while we impose hard relative detour constraints. They study the problem from an OR perspective, while we study the problem from a computational and parameterized complexity perspective, with a focus on graph-theoretical properties derived from the applications on rural areas. Several models similar to ours are already studied in the scientific literature, yet a fine-grained computational and parameterized complexity analysis seems to be missing to date. We complement our complexity analysis with experimental algorithmics.

1.1 Our Model and Decision Problem

We model the street network as an undirected graph G with a set V of vertices (representing junctions or points of interest like schools) and a set $E \subseteq \binom{V}{2}$ of edges (representing street segments), where the edge set E is partitioned into two sets E_s and E_u of *safe* and *unsafe* edges. Each edge is equipped with a (travel) length $\theta: E \rightarrow \mathbb{N}$, representing the length of or the time needed to traverse the street segment. Moreover, each unsafe edge is equipped with a cost $c: E_u \rightarrow \mathbb{N}$, representing the cost to make it safe (e.g., by constructing a bike lane on this street segment). For an edge subset $F \subseteq E_u$, we denote by $G\langle F \rangle$ the graph G with edge set partitioned into $E'_s = E_s \cup F$ and $E'_u = E_u \setminus F$.

A terminal pair consists of two distinct vertices from V , representing the endpoints of a requested bicycle trip. For two distinct vertices $s, t \in V$, an s - t path P is a sequence $(v_0, v_1, \dots, v_\ell)$ of distinct vertices from V with $v_0 = s$ and $v_\ell = t$ such that $\{v_{i-1}, v_i\} \in E$ for all $i \in [\ell]$. The length $\text{len}(P)$ of P is $\sum_{i=1}^{\ell} \theta(\{v_{i-1}, v_i\})$. With $\text{dist}_G(s, t)$ we denote the smallest length of an s - t path in G . We call path P *safe* if $\{v_{i-1}, v_i\} \in E_s$ for all $i \in [\ell]$, and *unsafe* otherwise. We often drop the graph from the notation when it is clear from the context. The detour factor of an s - t path P in G is $\text{len}(P) / \text{dist}(s, t) \in \mathbb{Q}_{\geq 1}$.

► **Problem 1 (Safe Bicycle Network with Bounded Detours (SBNBD)).** **Given** a network $G = (V, E, c, \theta)$ with $E = E_s \uplus E_u$, a set of $p \in \mathbb{N}$ terminal pairs $\mathcal{P} = \{\{s_i, t_i\} \mid i \in [p]\}$ from V , a budget $k \in \mathbb{N}_0$, and a number $\alpha \in \mathbb{Q}_{\geq 1}$, the **question** is whether there is a solution $F \subseteq E_u$ with $c(F) = \sum_{e \in F} c(e) \leq k$ such that for each $i \in [p]$, there is a safe s_i - t_i path in $G\langle F \rangle$ of detour factor at most α .

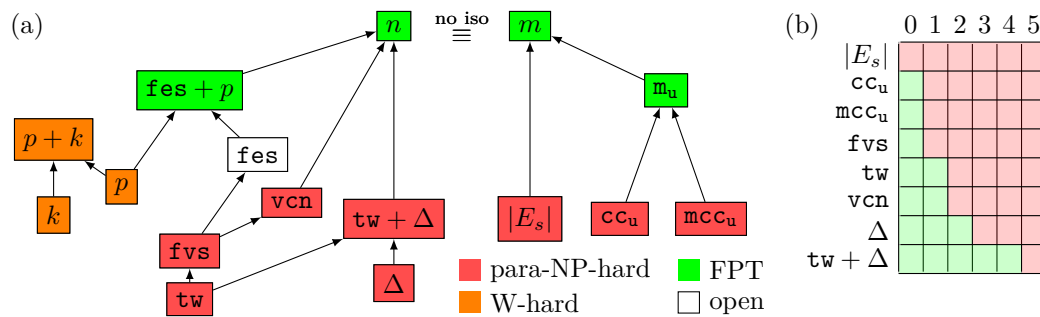


Figure 1 (a) Hasse diagram of our studied parameters with their complexity classification assuming unary encoding. Number n of vertices and number m of edges are parameterized equivalent when there are no isolated vertices. A parameter x points to another parameter y if there is a function f such that $x \leq f(y)$ holds for every instance. Thus, fixed-parameter tractability for x transfers to y , while hardness for y transfers to x . (b) Overview of the polynomial-time solvability (green) versus NP-hardness (red) borders for para-NP-hard parameters. (m_u : number of unsafe edges; cc_u : number of components in the graph induced by all unsafe edges; mcc_u : size of the largest component in the graph induced by all unsafe edges.)

1.2 Our Contributions

Our contributions are two-fold. First, we provide a computational and parameterized complexity analysis for SBNBD. Second, we study exact ILP-based algorithms experimentally, focusing on the impact of reduction rules and treewidth-based cut generation.

Computational and parameterized complexity. Figure 1 arranges our complexity-theoretic results. We prove that SBNBD is NP-hard in very restricted settings, such as vertex cover number two, or treewidth two and maximum degree three. Each of our para-NP-hardness results is tight in the sense that the next smaller parameter value leads to polynomial-time solvability. We prove W[1]-hardness when parameterized by the number of terminal pairs combined with the budget. For binary-encoded edge lengths and costs, we prove NP-hardness already for one terminal pair. Finally, we show several reduction rules that transform any instance to an equivalent instance whose combinatorial size is linear in the number of terminal pairs and the feedback edge number, i.e., the size of a minimum feedback edge set. These reduction rules are also effective as preprocessing in our experiments.

Experimental algorithmics. Using OpenStreetMap data, we extracted undirected road networks for 30 German villages and their 3 km surrounding rural areas. These villages were selected from a ranking of over 400 German villages according to their perceived bicycle friendliness. For these networks, we then computed the maximum degree, the size of a minimum feedback edge set, small upper bounds on the treewidth, and checked for planarity. Our main findings are that these networks have small maximum degree ≤ 5 , small treewidth upper bound ≤ 14 , and roughly 20% of the edges suffice on average as a feedback edge set.

We tested a plain ILP against the ILP with preprocessing via reduction rules, with additional cuts derived from a tree decomposition (TD), and with both. Our results are:

1. Most instances obtained for villages are solvable within seconds by each of the solvers.
2. For the larger instances, preprocessing combined with cut generation performed best.
3. The total length of upgraded unsafe edges decreases by 7.87% on average when α increases from 1.2 to 1.3, illustrating the trade-off between allowed detours and required upgrades.

2 Related Work

Improving bicycle networks. One line of research develops computational optimization models for bicycle-network design and improvement. Lim et al. [32] consider weighted terminal pairs and optimize the weighted detour penalty aggregated over each trip. They formulate their problem as mixed integer program (MIP) and solve it via Benders decomposition on a case-study instance. Duthie and Unnikrishnan [15] study “retrofitting” segments at minimum cost to ensure a minimum safety level with a given bounded detour. Compared to our model, their segments can have several safety levels and they express their length-bound through a function of the shortest path. While structurally, their focus is on roadway quality, ours is on parameters observed in rural areas. Mauttone et al. [34] model user route choices in a multi-commodity-flow MIP solved heuristically, allowing routes to use segments without bicycle infrastructure at higher cost. In contrast, we require each terminal pair to have a safe route of bounded relative detour. Natera et al. [36] use a greedy strategy to compute missing links to interconnect components of the bicycle network. Steinacker et al. [43] use greedy network sparsification to design efficient urban bicycle networks, incorporating cycling demand and route choices based on safety preferences. To the best of our knowledge, computational or parameterized complexity analyses are missing for bicycle-network improvement problems with infrastructure upgrades under hard detour constraints. Another line of research analyzes existing bicycle networks and their growth. Szell et al. [44] argue that uncoordinated enhancement of bicycle networks can lead to higher cost and outline the need for careful planning. Schoner and Levinson [40] analyzed 74 bicycle networks from the US and found that connectivity and directness are important factors. Finally, Furth et al. [21] developed a framework to classify street segments as low- to high-stress segments.

Algorithmics of related problems. By assigning zero cost to safe edges and positive cost to unsafe edges, our problem can be viewed as an undirected pairwise weighted spanner or as a distance-constrained Steiner-type problem. Cygan et al. [11] study sparse pairwise spanners in unweighted undirected graphs where the possible increase of a pairwise distance is controlled by a stretch function. They study existential bounds and give a polynomial-time construction for additive stretch functions. Approximation algorithms and hardness results for spanner-type problems have been studied on unweighted graphs [8] and weighted directed graphs [24]. Kobayashi [30] studies the all-pairs t -spanners in unweighted undirected graphs, where each pair’s distance is allowed to increase within a factor of t . They prove NP-hardness even in planar bounded-degree graphs. Moreover, they show an FPT algorithm when parameterized by the number of discarded edges (Kobayashi [31] proved such an algorithm to exist also for additive spanners). Regarding parameterized complexity, Feldmann and Lampis [18] study the related problem STEINER FOREST with terminal pairs (which has no detour constraints). They focus on structural parameters and prove an EPAS with running time depending on treewidth (the problem is para-NP-hard already for $\text{tw} = 3$ [22]). They also prove an FPT algorithm when parameterized by the size of a feedback edge set. Simonov [41] study the problem of computing pairwise distance preservers in undirected unweighted graphs, where no increase in path length is allowed between any terminal pair. They prove W[1]-hardness when parameterized by the number of terminal pairs and NP-hardness for vertex cover number $\text{vcn} = 3$. Thus, we can view our problem as an undirected common-stretch pairwise weighted spanner problem with designated safe edges of zero cost and unsafe edges of positive cost; to the best of our knowledge, a systematic parameterized complexity study is missing for this variant.

Structural parameters of real-world graphs. Maniu et al. [33] study the treewidth of 25 networks from eight different domains, including infrastructure. They find that many of these networks have treewidth too large for a direct application of treewidth-based algorithms. In contrast, we show that the street networks derived from rural areas admit small treewidth upper bounds. Analyzing large-scale street networks using OSM data has become common [4, 5, 16]. Related to the feedback edge number, the meshedness coefficient of planar real-world street networks is studied [7], which relates the numbers of bounded faces of the network and of a maximally connected planar graph with the same number of vertices. In connected planar graphs, the number of bounded faces equals the feedback edge number. To the best of our knowledge, classic FPT parameters such as treewidth or feedback edge number have not been measured for larger datasets of rural street networks.

3 Preliminaries

We denote by $\mathbb{N}$ and $\mathbb{N}_0$ the natural numbers excluding and including zero, respectively. We denote by $\mathbb{Q}$ and $\mathbb{Q}_{\geq 1}$ the rational numbers and the rational numbers at least one, respectively. We use basic notation from parameterized complexity [10]. We use common graph-theoretic notation and explain only notation that may be non-standard. For an undirected graph $G = (V, E)$, we denote the vertex and edge set of G by $V(G)$ and $E(G)$, respectively. For an edge subset $E' \subseteq E$, we denote by $G[E'] = (V', E')$ with $V' = \{v \in V \mid \exists e \in E' : v \in e\}$ the edge-induced graph G induced on the edge set E' . We write $G - E'$ for the graph $(V, E \setminus E')$. For a vertex subset $U \subseteq V$, we denote by $\partial_G(U) = \{\{v, w\} \in E \mid |\{v, w\} \cap U| = 1\}$ the *cut* for U in G , i.e., all edges in G with exactly one endpoint in U .

4 Tight Strong NP-hardness on Simple Graph Classes

► **Theorem 1.** *SBNBD is NP-hard even on planar graphs with treewidth $\text{tw} = 2$, and*

- (a) *feedback vertex set size $\text{fvs} = 1$, $\text{mcc}_u = 1$, and the graph is bipartite.*
- (b) *feedback vertex set size $\text{fvs} = 1$, vertex cover number $\text{vcn} = 2$, and $\text{cc}_u = 1$.*
- (c) *maximum degree $\Delta = 3$, $\text{mcc}_u = 1$, and the graph is bipartite.*

Moreover, unless the ETH breaks, there is no $2^{o(n+m)} \cdot \text{poly}(|I|)$ -time algorithm.

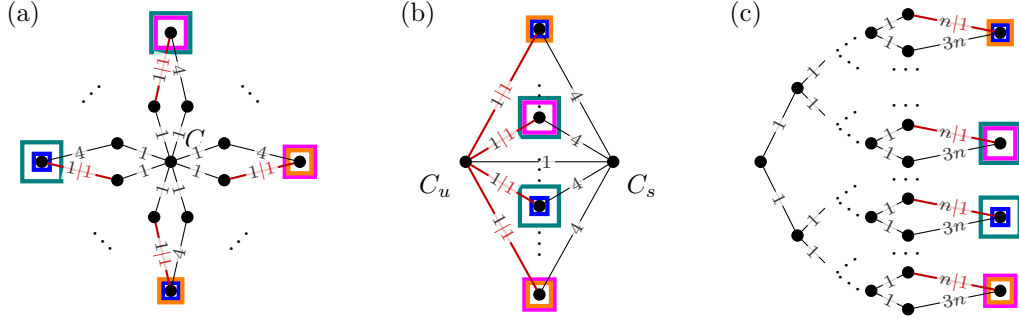
We give three polynomial-time many-one reductions, each from the NP-hard [28] VERTEX COVER problem, where, given an undirected graph $G = (V, E)$ and an integer r , the question is whether there is a subset $W \subseteq V$ with $|W| \leq r$ such that $e \cap W \neq \emptyset$ for every $e \in E$. VERTEX COVER admits no $2^{o(|V|+|E|)} \cdot \text{poly}(|I|)$ -time algorithm unless the ETH breaks [26, 27].

► **Construction 1.** Let $I = (G = (V, E), r)$ be an instance of VERTEX COVER with $\eta = |V|$ vertices. In each construction, we construct a graph G' that contains a vertex v^* for each vertex $v \in V$, the set of terminal pairs is $\mathcal{P} = \{\{v^*, w^*\} \mid e = \{v, w\} \in E\}$, and we have $k = r$. See Figure 2 for an illustration to each of the three constructions.

(a) Add a central vertex C . For each vertex $v \in V$, add two vertices v_u, v_s , and add unsafe edge $\{v_u, v^*\}$ of length 1 and cost 1, and safe edge $\{v^*, v_s\}$ of length 4. Make C adjacent to each of v_u, v_s for each $v \in V$ via a safe edge of length 1. Let $\alpha := 2$.

(b) There are two central vertices C_s, C_u (which form the vertex cover of the constructed graph) connected by a safe edge of length 1. For each vertex $v \in V$, connect v^* with C_s via a safe edge of length 4, and with C_u via an unsafe edge of length 1 and cost 1. Let $\alpha := 3$.

(c) We can assume that $\eta := |V| = 2^q$ for some $q \in \mathbb{N}$ and $2 \log(\eta) + 4 < \eta$ (by padding isolated vertices). Let G' be initially a binary tree T_η with set $\{\ell_v \mid v \in V\}$ of η leaves.



■ **Figure 2** Illustrations for Construction 1(a), (b), and (c). An unsafe edge e is drawn red and marked with its length and cost as $\theta(e)|c(e)$. Frames depict terminal pairs, i.e., two vertices with the same frame (size and color) form a terminal pair.

For each vertex $v \in V$, add two vertices v_u, v_s , and add unsafe edge $\{v_u, v^*\}$ of length η and cost 1, and safe edge $\{v^*, v_s\}$ of length 3η . Make ℓ_v adjacent with each of v_u, v_s for each $v \in V$ via a safe edge of length 1. Let $\alpha := 2$. $\lrcorner$

Each constructed network has treewidth at most two. For (a) and (b), removing C and C_u , respectively, leaves a forest. For (c), start with a tree decomposition of width 1 for T_η . For every $v \in V$, choose a node whose bag contains ℓ_v and append a node with bag $\{\ell_v, v_u, v_s\}$. To this new node, append another node with bag $\{v_u, v_s, v^*\}$.

Proof of Theorem 1. Let $I' = (G' = (V', E', c, \theta), \mathcal{P}, k, \alpha)$ be the instance of SBNBD obtained in polynomial time from an instance $I = (G = (V, E), r)$ of VERTEX COVER via Construction 1(a), (b), or (c). By construction, for each terminal pair $\{s, t\}$, every s - t path with detour factor at most α must contain an unsafe edge. For (a), the shortest path is of length 4, and the shortest safe path is of length $10 > 8 = \alpha \cdot 4$. For (b), the shortest path is of length 2, and the shortest safe path is of length $8 > 6 = \alpha \cdot 2$. For (c), the shortest path is of length $2\eta + 2 + x$, where x is the length of the shortest path through the binary tree, and the shortest safe path is of length $6\eta + 2 + x > 4\eta + 4 + 2x = \alpha \cdot (2\eta + 2 + x)$, using $x \leq 2 \log(\eta)$ and our assumption $2 \log(\eta) + 4 < \eta$. Consequently, for any solution F , every feasible safe s - t path in $G' \langle F \rangle$ must contain an edge of F .

Define the following convention for a subset F of unsafe edges and a set $W \subseteq V$: the unsafe edge incident with v^* is in F if and only if $v \in W$. Since in each of the three constructions (a), (b), and (c), each unsafe edge has cost one, the convention implies that F is of cost at most $k = r$ if and only if W is of size at most r . Now, for every terminal pair $\{v^*, w^*\} \in \mathcal{P}$, let $\{v, w\} \in E$ be the corresponding edge in the input graph. The following equivalences hold. There is a safe v^* - w^* path of detour factor at most α in $G' \langle F \rangle$ if and only if at least one of the two unsafe edges incident with v^* or w^* is in F . By our convention, we have that at least one of the two unsafe edges incident with v^* or w^* is in F if and only if at least one of v, w is in W . Finally, at least one of v, w is in W if and only if the edge $\{v, w\} \in E$ is covered by W . Thus, all terminal pairs admit a feasible safe path in $G' \langle F \rangle$ if and only if W is a vertex cover of G . Together with the cost correspondence, this proves that I is a *yes*-instance if and only if I' is a *yes*-instance.

Finally, note that in construction (b), G' contains $|V| + 2$ vertices and $2|V| + 1$ edges. Hence, the ETH-based lower bound follows. $\blacktriangleleft$

► **Remark 2.** The graph from Construction 1(b) also has neighborhood-diversity two: C_u, C_s form one type, and $V' \setminus \{C_u, C_s\}$ forms the other type. In fact, adding sufficiently long safe edges between any two non-adjacent vertices turns the graph into clique while preserving correctness, which results in neighborhood diversity one (but unbounded vertex cover number).

► **Remark 3.** None of the reductions has unit lengths. Yet, (a) and (c) can be turned via subdivisions into unit length; this is false for (b) as it would increase the vertex cover number.

In the remainder, we show that decreasing any of the bounds $\text{vcn} = 2$, $\text{fvs} = 1$, $\text{tw} = 2$, or $\Delta = 3$ by one leads to graph classes on which SBNBD is polynomial-time solvable.

► **Proposition 4.** *For every instance of SBNBD where the input graph is a tree, we can compute a minimum-cost solution in polynomial time. Thus, SBNBD is polynomial-time solvable on trees.*

The proof of Proposition 4, deferred to Appendix A.1, uses the following two reduction rules.

► **Reduction Rule 1** (Irrelevant pair). If a terminal pair is connected by a safe path of detour factor at most α , delete the terminal pair.

► **Reduction Rule 2** (Mandatory edge). Let terminal pair $\{s_r, t_r\}$ be not connected by a safe path of detour factor at most α . If an unsafe edge e lies on all s_r - t_r paths of detour factor at most α , then make e safe. If $c(e)$ exceeds the budget, then return a trivial *no*-instance, otherwise decrease the budget by $c(e)$.

Using Proposition 4, we also handle graphs of maximum degree at most two: path components are immediate, and for each cycle component we either upgrade all unsafe edges or delete one unsafe edge that remains unsafe, reducing the component to a path. The full proof is deferred to Appendix A.1.

► **Observation 5.** *SBNBD is polynomial-time solvable on graphs of maximum degree at most two.*

5 Number of Unsafe Edges

► **Observation 6.** *SBNBD can be solved in $2^{\mathfrak{m}_u} \cdot \text{poly}(|I|)$ time and hence is in FPT when parameterized by the number $\mathfrak{m}_u$ of unsafe edges.*

Observation 6 corresponds to testing all possible solutions in polynomial-time for feasibility. We show next that, assuming the SETH to hold, we cannot do better than this.

► **Theorem 7.** *SBNBD, even if all unsafe edges induce one component ($\text{cc}_u = 1$) and the input graph has diameter at most four, admits no $2^{\varepsilon \cdot \mathfrak{m}_u} \cdot \text{poly}(|I|)$ -time algorithm for any $\varepsilon < 1$ unless the SETH breaks and no problem kernel of size polynomial in $\mathfrak{m}_u$ unless $\text{NP} \subseteq \text{coNP/poly}$. Moreover, it is $W[2]$ -hard when parameterized by the budget k .*

We reduce from the HITTING SET problem, where we are given a set X of elements, a set $\mathcal{S}$ of nonempty subsets of X , and an integer $r \in \mathbb{N}$, and ask whether there is a subset $Y \subseteq X$ with $|Y| \leq r$ such that $S \cap Y \neq \emptyset$ for every $S \in \mathcal{S}$. HITTING SET is $W[2]$ -hard when parameterized by the solution size r [14] and admits no problem kernel of size polynomial in $|X|$ unless $\text{NP} \subseteq \text{coNP/poly}$ [13]. Moreover, unless the SETH breaks, HITTING SET admits no $2^{\varepsilon \cdot |X|} \cdot \text{poly}(|I|)$ -time algorithm for any $\varepsilon < 1$ [9].

► **Construction 2.** Let $I = (X, \mathcal{S}, r)$ be an instance of HITTING SET. Construct the graph G with vertex set $V := \{v_x \mid x \in X\} \cup \{w_S \mid S \in \mathcal{S}\} \cup \{v^*\}$, safe-edge set $E_s := \{\{v_x, w_S\} \mid x \in X, S \in \mathcal{S} : x \in S\}$, and unsafe-edge set $E_u := \{\{v_x, v^*\} \mid x \in X\}$. The length of each edge is one and the cost of each unsafe edge is one. Let $\mathcal{P} := \{\{v^*, w_S\} \mid S \in \mathcal{S}\}$ be the set of terminal pairs. Let $\alpha := 1$ and $k := r$. $\lrcorner$

Since every unsafe edge is incident with v^* , the subgraph induced by E_u is connected ($\text{cc}_u = 1$). For every $S \in \mathcal{S}$, since S is nonempty, w_S and v^* have a common neighbor v_x with $x \in S$. Thus, every two vertices are connected by a path of length at most four, and so G has diameter at most four. The correctness proof of Theorem 7 is deferred to Appendix A.1.

6 Number of Terminal Pairs

► **Theorem 8.** *SBNBD is weakly NP-hard even if the input graph is planar, of maximum degree at most four, and of treewidth two, all unsafe edges are pairwise disjoint ($\text{mcc}_u = 1$), and there is only one terminal pair.*

We reduce from the weakly NP-hard PARTITION problem, where, given a multiset $X = \{x_1, \dots, x_N\}$ of at least two positive integers with sum $\sum_{x_i \in X} x_i = 2T$, the question is whether there is a partition (X_1, X_2) of X such that $\sum_{x_i \in X_1} x_i = \sum_{x_i \in X_2} x_i = T$.

► **Construction 3.** Let $I = (X = \{x_1, \dots, x_N\})$ be an instance of PARTITION with sum $2T$. Construct the vertex set $V = \{v_i \mid i \in [0, N]\} \cup \{a_i, b_i \mid i \in [N]\}$ and the edge set as follows: For each $i \in [N]$, add the safe edges $\{v_{i-1}, a_i\}$ of length T and $\{v_{i-1}, b_i\}$ of length $2T$, the safe edge $\{b_i, v_i\}$ of length $2Nx_i$, and the unsafe edge $\{a_i, v_i\}$ of length T and cost x_i . The set $\mathcal{P}$ of terminal pairs consists only of one terminal pair $\{v_0, v_N\}$. Let $k := T$ and $\alpha := 2$. $\lrcorner$

The graph is outerplanar: an outerplanar embedding is obtained by placing v_0 at $(0, 0)$ and, for each $i \in [N]$, placing v_i at $(2i, 0)$, a_i at $(2i - 1, 1)$, and b_i at $(2i - 1, -1)$. Since the graph is outerplanar and contains a cycle, it has treewidth two. It has maximum degree four: as $N \geq 2$, each vertex v_i with $i \in [N - 1]$ has degree four, whereas each vertex a_i and b_i with $i \in [N]$, as well as v_0 and v_N , has degree two. Finally, the set $\{\{a_i, v_i\} \mid i \in [N]\}$ of unsafe edges consists of pairwise vertex-disjoint edges, and hence $\text{mcc}_u = 1$. The correctness proof of Theorem 8 is deferred to Appendix A.1.

► **Remark 9.** Note that fvs , and hence fes , is unbounded in the constructed graph. Yet, the graph has cutwidth two, witnessed by an ordering such as $(\dots, v_{i-1}, a_i, b_i, v_i, \dots)$, with exactly two edges between any two consecutive vertices.

Note that in the previous reduction there is only *one* terminal pair. For unary encodings, the reduction no longer applies; however, we obtain $W[1]$ -hardness in this case.

► **Theorem 10.** *SBNBD is $W[1]$ -hard when parameterized by the number p of terminal pairs combined with the budget k , even if $\text{cc}_u = 1$, all edges are unsafe, and all shortest terminal paths are of length three.*

We reduce from the MULTICOLORED CLIQUE problem, which asks, given a graph $G = (V, E)$ where $V = V_1 \uplus \dots \uplus V_r$ is partitioned into r color classes and at least one vertex from V_i is adjacent to a vertex from V_j for every distinct $i, j \in [r]$, whether there is a vertex set W such that W forms a clique in G and $|W \cap V_i| = 1$ for every $i \in [r]$. MULTICOLORED CLIQUE is $W[1]$ -hard when parameterized by r [19].

► **Construction 4.** Let $I = (G = (V = V_1 \uplus \dots \uplus V_r, E))$ be an instance of MULTICOLORED CLIQUE. Let G' be initially a copy of G . Make each edge in G' unsafe with cost 1 and length 1. For each color class add a verifier vertex w_i and make it adjacent by unsafe edges of cost r^3 and length 1 with all vertices from V_i . Let $\mathcal{P} := \{\{w_i, w_j\} \mid 1 \leq i < j \leq r\}$ be the set of terminal pairs. Let $\alpha := 1$ and $k := r \cdot r^3 + \binom{r}{2}$. $\lrcorner$

The correctness proof of Theorem 10 is deferred to Appendix A.1.

7 Feedback Edge Number

► **Theorem 11.** *Any instance of SBNBD can be mapped in polynomial time to an equivalent instance of SBNBD with $O(\text{fes} + p)$ vertices and edges.*

Note that together with Observation 6, it follows that SBNBD is FPT when parameterized by $\text{fes} + p$. Theorem 11 consists of carefully deleting leaves and compressing long paths in the tree $G - X$, where X denotes a minimum feedback edge set. Such an approach has already proved useful (see, e.g., [18, 29]). We call a vertex in $G = (V, E)$ with terminal pairs $\mathcal{P}$ and a feedback edge set $X \subseteq E$ *important* if it is contained in a terminal pair, incident with an edge of X , or is of degree at least three in $G - X$, and *unimportant* otherwise.

► **Reduction Rule 3 (Leaf).** If a leaf in $G - X$ is unimportant, delete it.

For terminal pairs $\mathcal{P}$ and a feedback edge set $X \subseteq E$, we call a path P in $G - X$ a *corridor* if it contains at least three vertices, each of its endpoints is important, and all its inner vertices are unimportant and of degree two in $G - X$. We exploit that any s - t path connecting a terminal pair $\{s, t\}$ contains either all edges of such a corridor or none. Hence, we can replace such a corridor by a path with at most two edges thereby preserving all relevant information (lengths and costs) for the instance. However, we need to carefully distinguish the type of the corridor in terms of contained safe and unsafe edges and whether its endpoints are connected by a feedback edge. Note that since P is a subgraph of $G - X$, if the endpoints of P are adjacent in G , then the edge belongs to X . Let v, w be the endpoints of corridor P . We call P *unsafe* if it contains at least one unsafe edge, and *safe* otherwise.

We call a safe corridor P *contractible* if (i) $\{v, w\} \notin E$, (ii) $e = \{v, w\} \in X \cap E_s$, or (iii) $e = \{v, w\} \in X \cap E_u$ and $\text{len}(P) \leq \theta(e)$, and *compressible* if it contains at least four vertices, $e = \{v, w\} \in X \cap E_u$, and $\text{len}(P) > \theta(e)$. A safe corridor can be neither contractible nor compressible, that is, when it has three vertices, $e = \{v, w\} \in X \cap E_u$, and $\text{len}(P) > \theta(e)$.

► **Reduction Rule 4 (Safe Corridor Contraction).** Let P be a contractible safe corridor with endpoints v, w .

1. If $\{v, w\} \notin E$, then remove all edges and inner vertices from P and add the edge $e = \{v, w\}$ to E_s and set $\theta(e) = \text{len}(P)$.
2. If $e = \{v, w\} \in X \cap E_s$, then remove e and all edges and inner vertices from P , add a new safe edge e' and set $\theta(e') = \min\{\theta(e), \text{len}(P)\}$.
3. If $e = \{v, w\} \in X \cap E_u$ and $\text{len}(P) \leq \theta(e)$, then remove e and all edges and inner vertices from P and add the edge $e' = \{v, w\}$ to E_s and set $\theta(e') = \text{len}(P)$.

Reduction Rule 4 is correct since an unsafe alternative not shorter than the safe corridor is irrelevant, and among two safe options, the longer is irrelevant. The next rule treats a shorter unsafe alternative to the safe corridor. In this case, we have to keep both, but can compress the safe corridor into a path with two edges that preserves the length of the corridor.

► **Reduction Rule 5** (Safe Corridor Compression). Let P be a compressible safe corridor with endpoints v, w and let $e = \{v, w\} \in X \cap E_u$. Then remove all edges and inner vertices from P , add a new vertex $x_{v,w}$ and the edges $e' = \{v, x_{v,w}\}, e'' = \{x_{v,w}, w\}$ to E_s , and set $\theta(e') = \text{len}(P) - 1$ and $\theta(e'') = 1$.

Let P be an unsafe corridor with total cost $M := \sum_{e \in E(P) \cap E_u} c(e)$. We call P *contractible* if (i) $\{v, w\} \notin E$, (ii) $e = \{v, w\} \in X \cap E_s$ and $\text{len}(P) \geq \theta(e)$, or (iii) $e = \{v, w\} \in X \cap E_u$ and

$$(\text{len}(P) \leq \theta(e) \wedge M \leq c(e)) \vee (\text{len}(P) \geq \theta(e) \wedge M \geq c(e)). \quad (1)$$

We call P *compressible* if it contains at least four vertices and either (i) $e = \{v, w\} \in X \cap E_s$ and $\text{len}(P) < \theta(e)$, or (ii) $e = \{v, w\} \in X \cap E_u$ and (1) does not hold. An unsafe corridor can be neither contractible nor compressible, that is, when it has three vertices and the conditions (i) or (ii) for compressible corridors hold.

► **Reduction Rule 6** (Unsafe Corridor Contraction). Let P be a contractible unsafe corridor with endpoints v, w . Let $M := \sum_{e \in E(P) \cap E_u} c(e)$ be the total cost of P .

1. If $\{v, w\} \notin E$, then remove all edges and inner vertices from P and add the edge $e = \{v, w\}$ to E_u and set $\theta(e) = \text{len}(P)$ and $c(e) = M$.
2. If $e = \{v, w\} \in X \cap E_s$ and $\text{len}(P) \geq \theta(e)$, then remove e and all edges and inner vertices from P , add the safe edge $e' = \{v, w\}$ and set $\theta(e') = \theta(e)$.
3. If $e = \{v, w\} \in X \cap E_u$ and (1) holds, then remove e and all edges and inner vertices from P , add a new unsafe edge e' and let $c(e') := \min\{M, c(e)\}$ and $\theta(e') := \min\{\theta(e), \text{len}(P)\}$.

Reduction Rule 6 is correct since when one unsafe connection weakly dominates the other (neither longer nor more costly), the dominated one is irrelevant for both shortest paths and upgrade costs. The next rule treats two incomparable unsafe paths with the same endpoints. In this case, we have to keep both, but can compress the unsafe corridor into a path with two edges that preserves the length of the corridor and the cost to upgrade all of its unsafe edges.

► **Reduction Rule 7** (Unsafe Corridor Compression). Let P be a compressible unsafe corridor with endpoints v, w . Let $M := \sum_{e \in E(P) \cap E_u} c(e)$ be the total cost of P . Then remove all edges and inner vertices from P , add a new vertex $x_{v,w}$ and the edges $e' = \{v, x_{v,w}\}$ to E_u and $e'' = \{x_{v,w}, w\}$ to E_s . Set $\theta(e') = \text{len}(P) - 1$, $c(e') = M$, and $\theta(e'') = 1$.

► **Reduction Rule 8** (Corridor). Let P be a corridor. If P is safe, then apply Reduction Rule 4 if it is contractible and Reduction Rule 5 if it is compressible. If P is unsafe, then apply Reduction Rule 6 if it is contractible and Reduction Rule 7 if it is compressible.

Each corridor replacement preserves the relevant v - w connections. Safe and unsafe corridors represent length- $\text{len}(P)$ connections that are already safe and that can be made safe at upgrade cost M , respectively. If the endpoints are joined by a feedback edge e , then e contributes a length- $\theta(e)$ connection that is either already safe or that can be made safe at upgrade cost $c(e)$. The contraction rules delete dominated connections, while the compression rules keep two incomparable connections by representing the corridor as a two-edge path. Since every s - t path for terminal pair $\{s, t\}$ uses either the whole corridor or none of it, these replacements preserve feasibility and optimum cost.

Proof of Theorem 11. Let X be a minimum feedback edge set of G with $|X| = \mathbf{fes}$ and let T denote the forest obtained from $G - X$ when Reduction Rule 3 is applied exhaustively. We mark all important vertices and keep these marks fixed throughout the reduction. Now, apply Reduction Rule 8 exhaustively. Whenever a rule deletes an edge of X , we remove

it from the current feedback edge set. New edges replacing subpaths of T are treated as forest edges. Let T' denote the forest obtained from T by the above replacements, $X' \subseteq X$ the remaining feedback edges (recall that applying corridor contractions can delete feedback edges as well), and let G' denote the graph T' together with the edges from X' . Let $V(T') = V_1 \cup V_2^+ \cup V_2^- \cup V_{\geq 3}$, where we denote by V_1 the set of leaves, by V_2^+ the set of important degree-2 vertices, by V_2^- the set of unimportant degree-2 vertices, and by $V_{\geq 3}$ the set of vertices of degree at least three. We know that $|V_1| + |V_2^+| \leq 2\text{fes} + 2p$, and it is well known that $|V_{\geq 3}| \leq |V_1|$. Next we show that $|V_2^-| \leq \text{fes}$. After the execution, every remaining unimportant degree-two vertex is the unique inner vertex of a corridor with three vertices whose endpoints are important and adjacent by a remaining feedback edge. Since T' is a forest, for each feedback edge $\{v, w\} \in X'$ there is at most one v - w path in T' . Hence, this defines an injective mapping from unimportant degree-two vertices to feedback edges in X' , and thus $|V_2^-| \leq |X'| \leq \text{fes}$. Hence, we get that $|V(G')| = |V(T')| = |V_1| + |V_2^+| + |V_2^-| + |V_{\geq 3}| \leq |V_1| + |V_2^+| + |V_2^-| + |V_1| \leq 5\text{fes} + 4p$. Finally, $|E(G')| = |E(T')| + |X'| \leq 6\text{fes} + 4p$. $\blacktriangleleft$

8 ILP, Preprocessing, and Cut Generation

For every $r \in [p]$, we denote by A_r^α the set of all α -admissible arcs, where, for an edge $e = \{v, w\}$ and terminal pair $\{s_r, t_r\}$, arc (v, w) is α -admissible for terminal pair $\{s_r, t_r\}$ if $\text{dist}(s_r, v) + \theta(e) + \text{dist}(w, t_r) \leq \alpha \cdot \text{dist}(s_r, t_r)$. For a vertex $v \in V$, we denote by $\text{out}_r^\alpha(v)$ and $\text{in}_r^\alpha(v)$ the sets of outgoing and incoming arcs at v in A_r^α , respectively. We say that an edge is α -admissible for terminal pair $\{s_r, t_r\}$ if at least one of its two associated arcs is α -admissible for terminal pair $\{s_r, t_r\}$. Let E_r^α denote the set of all α -admissible edges for terminal pair $\{s_r, t_r\}$. Finally, let $G_r^\alpha := G[E_r^\alpha]$ denote the α -admissible graph for terminal pair $\{s_r, t_r\}$. We use the following ILP as our baseline exact formulation.

$$\min \sum_{e \in E_u} c(e) \cdot x_e \quad (2a)$$

$$\text{s.t.} \quad \sum_{a \in \text{out}_r^\alpha(v)} f_a^r - \sum_{a \in \text{in}_r^\alpha(v)} f_a^r = \begin{cases} 1, & \text{if } v = s_r, \\ -1, & \text{if } v = t_r, \\ 0, & \text{otherwise,} \end{cases} \quad \forall r \in [p], \forall v \in V, \quad (2b)$$

$$\sum_{a \in A_r^\alpha \cap \{(v, w), (w, v)\}} f_a^r \leq x_e, \quad \forall r \in [p], \forall \{v, w\} \in E_u, \quad (2c)$$

$$\sum_{a \in A_r^\alpha} \theta(a) \cdot f_a^r \leq \alpha \cdot \text{dist}_G(s_r, t_r), \quad \forall r \in [p], \quad (2d)$$

$$0 \leq f_a^r \leq 1, \quad \forall r \in [p], \forall a \in A_r^\alpha. \quad (2e)$$

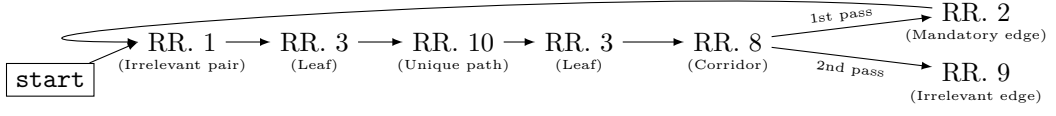
$$x_e \in \{0, 1\}, \quad \forall e \in E_u, \quad (2f)$$

Constraints (2b)–(2e) ensure that for each terminal pair $\{s_r, t_r\}$, a safe s_r - t_r path exists, that can use upgraded unsafe edges (2c), but must have detour factor at most α (2d).

8.1 FES-based Preprocessing

We apply the reduction rules that we described before together with the following two, where the second is a direct combination of Reduction Rules 1 and 2:

► **Reduction Rule 9 (Irrelevant edge).** If an edge is not α -admissible for any terminal pair, delete it.



■ **Figure 3** Flow chart of our shrink heuristic.

► **Reduction Rule 10 (Unique path).** If a terminal pair is not connected by a safe path of detour factor at most α but connected by a unique path of detour factor at most α containing at least one unsafe edge, then let C denote the sum of costs of all unsafe edges on this path. If C exceeds the budget, return a trivial *no*-instance, otherwise make all unsafe edges on the path safe, decrease the budget by C , and delete the terminal pair.

We apply each reduction rule exhaustively before moving to the next one. Since one reduction rule can make another rule applicable, their ordering matters. Figure 3 shows how we arrange our reduction rules. Empirically, this two-round execution provides a good trade-off between running time and preprocessing effectiveness compared to an exhaustive application until every reduction rule becomes inapplicable. Before each of Reduction Rules 3 and 8, we recompute a minimum feedback edge set. Notably, after applying the shrink heuristic, the treewidth of the obtained graph is not larger than the treewidth of the input graph, since deletion and edge contraction and compression do not increase the treewidth.

8.2 TD-separator Cuts

Let $(\mathcal{T}, \{B_i\}_{i \in I})$ with tree $\mathcal{T} = (I, Y)$ be a tree decomposition of G . Our cut generation is based on the following, simple insight about cuts consisting only of unsafe edges.

► **Observation 12.** Let $\{s_r, t_r\} \in \mathcal{P}$ be a terminal pair and $U \subseteq V(G_r^\alpha)$ such that $|\{s_r, t_r\} \cap U| = 1$. If $\emptyset \neq \partial_{G_r^\alpha}(U) \subseteq E_u$, then $|F \cap \partial_{G_r^\alpha}(U)| \geq 1$ for every solution F .

Proof. Since $\partial_{G_r^\alpha}(U)$ forms an s_r - t_r cut in G_r^α , it intersects all s_r - t_r paths of detour factor at most α in G . Thus, every feasible s_r - t_r path must traverse an edge from $\partial_{G_r^\alpha}(U)$. Since $\partial_{G_r^\alpha}(U)$ contains only unsafe edges, every solution must upgrade at least one edge from $\partial_{G_r^\alpha}(U)$. ◀

It is well known that every edge $\{i, j\} \in Y$ of $\mathcal{T}$ corresponds to a separator $S_{\{i, j\}} := B_i \cap B_j$ in the following sense. Let $I_i^{\{i, j\}}$ and $I_j^{\{i, j\}}$ denote all nodes of the connected component of $\mathcal{T} - \{\{i, j\}\}$ containing i and j , respectively. Let $B_x^{\{i, j\}} = \bigcup_{y \in I_x^{\{i, j\}}} B_y$ denote the corresponding set of vertices in G for each $x \in \{i, j\}$. Then $B_i^{\{i, j\}} \setminus S_{\{i, j\}}$ is disconnected from $B_j^{\{i, j\}} \setminus S_{\{i, j\}}$ in $G - S_{\{i, j\}}$. Now assume that $\mathcal{T}$ is rooted at some node. For an edge $\{i, j\} \in Y$, let $\min_{\mathcal{T}}(\{i, j\})$ denote the endpoint of larger depth. Every $X \subseteq S_{\{i, j\}}$ defines a *candidate side* $V_{\{i, j\}}^X := \left(B_{\min_{\mathcal{T}}(\{i, j\})}^{\{i, j\}} \setminus S_{\{i, j\}} \right) \cup X$. Altogether, we consider the set $\mathcal{V} = \{V_{\{i, j\}}^X \mid \{i, j\} \in Y, X \subseteq S_{\{i, j\}}\}$ of all candidate sides.

For each terminal pair $\{s_r, t_r\}$ and candidate side $V_{\{i, j\}}^X \in \mathcal{V}$, let $V_{\{i, j\}, r}^X := V_{\{i, j\}}^X \cap V(G_r^\alpha)$ be the candidate side when restricted to the vertices of the α -admissible graph for $\{s_r, t_r\}$. If $|\{s_r, t_r\} \cap V_{\{i, j\}, r}^X| = 1$ and $\emptyset \neq \partial_{G_r^\alpha}(V_{\{i, j\}, r}^X) \subseteq E_u$, then, due to Observation 12, every feasible solution upgrades at least one edge of this cut. Hence, for this terminal pair and candidate side, we add the valid inequality $\sum_{e \in \partial_{G_r^\alpha}(V_{\{i, j\}, r}^X)} x_e \geq 1$ to the ILP.

■ **Table 1** Structural statistics for our village and region networks, reported as means with median, minimum (lower), and maximum (upper) in parentheses. $\overline{tw}$ denotes the width of a TD computed via minimum fill-in.

Type	Vertices	Edges m	fes/m	$\overline{tw}$	Δ	Planar
Village	355.43 [341 ₆₁ ⁶⁵⁵]	443.67 [412.50 ₇₆ ⁸⁷¹]	0.2 [0.20 _{0.11} ^{0.25}]	6.00 [5 ₃ ¹¹]	4.17 [4 ₄ ⁵]	27/30
Region	738.63 [604.50 ₁₄₀ ¹⁹²⁹]	934.97 [755 ₁₇₆ ²⁶²³]	0.2 [0.20 _{0.14} ^{0.28}]	7.97 [7 ₄ ¹⁴]	4.43 [4 ₄ ⁵]	20/30

9 Experiments

Our experiments ran on Intel® Xeon® Silver 4310 CPU at 2.10GHz (12 cores), 125 GB RAM, Ubuntu 24.04.4 LTS (x86_64), using Gurobi(py) 13.0 for the ILPs.

9.1 Instances

We considered 30 village networks and 30 region networks. For each network, we combined three safety models, three terminal-pair fractions, two random samples, and three detour factors. In total, we constructed 1620 village instances and 1620 region instances.

Networks. We select municipalities based on the *ADFC Bicycle Climate Test* [1], a nationwide online cyclist satisfaction survey in which participants rate the perceived bicycle-friendliness of their municipality. From the ranking for towns of at most 20000 inhabitants, we took the subranking when filtered for towns of at most 10000 inhabitants. From this, we selected the top ten, bottom ten, and ten randomly sampled from the remaining towns. From OSM, we extracted the road network using type “drive”, made it undirected, kept only the largest connected component, and assigned the length in accordance with the OSM data. Herein, when two vertices are connected by two arcs, we only represent them separately (via paths with two edges) if they differ significantly in length or type. We set the cost of an unsafe edge to its length, assuming the cost to scale with length, thereby neglecting the type of the street segment and any further details. This abstracts from realistic costs, for which we did not find reliable comparable data. Analogously, for each of the selected towns, we computed the network corresponding to the 3 km radius region around the associated town’s center, mimicking inter-village networks. Table 1 gives an overview of selected statistics relevant to us for the derived networks. Remarkably, on average, 20% of the edges suffice as a feedback edge set, and the maximum observed treewidth upper bound is 11 for villages and 14 for regions. We also point out that only two thirds of our region networks are planar.

Safety models. For classifying edges as unsafe, we developed three models A, B, and C that are nested in spirit. Model A is an accommodating local-access model where local roads are generally considered bicycle-suitable. Model B is a balanced speed-aware model where local roads require low-speed evidence. Finally, model C is a conservative low-stress model where only clearly low-stress roads are considered safe. See Table 2 for further details.

Terminal pairs and detour-factors. For each network, we randomly sampled $x \cdot n$ many (distinct) terminal pairs for $x \in \{0.25, 0.5, 0.75\}$ for villages and for $x \in \{0.15, 0.3, 0.45\}$ for regions, where n denotes the number of vertices. For each network and such setup, we constructed two instances. Finally, for each of those, we chose detour factors $\alpha \in \{1.2, 1.3, 1.5\}$, where $\alpha = 1.2$ [20, p. 10] and $\alpha = 1.3$ [6] are used in bicycle-network design as target bounds.

■ **Table 2** Summary of the three edge safety models. The first row gives rules that make an edge safe in all models; subsequent rows list additional rules specific to road classes. Last two rows report average unsafe-edge fractions, with median, minimum, and maximum in parentheses, for villages (top row) and regions (bottom row).

	Model A	Model B	Model C
Safe in all models	Living streets, traffic-calmed roads, and extracted road segments with explicit bicycle infrastructure		
Residential roads	Safe	Safe if speed is missing or at most 30 km/h	Safe if speed is missing or at most 30 km/h
Service / unclassified roads			Safe only if speed is at most 20 km/h
Tertiary roads	Safe only if speed is at most 30 km/h		Unsafe by default
Paths / pedestrian streets, if present	Safe unless bicycles are explicitly forbidden		Safe only if bicycles are explicitly allowed
Average unsafe fraction	0.22 [0.21 ^{0.32} _{0.10}]	0.29 [0.28 ^{0.51} _{0.12}]	0.37 [0.36 ^{0.58} _{0.15}]
	0.22 [0.22 ^{0.30} _{0.14}]	0.30 [0.28 ^{0.56} _{0.15}]	0.36 [0.35 ^{0.57} _{0.15}]

9.2 Algorithmic Setup

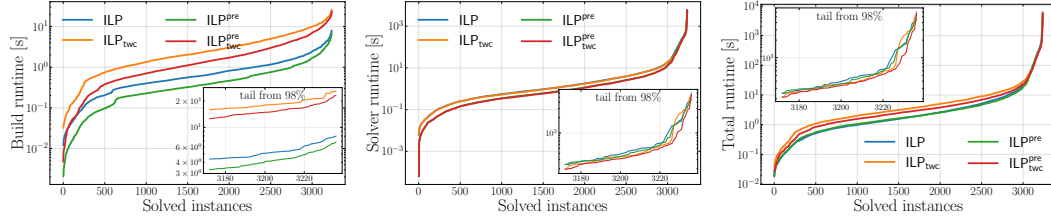
We compare ILP (cf. (2)) against ILP_{twc} , which is ILP with TD-separator cuts, ILP^{pre} , which is ILP where preprocessing is applied, and $\text{ILP}_{\text{twc}}^{\text{pre}}$, the combination of both. To compute a tree decomposition, we use NetworkX’s implementation `treewidth_min_fill_in` [25] of the minimum fill-in heuristic [3, 39]. On a high level, the heuristic greedily turns neighborhoods into cliques, thereby creating for the resulting graph a so-called perfect elimination ordering, which yields a tree decomposition. The number of candidate sides for each TD-separator can be as large as $2^{\overline{\text{tw}}+1}$. Each such candidate can be considered for every terminal pair, leading to many candidate cuts. Thus, our implementation limits the process of generating TD-separator cuts as part of the ILP building time in the following three ways.

1. We skip TD-separators of size larger than s^* , since enumerating all 2^{s^*} candidate sets can be expensive. We set $s^* = 14$; recall that the maximum $\overline{\text{tw}}$ in our experiments is at most 14, implying that the limit applies only to boundary cases in our experiments.
2. We only consider the best N^* candidate sides for cut generation according to the following ranking. Let $U \in \mathcal{V}$ be a candidate side with $\partial_G(U) \neq \emptyset$. Define the (negative) fraction $\rho_G(U) = -|\partial_G(U) \cap E_u|/|\partial_G(U)|$ of unsafe edges in the cut. The score of U in G is defined as $\text{score}(U) = (|\partial_G(U)|, \rho_G(U), |U|)$. Each dimension of the score is to minimize, and scores are compared lexicographically. That is, firstly we seek small cuts, secondly high fractions of unsafe edges, and thirdly small sides. Ties are broken arbitrarily (by their time of appearance) for any two candidate sides with the same score. We set $N^* = 5000$.
3. For each terminal pair $\{s_i, t_i\}$, we only consider cuts with at most m^* edges. From these, we only add at most c^* cuts to the ILP. We set $m^* = 10$ and $c^* = 20$.

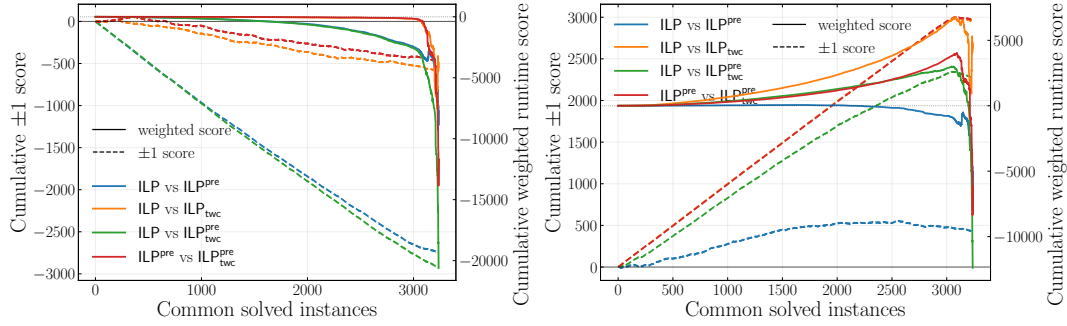
These limits affect only *which* valid cuts are generated; every added inequality is valid.

9.3 Results

Solver performance. Overall, from our experiments, we can conclude the following regarding the solver performance in terms of runtimes; Figure 4 and Figure 5 accompany our conclusions.



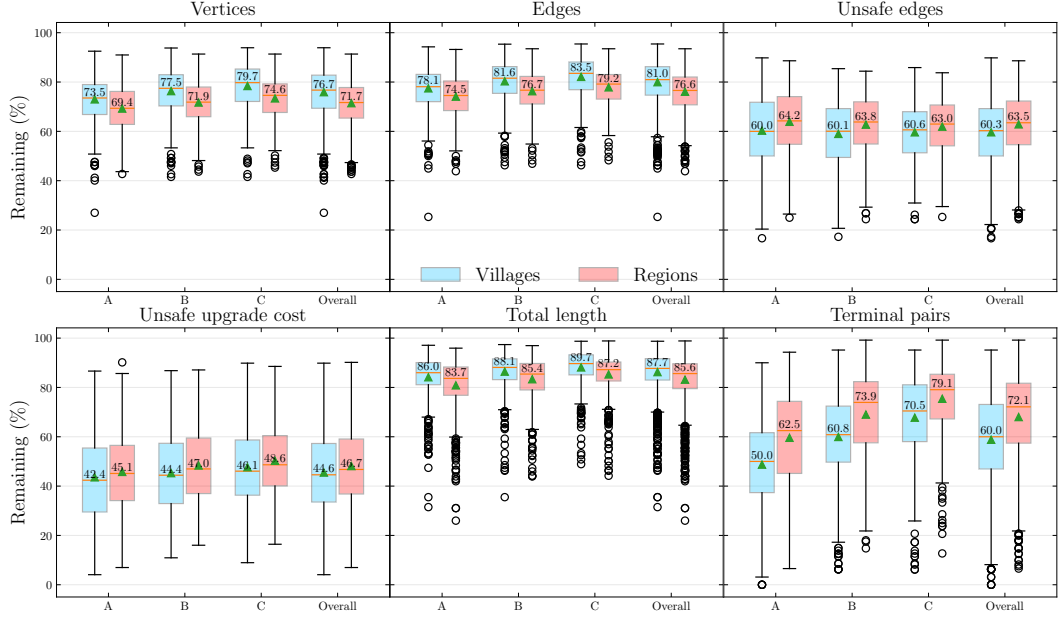
■ **Figure 4** Cactus plots for all instances solved by each of the four solvers regarding building time (left), solver runtime (middle), and total runtime (right). Each inset zooms into the tail of the corresponding cactus plot, i.e., the slowest 2% of the common solved instances.



■ **Figure 5** ± 1 - and weighted scores of solver pairs S_1 versus S_2 . The weighted score is $s(S_1, S_2) = \text{time}(S_2) - \text{time}(S_1)$, where time is either the solver runtime (left) or the total runtime (right). The ± 1 -score is +1 if $s(S_1, S_2) > 0$, and -1 if $s(S_1, S_2) < 0$. So, a line going upwards corresponds to S_1 is faster than S_2 , and downwards to S_1 is slower than S_2 . For each solver pair, the commonly solved instances are sorted separately.

1. Preprocessing significantly decreases the solver runtime; see Figure 4 (middle) and Figure 5. Due to the short runtimes of the preprocessing, both solvers using preprocessing ILP^{pre} and ILP^{pre}_{twc} score best versus ILP; see Figure 5.
2. The runtime for building the ILP with the TD-separator cuts but without preprocessing is 3.53 s on average and never exceeded 25.64 s; see Figure 4 (left). This indicates that the heuristic makes use of small treewidth, small degree, and sparsity. Preprocessing reduced the building times for both with and without TD-separator cut generation, to roughly 65% for the average time and 90% of the maximum observed time.
3. For instances solved within roughly 100 s, TD-separator cuts do not appear to pay off. For harder instances, however, ILP^{pre}_{twc} dominates; see Figure 4 (right) and Figure 5.
4. While ILP_{twc} performs worse than ILP in total, its solver runtime is mostly shorter; see Figure 5 (left). While preprocessing also decreases the solver runtime, finally ILP^{pre}_{twc} scores best against all other solvers, even against ILP^{pre} when it comes to solver runtime.

Effect of the preprocessing. Our preprocessing routine (cf. Section 8.1) is fast and effective. Over all of our instances, the routine takes 1.15 seconds on average (median 0.67 second), where the maximum observed time is 12.46 seconds. Yet, the routine significantly reduces the number of unsafe edges and of terminal pairs to 63% and 73% on average, respectively. See Figure 6 for a detailed overview. For our three different safety models A, B, and C, we find only a correlation regarding the reduction of terminal pairs. Comparing safety models A and C, the reduction changes from 50% to 70% for villages, and 62% to 79% for regions.



■ **Figure 6** Statistics for the preprocessing routine.

■ **Table 3** Overview of the average relative cost reduction when increasing the detour factor α , with median, minimum (bottom), and maximum (top) in parentheses.

	1.2 $\rightarrow$ 1.3	1.3 $\rightarrow$ 1.5	1.2 $\rightarrow$ 1.5
Cost reduction (in %)	7.87 [6.59 ^{37.86} _{0.00}]	10.76 [8.97 ^{39.21} _{0.00}]	17.63 [15.42 ^{57.29} _{0.00}]

Cost of detours. Table 3 gives an overview of the costs when the detour factor is increased. Recall that we set the cost of an unsafe edge to its length; hence, our results display the savings of the total upgraded length. We highlight that when moving from $\alpha = 1.2$ to $\alpha = 1.3$, on average 7.87% of the costs are reduced, ranging from no reduction even up to 37.86% in the maximum case. For decision makers, we can often assume that the upper bound on the detour factor lies within a reasonable interval, say $\alpha \in [1.2, 1.3]$. Our results hence indicate that it is worth computing solutions for several upper bounds on the detour factor in the given interval and compare the trade-offs between detour length and budget.

10 Discussion

On the experimental-modeling side, we randomly sample terminal pairs, whereas real-world demand is likely more structured around hotspots such as schools, supermarkets, or public-transport stops. Similarly, we set the upgrade cost of each street segment to its length. Besides length, however, other factors such as road type or local regulations may affect the construction costs. Incorporating such data would make the experiments more realistic.

On the algorithmic side, our experiments show that preprocessing substantially reduces solver runtimes, making further and improved reduction rules a promising direction. For TD-separator cuts, their effectiveness may depend strongly on the generated candidate sides. Larger heuristic parameters increase model-building time, and adding too many cuts can even slow down the ILP solver (see Appendix A.2 for details). A better understanding of how to select strong cuts and which candidate sides yield them is an important future direction.

On the theoretical side, it remains open which structural parameters beyond m_a and $fes + p$ lead to fixed-parameter tractability, in particular whether $fvs + p$ or the feedback edge number alone is sufficient.

References

- 1 Allgemeiner Deutscher Fahrrad-Club e.V. (ADFC). ADFC-Fahrradklima-Test 2020: Städteranking. https://www.adfc.de/fileadmin/user_upload/Im-Alltag/Fahrradklimatest/FKT_2020/ADFC-Fahrradklima-Test_2020_Staedteranking.pdf, 2021. Accessed: 2026-06-17.
- 2 Carolien Beckx, Steven Broekx, Bart Degraeuwe, Bart Beusen, and Luc Int Panis. Limits to active transport substitution of short car trips. *Transportation Research Part D: Transport and Environment*, 22:10–13, 2013. doi:10.1016/j.trd.2013.03.001.
- 3 Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3):259–275, 2010. doi:10.1016/j.ic.2009.03.008.
- 4 Geoff Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017. doi:10.1016/j.compenvurbsys.2017.05.004.
- 5 Geoff Boeing. A multi-scale analysis of 27,000 urban street networks: Every US city, town, urbanized area, and zillow neighborhood. *Environment and Planning B: Urban Analytics and City Science*, 47(4):590–608, 2020. doi:10.1177/2399808318784595.
- 6 Yutong Cai, Ghim Ping Ong, and Qiang Meng. Sidewalk-based bicycle path network design incorporating equity in cycling time. *Computer-Aided Civil and Infrastructure Engineering*, 39(20):3063–3082, 2024. doi:10.1111/mice.13240.
- 7 Alessio Cardillo, Salvatore Scellato, Vito Latora, and Sergio Porta. Structural properties of planar graphs of urban street patterns. *Physical Review E*, 73(6):066107, 2006. doi:10.1103/PhysRevE.73.066107.
- 8 Eden Chlamtác, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating spanners and directed steiner forest: Upper and lower bounds. *ACM Transactions on Algorithms*, 16(3):33:1–33:31, 2020. doi:10.1145/3381451.
- 9 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. doi:10.1145/2925416.
- 10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 11 Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On pairwise spanners. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 209–220. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.STACS.2013.209.
- 12 Claus Doll, Clemens Brauer, and Dorien Duffner-Korbee. The potential of cycling for climate protection and livable urban centers and regions: New methods for forecasting supply and demand in germany as a cycling nation up to 2035. Technical report, Fraunhofer Institute for Systems and Innovation Research ISI, Karlsruhe, Germany, May 2024. Prepared on behalf of the Allgemeiner Deutscher Fahrrad-Club (ADFC). URL: https://www.adfc.de/fileadmin/user_upload/Expertenbereich/Politik_und_Verwaltung/Studie_Potenzial_des_Radverkehrs_in_Deutschland/Cycling_Potentials_Germany_Summary_Report.pdf.
- 13 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014. doi:10.1145/2650261.
- 14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, London, 2013. doi:10.1007/978-1-4471-5559-1.

- 15 Jennifer Duthie and Avinash Unnikrishnan. Optimization framework for bicycle network design. *Journal of Transportation Engineering*, 140(7):04014028, 2014. doi:10.1061/(ASCE)TE.1943-5436.0000690.
- 16 David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 16:1–16:10. Association for Computing Machinery, 2008. doi:10.1145/1463434.1463455.
- 17 European Conference of Ministers of Transport. *Public Transport in Rural Areas*. OECD Publishing, 1984. doi:10.1787/9789282105252-en.
- 18 Andreas Emil Feldmann and Michael Lampis. Parameterized algorithms for steiner forest in bounded width graphs. *ACM Transactions on Algorithms*, 2025. doi:10.1145/3748724.
- 19 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.
- 20 Forschungsgesellschaft für Straßen- und Verkehrswesen. *Empfehlungen für Radverkehrsanlagen: ERA*. FGSV 284. FGSV Verlag, Köln, Ausgabe 2010 edition, 2010. See p. 10 for network-level detour-factor recommendations. URL: <https://www.fgsv-verlag.de/era>.
- 21 Peter G. Furth, Maaza C. Mekuria, and Hilary Nixon. Network connectivity for low-stress bicycling. *Transportation Research Record: Journal of the Transportation Research Board*, 2587(1):41–49, 2016. doi:10.3141/2587-06.
- 22 Elisabeth Gassner. The steiner forest problem revisited. *Journal of Discrete Algorithms*, 8(2):154–163, 2010. Selected papers from the 3rd Algorithms and Complexity in Durham Workshop ACiD 2007. doi:10.1016/j.jda.2009.05.002.
- 23 Hannah Gorges and Giuliano Mingardo. The potential of active modes to reduce short car trips. a data-driven approach. *Transport Policy*, 168:1–14, 2025. doi:10.1016/j.tranpol.2025.03.015.
- 24 Elena Grigorescu, Nithin Kumar, and Young-San Lin. Approximation algorithms for directed weighted spanners. *Algorithmica*, 88(3):38, 2026. doi:10.1007/s00453-026-01384-6.
- 25 Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA, 2008.
- 26 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 27 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
- 28 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-3-540-68279-0_8.
- 29 Leon Kellerhals and Tomohiro Koana. Parameterized complexity of geodetic set. *Journal of Graph Algorithms and Applications*, 26(4):401–419, 2022. doi:10.7155/jgaa.00601.
- 30 Yusuke Kobayashi. NP-hardness and fixed-parameter tractability of the minimum spanner problem. *Theoretical Computer Science*, 746:88–97, 2018. doi:10.1016/j.tcs.2018.06.031.
- 31 Yusuke Kobayashi. An FPT algorithm for minimum additive spanner problem. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *LIPICs*, pages 11:1–11:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.STACS.2020.11.
- 32 Jisoon Lim, Kevin Dalmeijer, Subhrajit Guhathakurta, and Pascal Van Hentenryck. The bicycle network improvement problem. *Journal of Transportation Engineering, Part A: Systems*, 148(11):04022095, 2022. doi:10.1061/JTEPBS.0000742.
- 33 Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In *22nd International Conference on Database Theory, ICDT 2019, Lisbon, Portugal, March 26-28, 2019*, *LIPICs*, pages 12:1–12:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICDT.2019.12.

- 34 Antonio Mauttone, Gonzalo Mercadante, María Rabaza, and Fernanda Toledo. Bicycle network design: Model and solution algorithm. *Transportation Research Procedia*, 27:969–976, 2017. doi:10.1016/j.trpro.2017.12.119.
- 35 Natalie Mueller, David Rojas-Rueda, Tom Cole-Hunter, Audrey de Nazelle, Evi Dons, Regine Gerike, Thomas Götschi, Luc Int Panis, Sonja Kahlmeier, and Mark Nieuwenhuijsen. Health impact assessment of active transportation: A systematic review. *Preventive Medicine*, 76:103–114, 2015. doi:10.1016/j.ypmed.2015.04.010.
- 36 Luis Guillermo Natera Orozco, Federico Battiston, Gerardo Iñiguez, and Michael Szell. Data-driven strategies for optimal bicycle network growth. *Royal Society Open Science*, 7(12):201130, 2020. doi:10.1098/rsos.201130.
- 37 Pekka Oja, Sylvia Titze, Adrian Bauman, Bas de Geus, Patrick Krenn, Bill Reger-Nash, and Thomas Kohlberger. Health benefits of cycling: A systematic review. *Scandinavian Journal of Medicine & Science in Sports*, 21(4):496–509, August 2011. Epub 2011 Apr 18. doi:10.1111/j.1600-0838.2011.01299.x.
- 38 Helen Poltimäe, Merlin Rehema, Janika Raun, and Age Poom. In search of sustainable and inclusive mobility solutions for rural areas. *European Transport Research Review*, 14(1):13, 2022. doi:10.1186/s12544-022-00536-3.
- 39 Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi:10.1137/0205021.
- 40 Jessica E. Schoner and David M. Levinson. The missing link: Bicycle infrastructure networks and ridership in 74 US cities. *Transportation*, 41(6):1187–1204, 2014. doi:10.1007/s11116-014-9538-1.
- 41 Kirill Simonov, Farehe Soheil, and Shaily Verma. Finding minimum distance preservers: A parameterized study. *arXiv preprint arXiv:2603.21442*, 2026. doi:10.48550/arXiv.2603.21442.
- 42 SINUS-Institut und Bundesministerium für Verkehr. Fahrrad-Monitor Deutschland 2025 – Erwachsenen- und Jugendbefragung (14–69 Jahre). Technical report, Bundesministerium für Verkehr, 2026. URL: <https://www.bmv.de/SharedDocs/DE/Anlage/StV/fahrrad-monitor-deutschland-2025-langfassung-erwachsenen-jugendbefragung.pdf>.
- 43 Christoph Steinacker, David-Maximilian Storch, Marc Timme, and Malte Schröder. Demand-driven design of bicycle infrastructure networks for improved urban bikeability. *Nature Computational Science*, 2:655–664, 2022. doi:10.1038/s43588-022-00318-w.
- 44 Michael Szell, Sayat Mimar, Tyler Perlman, Gourab Ghoshal, and Roberta Sinatra. Growing urban bicycle networks. *Scientific Reports*, 12:6765, 2022. doi:10.1038/s41598-022-10783-y.
- 45 Christoph van Dülmen, Martin Šimon, and Andreas Klärner. Transport poverty meets car dependency: A gps tracking study of socially disadvantaged groups in european rural peripheries. *Journal of Transport Geography*, 101:103351, 2022. doi:10.1016/j.jtrangeo.2022.103351.
- 46 Ane Rahbek Vierø and Michael Szell. Network analysis of the danish bicycle infrastructure: Bikeability across urban–rural divides. *Geographical Analysis*, 57(4):616–640, May 2025. doi:10.1111/gean.70012.

A Appendix

A.1 Deferred Proofs

Proof of Proposition 4. Let I be an instance of SBNBD with G being a tree, budget k , and let κ be the sum of costs of all unsafe edges. Let I' be the instance obtained from I by setting the budget to κ . Apply Reduction Rules 1 and 2 exhaustively to I' . Since each terminal pair has a unique path, each unsafe edge on this path is mandatory and is made safe by Reduction Rule 2. Hence, every edge made safe by Reduction Rule 2 is contained in every feasible

solution. Consequently, once all unsafe edges on the unique path of a terminal pair have been made safe, this terminal pair is deleted by Reduction Rule 1. It follows that no terminal pair is left. Let F^* denote the set of all unsafe edges made safe by Reduction Rule 2 during the exhaustive application, and let κ' denote the remaining budget in the final instance. Note that $c(F^*) = \kappa - \kappa'$. Since every edge in F^* is mandatory and upgrading F^* satisfies all terminal pairs, $c(F^*)$ is the minimum cost of any solution. Return *yes* if $c(F^*) \leq k$, and *no* otherwise. $\blacktriangleleft$

Proof of Observation 5. Let I be an instance of SBNBD with detour factor α where G has maximum degree at most two. Then, graph G decomposes into paths and cycles. If a terminal pair is not contained in the same connected component, then return *no*. Otherwise, solve each connected component independently. If a connected component is a path, compute a minimum-cost solution using Proposition 4. Now consider a connected component C that is a cycle. First, consider the solution that upgrades all unsafe edges of C . This is feasible for all terminal pairs contained in C . We compare this solution with possible solutions where at least one unsafe edge is not upgraded. Thus, for each unsafe edge e of C , consider the case where e remains unsafe and hence no feasible safe path can use e . Then, every terminal pair $\{s, t\}$ contained in C must be connected along the unique s - t path in $C - \{e\}$. This is feasible only if $\text{dist}_{C-\{e\}}(s, t) \leq \alpha \cdot \text{dist}_G(s, t)$ for every terminal pair $\{s, t\}$ contained in C . If this condition holds, we delete e and compute a minimum-cost solution on the path $C - \{e\}$ using Proposition 4. Among the solution upgrading all unsafe edges and all solutions obtained by deleting one unsafe edge, we keep one of minimum cost. This enumeration is exhaustive since every solution either upgrades all unsafe edges of C or leaves at least one unsafe edge e not upgraded. Finally, the optimal solution for I is composed of the optimal solutions for each of its connected components. If the total cost exceeds the budget, return *no*, otherwise return *yes*. $\blacktriangleleft$

Proof of Theorem 7. Let $I' = (G = (V, E, c, \theta), \mathcal{P}, k, \alpha)$ be the instance of SBNBD obtained in polynomial time from an instance $I = (X, \mathcal{S}, r)$ of HITTING SET via Construction 2. For every $S \in \mathcal{S}$, we have $\text{dist}_G(w_S, v^*) = 2$. Moreover, w_S is adjacent to v_x if and only if $x \in S$. Since $\alpha = 1$, for every $F \subseteq E_u$, any safe w_S - v^* path of detour factor 1 in $G\langle F \rangle$ must have length exactly 2. Hence, it contains exactly one inner vertex v_x with $x \in S$, and the edge $\{v_x, v^*\}$ must be made safe by F . We make the following convention for a set $Y \subseteq X$ and a set $F \subseteq E_u$: $x \in Y$ if and only if $\{v^*, v_x\} \in F$. Since every unsafe edge has cost one, the convention implies that $|Y| \leq r$ if and only if $c(F) \leq k$. The following equivalences hold. Set $S \in \mathcal{S}$ is hit by Y if and only if there exists $x \in S \cap Y$. By our convention, for every $x \in S$, $x \in Y$ if and only if $\{v^*, v_x\} \in F$. Moreover, $\{v^*, v_x\} \in F$ for some $x \in S$ if and only if there is a safe w_S - v^* path of detour factor 1 in $G\langle F \rangle$. It follows that every set $S \in \mathcal{S}$ is hit by Y if and only if every terminal pair admits a safe path of detour factor 1 in $G\langle F \rangle$. Together with the cost correspondence, this proves that I is a *yes*-instance if and only if I' is a *yes*-instance.

Finally, note that $m_u = |X|$. Hence, both the SETH-based lower bound and the kernelization lower bound follow. W[2]-hardness transfers since $k = r$. $\blacktriangleleft$

Proof of Theorem 8. Let $I' = (G = (V, E, c, \theta), \mathcal{P}, k, \alpha)$ be the instance of SBNBD obtained in polynomial time from an instance $I = (X = \{x_1, \dots, x_N\})$ of PARTITION with sum $2T$ via Construction 3. The shortest v_0 - v_N path is of length $2NT$ by using the unsafe branch when traversing from v_{i-1} to v_i via a_i for each $i \in [N]$. Hence, since $\alpha = 2$, every feasible path in a solution must be of length at most $4NT$. We show that I is a *yes*-instance if and only if I' is a *yes*-instance.

For the forward direction, let (X_1, X_2) be a solution to I . We claim that $F := \{\{a_i, v_i\} \mid x_i \in X_1\}$ is a solution to I' . We have that $c(F) = \sum_{x_i \in X_1} c(\{a_i, v_i\}) = \sum_{x_i \in X_1} x_i = T \leq k$. Moreover, consider the shortest safe v_0 - v_N path in $G\langle F \rangle$. For each $x_i \in X_1$, it uses the branch traversing from v_{i-1} to v_i via a_i , since the unsafe edge $\{a_i, v_i\}$ is upgraded by F ; this branch has length $2T$. For each $x_i \in X_2$, it uses the branch traversing from v_{i-1} to v_i via b_i , which has length $2T + 2Nx_i$. Altogether, the path has length $2NT + 2N \sum_{x_i \in X_2} x_i = 4NT$. Thus, F is a solution to I' .

For the backward direction, let F be a solution to I' . We claim that (X_1, X_2) , where $X_1 := \{x_i \mid \{a_i, v_i\} \in F\}$ and $X_2 := X \setminus X_1$, is a solution to I . Since F is a solution, the shortest safe v_0 - v_N path in $G\langle F \rangle$ is of length $2NT + 2N \sum_{x_i \in X_2} x_i \leq 4NT$. It follows that $\sum_{x_i \in X_2} x_i \leq T$. Moreover, we know that $\sum_{x_i \in X_1} x_i = \sum_{\{a_i, v_i\} \in F} c(\{a_i, v_i\}) = c(F) \leq k = T$. Finally, since $\sum_{x_i \in X_2} x_i + \sum_{x_i \in X_1} x_i = 2T$, (X_1, X_2) is a solution to I . ◀

Proof of Theorem 10. Let $I' = (G' = (V', E', c, \theta), \mathcal{P}, k, \alpha)$ be the instance of SBNBD obtained in polynomial time from an instance $I = (G = (V = V_1 \uplus \dots \uplus V_r, E))$ of MULTI-COLORED CLIQUE via Construction 4. We show that I is a *yes*-instance if and only if I' is a *yes*-instance.

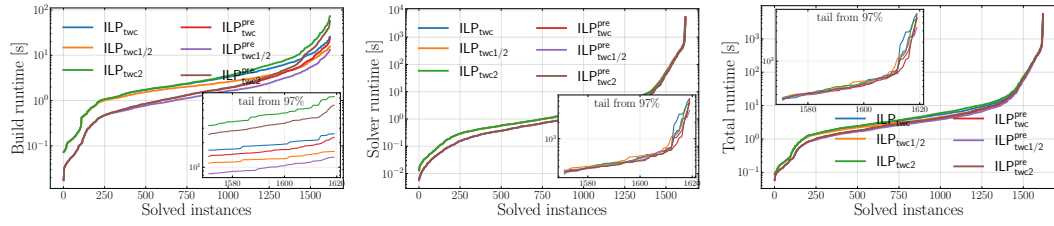
For the forward direction, let $W = \{v_1^*, \dots, v_r^*\}$ be a solution to I with $v_i^* \in V_i$ for each $i \in [r]$. We claim that $F := \{\{w_i, v_i^*\} \mid i \in [r]\} \cup \{\{v_i^*, v_j^*\} \mid i, j \in [r], i < j\}$ is a solution to I' . We have that $c(F) = \sum_{i \in [r]} c(\{w_i, v_i^*\}) + \sum_{i, j \in [r], i < j} c(\{v_i^*, v_j^*\}) = r \cdot r^3 + \binom{r}{2} = k$. Moreover, for each $i, j \in [r]$, $i < j$, there is a safe w_i - w_j path in $G'\langle F \rangle$ from w_i to v_i^* to v_j^* to w_j of length $3 = \text{dist}_{G'}(w_i, w_j)$. Hence, F is a solution to I' .

For the backward direction, let F be a solution to I' . By construction, for each color class $i \in [r]$, V_i separates w_i in G' from the rest of the graph. Since w_i appears in at least one terminal pair, at least one edge incident with w_i must be upgraded by F at cost r^3 . As there are r color classes, at least $r \cdot r^3$ budget is spent on these upgrades. Thus, the remaining budget is at most $\binom{r}{2} < r^3$, and hence no further edge incident with any w_i can be upgraded. Hence, for every $i \in [r]$, there is exactly one upgraded edge incident with w_i in F . Let this edge be $\{w_i, v_i^*\}$ with $v_i^* \in V_i$. We claim that $W := \{v_1^*, \dots, v_r^*\}$ is a solution to I . Consider distinct $i, j \in [r]$. By the assumption on the input instance, at least one vertex from V_i neighbors a vertex from V_j , and hence $\text{dist}_{G'}(w_i, w_j) = 3$. Since $\alpha = 1$, every feasible safe w_i - w_j path in $G'\langle F \rangle$ must have exactly three edges. Since $\{w_i, v_i^*\}$ is the only upgraded edge incident with w_i and $\{w_j, v_j^*\}$ is the only upgraded edge incident with w_j , the middle edge of such a path must be $\{v_i^*, v_j^*\}$. Thus, $\{v_i^*, v_j^*\} \in E$. Since this holds for every pair of distinct color classes, it follows that W induces a clique. Hence, W is a solution to I .

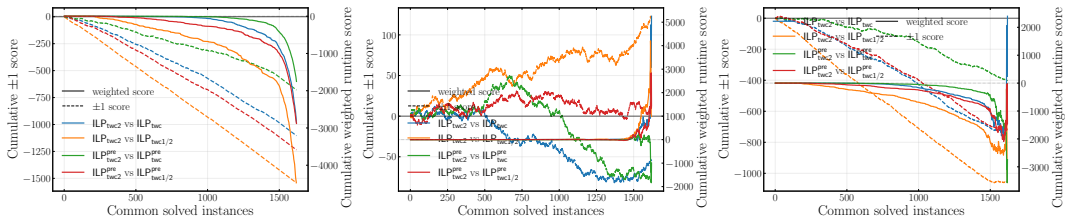
Since $|\mathcal{P}| = \binom{r}{2}$ and $k = r \cdot r^3 + \binom{r}{2}$ depend only on r , W[1]-hardness for the combined parameter $p + k$ follows. ◀

A.2 Further Experimental Results

Recall that for parameters (s^*, N^*, m^*, c^*) of the TD-separator cut heuristic, we selected the setup $\chi_1 = (14, 5000, 10, 20)$ (cf. Section 9.2). We also compared this setup with two other setups $\chi_{1/2} = \frac{1}{2} \cdot \chi_1$ and $\chi_2 = 2 \cdot \chi_1$, that is, when halving and doubling the parameters. Denote the corresponding ILP-based algorithms by $\text{ILP}_{\text{twc1/2}}$ and ILP_{twc2} without preprocessing, and by $\text{ILP}_{\text{twc1/2}}^{\text{pre}}$ and $\text{ILP}_{\text{twc2}}^{\text{pre}}$ when preprocessing is applied. Figure 7 and Figure 8 show their performance and scores on region instances, respectively. Larger parameters lead to larger model-building times, but do not improve performance overall. Concretely, ILP_{twc2} has the largest average model-building time (almost twice the average model-building time of $\text{ILP}_{\text{twc1/2}}$), but the smallest average and median solver runtime (marginally, e.g. regarding the



■ **Figure 7** Cactus plots for all region instances solved by each of the three solvers based on different parameters with respect to the TD-separator cut heuristic with and without preprocessing. Presented are building time (left), solver runtime (middle), and total runtime (right). Each inset zooms into the tail of the corresponding cactus plot, i.e., the slowest 3% of the common solved region instances.



■ **Figure 8** Scores (cf. Figure 5) of all TD-separator cut heuristics for all commonly solved region instances, where the largest setup competes against both the lower two setups, with and without preprocessing. Presented are building time (left), solver runtime (middle), and total runtime (right).

average solver runtime, 28.99 s compared to 29.60 s for $\text{ILP}_{\text{twc}1/2}$) among the commonly solved instances. In particular, larger candidate sets interact nontrivially with preprocessing and can even degrade performance. Concretely, the average total runtime of $\text{ILP}_{\text{twc}}^{\text{pre}}$ is the smallest with 28.92 s, yet $\text{ILP}_{\text{twc}1/2}^{\text{pre}}$ achieves the smallest median (2.79 s compared to second place 3.07 s by $\text{ILP}_{\text{twc}}^{\text{pre}}$) and maximum (3373.82 s compared to second place 4780.89 s by $\text{ILP}_{\text{twc}2}^{\text{pre}}$) total runtime among the commonly solved instances.

Large Scale Middle Mile Network Design Through Efficient Local Search

Carlo S. Sartori ✉ 

Amazon Science and Technology, Luxembourg

Jian Shen ✉ 

Amazon Science and Technology, London, UK

Anastasia Kireeva ✉ 

ETH Zurich, Switzerland

Mattia Neroni ✉ 

Amazon Science and Technology, Barcelona, Spain

Philipp Loick ✉ 

Amazon Science and Technology, Luxembourg

Abstract

E-commerce competition has intensified in recent years, further raising the delivery speed and cost efficiency standards that companies must meet. The middle mile network that connects fulfillment centers to local distribution centers plays a decisive role in achieving these standards. In this paper, we study the Middle Mile Network Design Problem, which determines optimal network investments that balance the trade-off between fastest delivery speed and minimal operational costs. The problem requires computing truck schedules, shipment routes, and parcel flows through intermediate facilities subject to complex capacity constraints. Two key challenges make this problem computationally hard to solve for standard exact and heuristic approaches: (i) the use of complex black-box evaluations to predict operational outcomes, and (ii) problem instances involving over 200,000 commodities that must be solved within hours for operational planning. We propose a metaheuristic based on variable neighborhood search with specialized local search operators alongside lazy neighborhood evaluation that enable efficient exploration of the high-dimensional solution space. Our algorithm achieves up to 7.6% solution improvement over an incumbent heuristic while reducing runtime by up to 44%.

2012 ACM Subject Classification Applied computing → Transportation

Keywords and phrases Network design, Middle mile logistics, Black-box evaluation, Local search, Variable Neighborhood Search

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.15

Acknowledgements We would like to thank the three anonymous reviewers for their valuable feedback. Anastasia Kireeva has contributed to this paper during her internship at Amazon.

1 Introduction

Worldwide e-commerce transported 110 billion parcels in 2024 [19], generating six trillion dollars in sales - a figure projected to reach eight trillion by 2028 [7]. This explosive growth has further intensified pressure in an already highly competitive market, requiring carriers to simultaneously improve delivery speed and reduce operational costs to maintain market share and sustain growth.

Recent studies show customers prefer delivery promises of two days or less [8]. It is not surprising then that the average delivery times in the United States fell from 6.5 days in October 2020 to 4.4 days by July 2022, with Amazon leading in delivery speed with 1.9 days [18]. However, post-COVID consumers have grown equally attentive to shipping costs [11]. Even if reducing delivery promises by one day can increase sales by 13%, the



© Carlo S. Sartori, Jian Shen, Anastasia Kireeva, Mattia Neroni, and Philipp Loick;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 15; pp. 15:1–15:18

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

operational costs to achieve such speed improvements may be prohibitively expensive and defeat the purpose [10]. This raises the fundamental question for carriers of whether the investments are worth it, and more specifically, *which* investments are the right ones.

This paper addresses the core challenge in e-commerce of designing a middle mile transportation network that is both fast and cost-efficient. The middle mile network refers to moving parcels from fulfillment centers (FCs), where items are stored, to delivery centers (DCs), where last mile delivery begins to the final customer. The speed of the middle mile network directly impacts both customer delivery promises and operational costs. A theoretical network connecting all FCs directly to all DCs could maximize speed, but prove prohibitively expensive to operate. Instead, companies consolidate shipments through intermediate sort centers (SCs) to reduce costs, while maintaining some direct connections for speed.

Designing such a network requires determining truck departures between sites, which legs of the network are available for shipment and how parcels travel through intermediate SCs. All of this while respecting the resource limits of the network such as site’s physical and labor capacities. We call this the *Middle Mile Network Design Problem* (MMNDP), which has been first studied by [4]. The MMNDP seeks to maximize speed while minimizing cost. We focus on Amazon’s context, where a black-box evaluation function maps a given network design to a speed metric that can be translated into expected financial gain, enabling direct tradeoff against the operational cost of achieving that speed [4]. Cost encompasses two components: linehaul transportation (trucks) and sorting operations at SCs. The MMNDP’s high-dimensional decision space, black-box objective function, and large-scale real-world instances make exact methods computationally intractable. The networks we consider involve up to 200,000 commodity flows, whereas exact approaches in the literature typically handle only a few thousand commodities. Moreover, even simpler variants of the MMNDP, such as capacitated multi-commodity network flow, are NP-Hard [15].

Our work extends Benidis et al. [2] and Brubaker et al. [4] by addressing real-world problem complexity at industrial scale. We handle instances at least ten times larger than those in prior academic literature while incorporating the full complexity of operational constraints and black-box objective functions. To achieve this scale, we design a metaheuristic based on variable neighborhood descent with specialized local search operators, lazy neighborhood evaluation, and move caching mechanisms that enable the algorithm to quickly produce high-quality solutions. Our work also addresses a recognized gap in network design research, where proper metaheuristic approaches remain underdeveloped [20]. Moreover, because many real-world optimization problems share similar characteristics (high-dimensional search space, large-scale instances, and black-box objectives), our algorithmic contributions are broadly applicable beyond the scope of middle mile network design.

The remainder of the paper is structured as follows. Section 2 describes the problem and provides illustrative examples to help understand the decision space. We then review the literature of network design problems in Section 3. Section 4 presents our algorithmic approach, including local search moves and computational techniques employed to speed up runtime. We discuss experiments and results in Section 5. Finally, we conclude the paper and provide future work directions in Section 6.

2 Problem

The Middle Mile Network Design Problem (MMNDP) is defined on a directed graph $G = (V, A)$ where V represents network sites and A represents arcs connecting these sites, each with an associated transit time. Sites are partitioned into fulfillment centers $F \subset V$, sort centers $S \subset V$, and delivery centers $D \subset V$.

Volume flow through the network is represented by a set of commodities K . Each commodity $k \in K$ has an origin $o_k \in F$, a destination $d_k \in D$, and a volume v_k (number of packages) expected to reach d_k . Each delivery center d_k has a critical injection time, defining the latest arrival time for packages to be delivered on the same day via last mile operations (e.g., 10:00 AM). While planning spans a week, we describe a single day for simplicity. Time is discretized into 15-minute intervals (buckets), with $T = \{1, \dots, 96\}$ denoting the 96 daily buckets.

Decisions

The MMNDP then consists of making four simultaneous decisions to optimize the network for transporting these commodities

1. **Schedule Fixed Truck Departures (FTDs):** An FTD represents a guaranteed truck departure from site i to site j at a specific time on arc $(i, j) \in A$. FTDs depart regardless of volume, even if empty. When total volume exceeds truck capacity, additional ad-hoc trucks are scheduled to transport excess volume earlier. These ad-hoc trucks are not decision variables, but rather operational responses simulated during network evaluation based on historical information about volume availability for each point in time. The ad-hoc trucks also define the profile of volume arrival at a site (SC or DC) which directly impacts labor capacity constraints. An arc may have multiple FTDs, subject to operational constraints limiting their number per arc. Formally, we define a binary variable x_{ij}^t that takes value 1 when an FTD is placed at time bucket $t \in T$ on arc $(i, j) \in A$.
2. **Select active arcs:** An arc is active if at least one commodity uses it and it contains at least one FTD - alternatively, a commodity can only use an active arc and an FTD can only be placed on an active arc. We denote with binary variable $y_{ij} = 1$ whether arc $(i, j) \in A$ is active.
3. **Determine sort center operations:** For paths containing at least one sort center, we decide whether packages are sorted or cross-docked at each intermediate facility. Sorting allows consolidation: packages from multiple commodities originating at o_k can be placed on shared pallets for transportation, then separated at the sort center. This increases processing time and costs at the sort center but improves consolidation. Cross-docking transfers packages directly without sortation, reducing processing time but requiring origin facilities to presort packages into destination-specific containers. This consumes additional resources at fulfillment centers and may reduce truck fill rates. We denote with binary variable $z_{ij} = 1$ whether the (pre)sort between facilities $i \in V$ and $j \in V$ is used (note that i and j are not necessarily an arc in case of cross-docking).
4. **Route commodities through the network:** Each commodity k is assigned a path p_k from origin o_k to destination d_k using active arcs in A . Paths are classified by the number of intermediate sort centers: *direct* paths connect origin to destination without intermediate stops ($p_k = (o_k, d_k)$), *one-touch* paths use a single sort center ($p_k = (o_k, s, d_k)$ for $s \in S$), and *multi-touch* paths use two or more sort centers. Furthermore, each commodity path must use *exactly* one FTD per arc in p_k as well as define sort center operations. To that end, we denote with binary variables $\psi_{ij}^k \leq y_{ij}$ whether commodity $k \in K$ uses arc y_{ij} in its path p_k , for $(i, j) \in A$; with variable $\chi_{ij}^{kt} \leq x_{ij}^t$ whether k uses FTD x_{ij}^t ; and with variable $\zeta_{ij}^k \leq z_{ij}$ whether k uses sort z_{ij} .

Constraints

All four decisions have to be made while accounting for the following constraints.

- C1** *Shift opening hours*: Each site has operating hours defining when trucks may depart (FCs and SCs) or arrive (SCs and DCs).
- C2** *Minimum time difference between two FTDs on the same arc*: Multiple FTDs on the same arc must be separated by a minimum time interval (e.g., 1 hour).
- C3** *Maximum FTDs departure and arrival per hour*: Sites can accommodate only a limited number of truck arrivals or departures per hour due to yard, dock door, and labor constraints. This is enforced by limiting inbound and outbound FTDs within any one-hour window.
- C4** *Labor capacity at SCs and DCs*: Each site can process a limited volume per time bucket (15-minute interval), constraining when and how much volume can arrive due to labor limitations. This prevents scheduling all FTDs late in the shift, which would create arrival peaks exceeding processing capacity before outbound connections. At SCs, outbound connections are downstream FTDs continuing middle mile transportation; at DCs, outbound connections represent last mile delivery. This is one of the most complex constraints because it requires simulating, via another black-box evaluation, how ad-hoc trucks are scheduled for each FTD to bring volume earlier for each intermediate site (more information in Appendix A).
- C5** *Sort capacity at FCs*: FCs must reserve physical space for outbound pallets. Sorted volume destined for a SC typically requires one space regardless of how many commodities share the pallet. However, cross-docked volume requires separate spaces for each final destination, as packages must be pre-sorted at the FC rather than consolidated at the SC. This increases FC resource consumption.

The actual set of constraints in our real problem is much broader than (C1)-(C5). However, we define this subset with the main constraints for brevity. Beyond constraints (C1)-(C5), we also require mathematical constraints to ensure solution validity. Each commodity k uses exactly one FTD per arc: $\sum_{t \in T} \chi_{ij}^{kt} = 1$ for all $(i, j) \in p_k$. Similarly, sort operations cannot overlap on sub-paths: for any $i, j, l \in V$ where l appears between i and j in p_k , if $\zeta_{ij}^k = 1$, then $\zeta_{il}^k = 0$ and $\zeta_{lj}^k = 0$. We omit the full mathematical formulation for brevity; appendix A provides a mathematical description of constraints (C1)-(C5). Interested readers are referred to [4] for a more detailed model.

We denote by σ a solution to the MMNDP and by $\mathcal{X}$ the set of all *feasible solutions* to the problem, for which the assignment of decision variables comply with constraints (C1)-(C5). Unlike typical academic studies that assume empty initial networks, we assume an initial solution σ_0 representing the current operational network. This reflects the practical setting where planners incrementally improve the existing network design. Solution quality is assessed via the objective function defined next.

Objective function

The MMNDP maximizes the expected entitlement $f(\sigma)$ of the designed network σ :

$$\max_{\sigma \in \mathcal{X}} f(\sigma) = \mathcal{S}(\sigma) - \mathcal{L}(\sigma) - \mathcal{C}(\sigma) \quad (1)$$

where $\mathcal{S}(\sigma)$ represents the monetary value of network speed, $\mathcal{L}(\sigma)$ represents linehaul transportation costs, and $\mathcal{C}(\sigma)$ represents sort center processing costs. Let us now turn to each component individually.

Speed evaluation $\mathcal{S}(\sigma)$: We use the same black-box evaluation function from [4], trained on historical data to predict customer order coverage. The function simulates customer orders associated with a particular DC and their assignment to the FC providing fastest delivery and with inventory available.

Let $\Pi(k, \sigma)$ denote the promise (delivery days) for commodity $k \in K$ under solution σ , which depends on the FTD placements χ used by path p_k . For promises $\pi \in [0, 7]$ (from same-day delivery up to a full week), the order coverage of commodity k is:

$$\Phi(k, \sigma, \pi) = \begin{cases} 0 & \text{if } \Pi(k, \sigma) > \pi \\ \phi_{\text{black-box}}(k, \sigma, \pi) & \text{if } \Pi(k, \sigma) \leq \pi \end{cases}$$

where $\phi_{\text{black-box}}$ is the black-box coverage function. When $\pi = \Pi(k, \sigma)$, coverage depends on the order cutoff time determined by which FTD $\chi_{o_k j}^{kt}$ is used departing from origin o_k . This is because with early departure, fewer customer orders can be attained by the FTD and will depart only later or next day - hence, later FTDs cutoffs enable higher coverage. When $\pi > \Pi(k, \sigma)$, we assume all orders are covered, regardless of cutoff. Order coverage is converted to monetary value using promise-dependent weights w_π (lower π yields higher value). The total speed evaluation is:

$$\mathcal{S}(\sigma) = \sum_{k \in K} \sum_{\pi \in [0, 7]} w_\pi \cdot \Phi(k, \sigma, \pi)$$

Linehaul costs $\mathcal{L}(\sigma)$: For an FTD x_{ij}^t , we first define the volume it needs to transport as v_{ij}^t computed by:

$$v_{ij}^t = \sum_{k \in K} \chi_{ij}^{kt} \cdot v_k$$

For a truck capacity q and fixed cost τ per truck, the linehaul cost is then:

$$\mathcal{L}(\sigma) = \sum_{t \in T} \sum_{(i,j) \in A} \left\lceil \frac{v_{ij}^t}{q} \right\rceil \cdot \tau$$

The ceiling operator ensures we always count full truck departures, even when volume is low, reflecting the guaranteed departure property of FTDs.

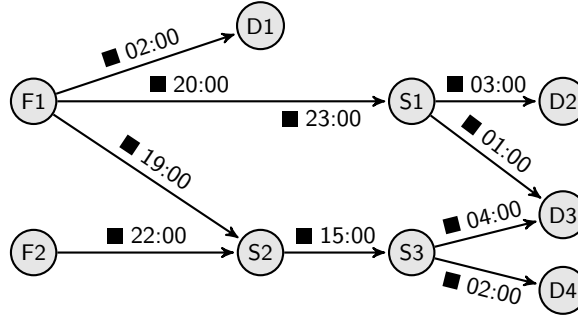
Sort costs $\mathcal{C}(\sigma)$: Sort centers incur per-package costs that depend on the operation type. Sorted volume costs α_{sort} per package, while cross-docked volume costs α_{xd} per package - typically $\alpha_{\text{xd}} < \alpha_{\text{sort}}$. We denote for a sort center s the total number of sorted packages processed as v_{sort}^s and total cross-docked packages as v_{xd}^s . The total sort cost then is:

$$\mathcal{C}(\sigma) = \sum_{s \in S} \alpha_{\text{sort}} \cdot v_{\text{sort}}^s + \alpha_{\text{xd}} \cdot v_{\text{xd}}^s$$

Let us now turn to a brief example that will help illustrate the key concepts of our problem.

Example

To illustrate the MMNDP, consider the example network in Figure 1. We assume six commodities (F1-D1, F1-D2, F1-D3, F1-D4, F2-D3, F2-D4), each with a volume of 1000 packages. Trucks have a capacity of 1000 packages and cost $\tau = \$500$ per trip. All paths through sort centers (SCs) require a 3-hour processing time at $\alpha_{\text{sort}} = \$1$ per package. Each arc has a 1-hour travel time. Commodities follow shortest-path routes: F1-D1 is a *direct* connection, F1-D2 is a *one-touch* path through S1, and F1-D4 is a *two-touch* path via S2 and S3.



■ **Figure 1** Example of an MMNDP network.

Timing and customer promises. The FTDs, shown as black squares with departure times, create time connections through the network. For example, F1-D1 has one FTD at 02:00. Between F1 and S1, there are two FTD waves at 20:00 and 23:00. Volume for F1-D2 flows via the 23:00 FTD in the first leg and 03:00 in the second leg. The 23:00 departure arrives at S1 by midnight, completes the 3-hour processing, and connects to the 03:00 departure to D2. This yields a 23:00 order cutoff for D2 customers with $\Pi(\text{F1-D2}, \sigma) = 1$ day promise delivery. In contrast, F1-D3 must use the 20:00 and 01:00 FTDs because the 23:00 departure arrives too late for the 01:00 downstream connection. This forces a much earlier 20:00 cutoff for D3, despite maintaining the same 1-day promise. The earlier cutoff likely reduces order volume significantly, as customers ordering after 20:00 see a 2-day promise instead. While both paths offer 1-day delivery, the monetary value of a 23:00 is likely to exceed that of a 20:00 cutoff due to captured demand. Meanwhile, F2-D3 has connections 22:00, 15:00 (+1 day) and 04:00 (+2 days), translating into $\Pi(\text{F2-D3}, \sigma) = 2$ days of promise with a cutoff of 22:00. This information is what constitutes the main input to $\mathcal{S}(\sigma)$.

Cost. The FTD in F1-D1 requires only one truck since all 1000 packages fit within capacity. However, the S2-S3 FTD requires three trucks since it carries volume for three commodities (F1-D4, F2-D3 and F2-D4), for a total of \$1500. In terms of sort costs, S2 processes 3000 packages across these flows, incurring \$3000 in sortation costs. Similar computations can be performed for all other components of the network to obtain the total linehaul and sortation costs.

3 Related work

The Middle Mile Network Design Problem belongs to the Service Network Design (SND) family of problems [5]. For a complete view over this rich literature, readers are referred to [6, 20]. The SND determines shipment paths for commodities and the required resources for transportation while consolidating flows to minimize network operating costs.

Early SND research considered basic capacity constraints [9, 20], but only recently has the literature incorporated more realistic operational constraints such as maximum inbound and outbound truck limits at sort centers [21, 14], which make the problem significantly more difficult to solve. Most SNDs ignore *labor efficiency* for sorting operations at intermediate facilities [4]. When labor capacity is taken into account, the volume arrival profile at a sort center directly impacts network speed and feasibility, as insufficient capacity can prevent appropriate processing for downstream connections. Moreover, the time required to move volume through a sort center may depend on how commodities were packed at upstream nodes, introducing interdependent capacity and routing decisions rarely addressed in prior work. To our knowledge, only Brubaker et al. [4] incorporates these constraints.

When modeling the time dimension in SNDs, most research opted for time discretization [21, 3]. While effective, this technique can quickly produce mathematical models that are too large to solve or even store in memory [3], something that can lead to several computational challenges. This in turn may force practitioners to use time granularities unsuitable for real applications. For example, we are unaware of exact models that can handle the 15-minute granularity and network scale we solve.

Regarding delivery speed, most SNDs specify the time when a commodity is available and its delivery deadlines, but lack any notion of network speed, as faster delivery provides no objective gain. The SND literature focuses mostly on cost minimization rather than speed maximization [6, 20]. Only two recent works consider speed optimization [2, 4], though Benidis et al. [2] examine networks with only FCs and DCs, eliminating the consolidation decisions central to typical SNDs. Even in this simplified setting, they show that the problem of scheduling FTDs to maximize network speed is NP-Hard.

In fact, most practical SNDs are NP-Hard, yet research has predominantly focused on exact methods [20], leaving a significant gap in scalable heuristic approaches for large-scale instances. This gap is significant for problems like the MMNDP, where high dimensionality and black-box evaluations for both objective function and constraints eliminate classical mathematical programming entirely. Brubaker et al. [4] showed that constraint programming required multiple hours on instances ten times smaller than ours, while a generic random key heuristic needed *multiple days* to obtain high-quality solutions. To our knowledge, no existing heuristic approach has been successfully demonstrated at the scale of our problem.

This motivates our adoption of Variable Neighborhood Search (VNS) [12], a metaheuristic successfully applied to diverse combinatorial problems including vehicle routing [17], graph coloring [1], and network design with relay operations [22]. Xiao and Konak [22] tackled an entirely different type of network design problem but still faced the challenge of having interdependent decision variables, much like us. In this context, they were able to achieve high-quality solutions by applying multiple search procedures within the VNS framework. Similarly, we employ Variable Neighborhood Descent (VND), a greedier VNS variant well-suited to the MMNDP's need for short runtimes while exploring high-dimensional decision spaces. Embedding several specialized neighborhoods within VND enables significant solution improvements, as we will discuss in the next section.

4 Algorithmic approach

The MMNDP's high dimensionality makes neighborhood-based local search particularly challenging. With over a hundred thousand commodities, hundreds of sites, and multiple decision types (FTDs, paths, sorting), the search space is enormous. Our heuristic defines multiple neighborhood structures, each containing a large number of candidate moves. Two challenges arise: first, fully enumerating and evaluating a neighborhood from scratch after every accepted move is prohibitively expensive - the black-box simulations incur several milliseconds per candidate, which can add up to an hour of computation time for a single exploration. Second, searching all neighborhoods jointly as one combined structure would yield an impractically large candidate set, making move selection too expensive within practical time budgets.

We address these challenges through two key design decisions. For the first, we compute neighborhoods once and cache them in priority queues, selectively updating them when the incumbent solution changes rather than regenerating from scratch. This amortizes the high cost of candidate generation across multiple iterations of the search. For the second, we

partition the search into smaller, carefully composed neighborhoods that define our VND structure. Each neighborhood groups moves that compete for similar resources and could block each other, whereas neighborhoods are ordered in such a way that earlier moves can enable better moves in later waves.

The metaheuristic approach we propose is described by Algorithm 1. We assume that an initial solution is already provided representing the current network in operation, which should be improved. The algorithm runs for up to $ITER$ iterations. In each of these, we loop over the neighborhood sets defined in $N = (N_1, N_2, N_3, \dots)$ and pick the best solution from all those which are reachable with the current neighborhood $N_n \in N$ (line 6). We denote by Q the set of priority queues holding the ordered list of candidate moves available - i.e., the neighborhood (see Section 4.2 for more details). Following the basic principle in VND, we explore neighborhood N_n until full exhaustion, always updating the best solution found with the new improving solution (lines 8-9). Once no improvement can be found, we move to the next neighborhood (lines 11-12).

■ **Algorithm 1** Variable Neighborhood Descent.

Input: Initial solution σ_0 , set of neighborhoods N , max iterations $ITER$
Output: Improved solution σ_{best}

```

1  $\sigma_{best} \leftarrow \sigma_0$ ;
2 for  $i \leftarrow 1$  to  $ITER$  do
3    $n \leftarrow 1$ ;
4    $Q \leftarrow \text{initNeighborhood}(\sigma_{best}, N_1)$ ;
5   while  $n \leq |N|$  do
6      $\sigma_* \leftarrow \text{bestImproving}(Q)$ ;
7     if  $\sigma_* \neq \emptyset$  then
8        $\sigma_{best} \leftarrow \sigma_*$ ;
9        $Q \leftarrow \text{updateNeighborhood}(\sigma_{best}, Q)$ ;
10    else
11       $n \leftarrow n + 1$ ;
12       $Q \leftarrow \text{initNeighborhood}(\sigma_{best}, N_n)$ ;
13    end
14  end
15 end
16 return  $\sigma_{best}$ 

```

Note that here we apply the *pipe* neighborhood change [13], in which we keep exploring the current neighborhood instead of resetting $n = 1$ as in standard VND. The rationale behind this decision is runtime, given how time consuming it can be to initialize a neighborhood exploration in our case - some neighborhoods can take up to one hour to initialize.

In what follows, we will describe the move types we designed for local search in Section 4.1. We define our neighborhoods formed by these moves in Section 4.2, while discuss some of our main contributions towards algorithmic techniques enabling Algorithm 1 to scale in Section 4.3. The reasoning behind multiple iterations is explained in the final Section 4.4.

4.1 Move types

Let us now turn our attention to the moves used by our local search to explore the large search space.

Direct FTD optimization

Given a direct path p_1 , this move tries to reoptimize the FTD by pushing it later in time and therefore gaining speed. When this is not possible because, e.g., there is too much volume already inbound late at the DC (recall constraint C4), another direct path p_2 may be considered to *pull its FTD earlier*. This leads to speed loss of orders in p_2 , but may enable the push of p_1 's FTD. The move then tries to tradeoff different coverages of FCs. If the FC for p_1 has a dominating coverage, we may even be able to gain most (if not all) of the speed lost from pulling p_2 . The overall idea of this move is depicted in Figure 2(a).

Indirect FTD optimization

Indirect FTD optimization differs from direct FTD due to the concept of *waves*. While in direct FTD a single placement is sufficient (because a single commodity is shipped), in the indirect network multiple commodities are shipped, potentially with largely different downstream connections and arrival dates. To maximize speed in this context, it may be worth to spend extra linehaul by adding a second (or more) FTD on the same leg, potentially reducing the promise of some commodities by an entire day. Indirect optimization can also have the reverse effect and *remove* extra waves which may not be needed given other optimizations done to the network. Figure 2(b) shows the idea behind this move type, where one FTD is converted into two waves.

Direct path addition and removal

This type of move selects one commodity $k \in K$ and converts its path from (or to) a direct path to (or from) a one-touch path. Figure 2(c) depicts the basic process behind the move. When converting to a direct path, the move also includes design of FTDs, since we always need at least one FTD per active leg to move volume between sites. When removing a direct path, FTDs are already available since we use existing active legs. The removal of a direct path may, however, re-design existing FC-SC FTDs to make the move feasible.

Adding a direct path contributes speed to the network at the expense of less consolidation. Meanwhile, removing the direct offers benefits of consolidation while losing speed - the typical tradeoff of the MMNDP. Note that we have specialized moves for direct and one-touch paths because they contribute the most speed to the network and carry the most volume.

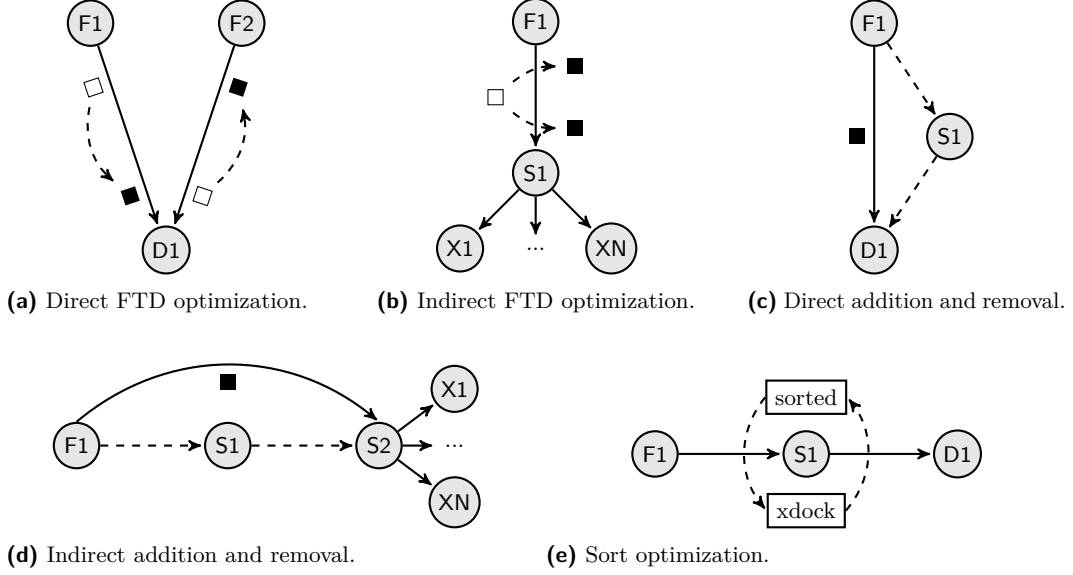
Indirect path addition and removal

Similar to the direct path addition and removal, this move is responsible for changing legs between FCs and SCs - we currently do not consider changes between SCs and other SCs or DCs. Adding a leg always involves designing new FTDs with the goal of increasing network speed, whereas removing an indirect leg will often reduce speed, but also reduce cost via consolidation. When an indirect leg is added or removed, a large number of commodities will be impacted, making this one of the most computationally expensive moves in our selection. Figure 2(d) depicts the general idea behind this type of move.

Sort optimization

As discussed in Section 2, one of the decisions in the MMNDP is also how individual commodities flow through SCs. They have essentially two modes: *sorted* and *cross-docked* (xdock). This move is responsible for looking at opportunities of optimizing this decision by either converting a sorted one-touch path into a xdock one-touch path, or by swapping two

paths (one sorted, one xdock). Note that FCs have a maximum capacity to send presorted (xdock) volume to SCs, which is why in some cases a swap may be needed. Sorted paths are slower at the SC and typically cost more, but given the limited capacity at the FCs, the decision of which path to xdock needs to be considered carefully. Figure 2(e) shows the idea behind this move.



■ **Figure 2** Move types available during local search.

4.2 Neighborhood definition

In our VND, each neighborhood is comprised of *multiple* move types. We have a total of five neighborhoods defined as follows:

- N_1 - direct path addition and direct FTD optimization
- N_2 - direct path removal and indirect FTD optimization
- N_3 - direct path addition and direct FTD optimization
- N_4 - indirect path addition and removal
- N_5 - sort optimization

The idea is that a neighborhood is then formed by the union of the neighborhoods defined by each move. The selection of move combinations to create a neighborhood is largely based on experimentation, experience and how synergies work across moves. For example, direct path removal can directly impact indirect FTD optimization since it adds more volume to an FC-SC leg. Hence, it makes sense to have both in the same neighborhood to compete for the best solution in a given iteration, rather than having them in separate neighborhoods when one could block the other move.

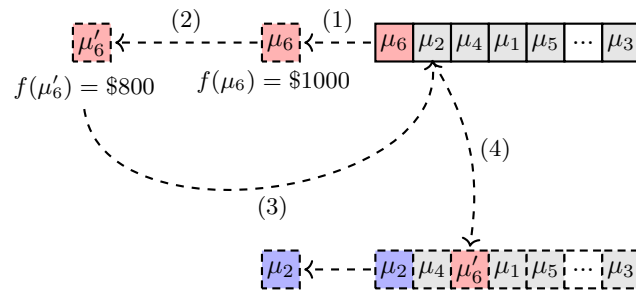
Note that neighborhoods N_1 and N_3 are the same. This is on purpose and comes from the fact that some opportunity may open after N_2 and we would like to capture these moves quickly to help ensure the speed of the network is maximized.

4.3 Neighborhood exploration, evaluation and caching

Our local search scales through several speedup techniques. Each neighborhood maintains a priority queue of candidate moves ordered by potential objective gain, initialized in lines 4 and 12 of Algorithm 1.

Neighborhood initialization. We enumerate all possible moves and add them to the priority queue. Full evaluation of the move feasibility and its objective value is computationally expensive. To reduce initialization cost, for three move types (direct path addition/removal and direct FTD optimization) we check only a subset of the feasibility constraints and order the priority queue by the upper bound estimate of the objective gain instead of the exact value of the objective. For example, direct path addition assigns the latest possible FTD, which injects at the critical time at the destination, regardless of labor capacity, providing an upper bound on potential gain without expensive computation. The remaining move types generate fully feasible candidates, as their complexity makes it more difficult to design simple upper bounds. These queues also serve as move caches to avoid redundant computations and candidate generation in our large search space.

Move selection. We employ *lazy evaluation* when selecting moves (line 6, Algorithm 1), illustrated in Figure 3. Here we start with a priority queue filled with candidate moves μ . Recall that these candidate moves may be only partially feasible (due to our initialization). The first step (1) is to take the top candidate μ_6 and optimize it. Optimizing a candidate means that we create a (new) fully feasible move over the current network state σ_{best} . Note that even fully feasible moves over a previous network state may need to be re-created at this stage. For example, if the move is to add a direct path, we will perform this path addition alongside a *proper* FTD design - and not the FTD design done during initialization. This process produces a new move μ'_6 (step 2). Optimization typically reduces the gain (here to \$800), as constraints are enforced or the network changes. We then compare μ'_6 against the next-best candidate μ_2 (step 3). If μ'_6 dominates, we select it; otherwise, we reinsert μ'_6 into the queue and repeat for μ_2 (step 4). This continues until finding a move with no superior alternative. Infeasible moves after optimization are always discarded.

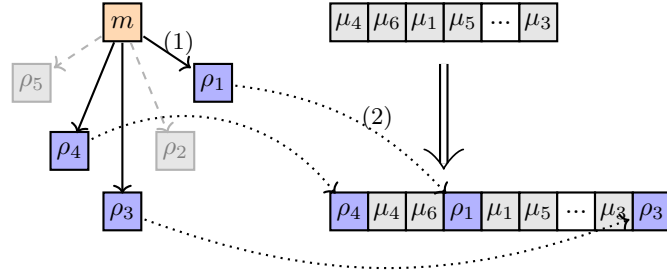


■ **Figure 3** Move selection process to extract a move from its queue during neighborhood exploration.

Neighborhood update. Full queue update is computationally prohibitive and therefore, we update the queue only partially. When a move m is accepted (line 8, Algorithm 1), the priority queues are partially updated to include newly available moves (line 9), as illustrated in Figure 4. In step (1), we enumerate new moves enabled by the acceptance of m , shown as blue moves ρ_1, ρ_3 , and ρ_4 . The remaining gray moves ρ_2 and ρ_5 represent opportunities

the algorithm does not detect, as full enumeration would be prohibitively expensive. For example, blue candidates may be those sharing the same leg as m , while gray ones are candidates on other legs which we have decided not to explore (see below for a more detailed example). This partial update increases search myopia but maintains scalability - we discuss a mitigation strategy in Section 4.4. Detected moves are added to the queue in step (2) ordered by objective gain. Note that these moves may include candidates with a structure similar to those already in the queue. This choice is by design because iterating over existing moves and updating the estimate one by one is time consuming; therefore we add the new move to the queue without removing an old move. Because the move selection process fully reevaluates a move, this prevents accepting infeasible or suboptimal moves, even if it may create duplicates of moves in the queue.

For example, when updating the queue for indirect FTD optimization, we look at the new paths created by the accepted move m (if any) and try to reoptimize all indirect legs in this new path. However, there may still be other indirect FTD moves that had their objective function reduced or increased because of the accepted move, but which will not be updated and reordered in the queue, leading the search to potentially miss some candidates.



■ **Figure 4** Process to update a move queue after a move has been accepted and the incumbent solution has changed.

Variable neighborhoods. Full queue updates or regeneration after each move acceptance are impractical. However, neighborhoods are fully initialized based on the current σ_{best} when starting exploration (lines 4 and 12 in Algorithm 1), allowing later neighborhoods to exploit information from earlier ones. For instance, N_2 generates all candidates from the σ_{best} modified by N_1 , not from the initial σ_0 . In contrast, placing all moves in a single neighborhood would only generate candidates once from σ_0 , substantially limiting move discovery. The other extreme, placing a single move per neighborhood causes the search to be too greedy and myopic due to the *pipe* search process [13] - yet, using the typical VNS process of always resetting to the first neighborhood upon solution improvement proved too time consuming.

4.4 Multiple iterations

Algorithm 1 can perform multiple iterations. The idea is to enable an easy trade-off between runtime and solution quality. While a single iteration is the most commonly setup we use in planning cycles, we have observed that up to three iterations can further provide significant benefits, particularly cost savings while maintaining the same network speed. This comes mainly from the fact that new iterations reset the candidates of each neighborhood, by-passing to some extent the limitations of our myopic queue update described in Section 4.3.

5 Experiments

To confirm the effectiveness of our algorithmic approach we perform a series of computational experiments using real world instances from previous network planning periods at Amazon. We first consider the specific characteristics of each instance, then compare results of VND against a previous algorithmic approach and finally consider the benefit brought by the multiple iterations. All of our experiments were performed in a computer equipped with Intel(R) Xeon(R) Gold 6455B running at 3GHz and equipped with 32 GB of RAM.

5.1 Instances

We consider six real-world instances in our experiments spanning different regions where Amazon operates. Table 1 shows their key characteristics, such as number of FCs, SCs, DCs and commodities.

■ **Table 1** Real-world instances used in the experiments and their key characteristics.

Instance	$ F $	$ S $	$ D $	$ K $
RW1	650	38	1297	121,098
RW2	602	27	1254	110,251
RW3	692	41	1080	180,982
RW4	570	36	1101	173,346
RW5	82	29	170	11,071
RW6	675	110	908	219,796

From this table, we note that the magnitude of the number of commodities $|K|$, one of the main drivers of instance complexity in SNDs, is approximately 10x larger than in the closely related work from Brubaker et al. [4], but is also almost 100x larger than instances in other network design studies that often include at most a few thousand commodities.

5.2 Comparison with a previous algorithmic approach

We compare the VND performance against the previous algorithmic approach used in planning and the same one used as baseline in [4]. This algorithm can be seen as our VND approach, but one where there is a single neighborhood N_1 with *all move types* in this single neighborhood N_1 . We refer to this as *baseline*. Note that both Baseline and VND are completely deterministic and therefore we report results for a single run per instance.

Table 2 presents the comparison of the two approaches. We report the relative difference between the final solution found by Baseline and the one found by VND. We do not provide absolute values to protect business financial values. Additionally, we report the total number of moves accepted by each approach, as well the absolute difference and the runtime, in hours, required by each algorithm as well as their relative difference.

It becomes clear from the results in Table 2 that VND dominates Baseline in essentially all instances except RW2. In this case, the entitlement drop is somewhat significant, reaching 7.5%. Despite that, we see that the runtime of VND is almost half that of Baseline for this same instance and we could likely achieve better solutions by fine tuning the neighborhood exploration further. However, VND provides an average improvement over these instances of 2.9% with an average runtime decrease of 24%. It is also important to note that the number of moves always increases, clearly demonstrating the additional ability of VND of exploring the search space further than the simple greedy Baseline approach.

■ **Table 2** Comparison of Baseline and VND.

Instance	Objective function	Moves			Time (h)		
	Δ (%)	Baseline	VND	Δ	Baseline	VND	Δ (%)
RW1	+1.7	338	381	+43	7.02	4.48	-36.1
RW2	-7.5	424	433	+9	8.36	4.68	-44.0
RW3	+4.4	278	314	+36	3.02	2.29	-24.1
RW4	+5.5	256	311	+55	1.99	1.84	-7.5
RW5	+5.3	387	407	+20	0.73	0.55	-24.1
RW6	+3.9	1414	1621	+207	1.70	1.49	-12.4

5.3 Improvement over multiple iterations and search progress

Let us now turn our attention to the tradeoff between runtime and solution quality when we increase the number of iterations of VND. Table 3 presents results for VND with one iteration and VND with three iterations. We present the objective function relative difference against both Baseline (Δ_B) and VND with one iteration (Δ_V). The number of moves and time are all comparing VND with one iteration against three iterations.

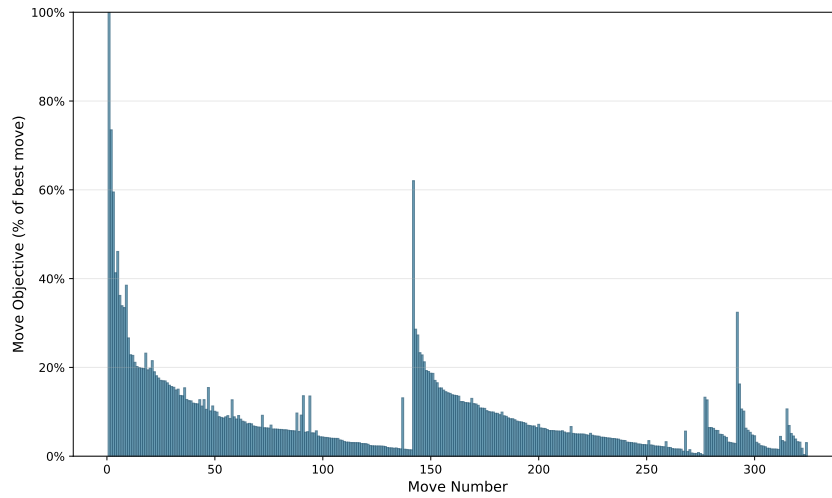
■ **Table 3** Comparison of VND runs with one iteration and three iterations.

Instance	Objective function		Moves			Time (h)		
	Δ_B (%)	Δ_V (%)	1-iter	3-iter	Δ	1-iter	3-iter	Δ (%)
RW1	+3.6	+1.9	381	455	+74	4.48	10.00	+122.9
RW2	-2.7	+5.3	433	484	+51	4.68	10.83	+131.4
RW3	+7.7	+3.2	314	346	+32	2.29	4.59	+100.3
RW4	+7.5	+1.9	311	324	+13	1.84	3.36	+82.6
RW5	+9.4	+3.9	407	440	+33	0.55	1.07	+93.9
RW6	+5.7	+1.7	1621	1734	+113	1.49	4.06	+172.4

It becomes clear that increasing the number of iterations does bring significant extra improvements compared to the single-iteration run - and even more so compared to Baseline. However, while the average gain on top of VND with one iteration is of almost 3% (and 5% for Baseline), the runtime explodes up to almost 11h in some cases. The number of moves performed also increases significantly, clearly demonstrating the capability of the algorithm in finding further new improving moves, albeit with ever increasing diminishing returns.

However, the table does not show the whole picture. Figure 5 depicts the evolution of moves performed for dataset RW4 for VND with three iterations. The x-axis shows the move acceptance index (0 is the first accepted move), while the y-axis shows the relative objective function gain of the move over the whole search (100% is the move with highest gain).

The local search progresses in the typical way: quick progression and improvement at the early stages and then quickly converging towards diminishing returns for moves. The key difference happens when we see the spikes after move index 150. This spike marks the change of neighborhood from N_1 to N_2 . Meanwhile, the transition from iteration one to iteration two happens at around move 270, at which point a new group of moves is found, with some occasional new spikes bringing additional gain, though not as high as at the beginning of the search. It becomes clear from this graph that iteration three brings little value. Indeed, a closer inspection shows that the third iteration produces only a single accepted move. Hence,



■ **Figure 5** Progress of accepted moves for RW4 and their relative objective function gain.

we could have stopped at the second iteration to save runtime without any significant loss for RW4. Nevertheless, the key message here is that re-initializing the search does bring benefits even if the precise number of iterations is instance-dependent.

6 Conclusion

In this work we have tackled a large scale middle mile network design problem, pushing the boundaries of what has been previously explored in the literature in terms of instance size, including cases with 100,000-200,000 commodities. The problem we solved seeks to maximize network speed at minimum cost, while respecting many realistic constraints with a high-dimensional decision space.

To tackle this Middle Mile Network Design Problem at scale, we design a specialized metaheuristic able to provide average improvements of 2.9% to solution quality, and a runtime up to 44% shorter when compared to a previous algorithm. We achieve these significant improvements by using a highly specialized neighborhood exploration to navigate the large search space combined with caching mechanisms, lazy evaluation and partial neighborhood updates. Furthermore, the algorithm explores different neighborhoods in a combined and sequential approach to ensure short runtimes at minimal solution quality loss. The techniques presented here are applicable beyond network design, as many real-world optimization problems share similar characteristics with the MMNDP, namely high-dimensional search spaces and expensive black-box evaluations.

Looking forward, we value the diversification of the search process. Initial experiments with perturbation mechanism to escape local minima demonstrated that we could gain further improvements of up to 5%. Unfortunately, this comes with prohibitively long computational times, which brings additional questions on how can we better explore the search space to shorten these runtimes. It may be worth exploring ways of decomposing the search, or identifying specific parts of the network which are worth (or not worth) exploring. Finally, we recognize that the order of neighborhood and move combinations in our algorithmic approach have a big impact in solution quality. Hence, it is also worth exploring approaches for automatic algorithm selection [16], where instance characteristics could be learned to automatically identify the best orders and moves to use to maximize solution quality while keeping runtimes controlled for practical use.

References

- 1 Cédric Avanthay, Alain Hertz, and Nicolas Zufferey. A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151(2):379–388, 2003. Meta-heuristics in combinatorial optimization. doi:10.1016/S0377-2217(02)00832-9.
- 2 Konstantinos Benidis, Georgios Paschos, Martin Gross, and George Iosifidis. Last truck scheduling for middle-mile next-day delivery coverage, 2025. arXiv:2310.18388.
- 3 Natasha Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017. doi:10.1287/opre.2017.1624.
- 4 Kyle Brubaker, Kyle E. C. Booth, Martin J. A. Schuetz, Philipp Loick, Jian Shen, Arun Ramamurthy, and Georgios Paschos. Optimization of next-day delivery coverage using constraint programming and random key optimizers, 2025. doi:10.48550/arXiv.2504.18749.
- 5 Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000. doi:10.1016/S0377-2217(99)00233-7.
- 6 Teodor Gabriel Crainic and Walter Rei. 50 years of operations research for the tactical planning of consolidation-based freight transportation. *EURO Journal on Transportation and Logistics*, 14:100157, 2025. doi:10.1016/j.ejtl.2025.100157.
- 7 Ethan Cramer-Flood. Worldwide retail ecommerce forecast 2025. <https://www.emarketer.com/content/worldwide-retail-ecommerce-forecast-2025>, 2025. Accessed: 2026-04-19.
- 8 Vinayak Deshpande and Pradeep K. Pendem. Logistics performance, ratings, and its impact on customer purchasing behavior and sales in e-commerce platforms. *Manufacturing & Service Operations Management*, 25(3):827–845, 2023. doi:10.1287/msom.2021.1045.
- 9 Antonio Frangioni and Bernard Gendron. 0–1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics*, 157(6):1229–1241, 2009. Reformulation Techniques and Mathematical Programming. doi:10.1016/j.dam.2008.04.022.
- 10 Katrijn Gielens. Fulfillment, the new retailer battlefield. *Journal of Retailing*, 99(1):1–4, 2023. doi:10.1016/j.jretai.2023.03.001.
- 11 Sandy Gosling, Isabelle Pan, Loraine Li, and Sandeep Shenoy. What do US consumers want from e-commerce deliveries? <https://www.mckinsey.com/industries/logistics/our-insights/what-do-us-consumers-want-from-e-commerce-deliveries#/>, 2025. Accessed: 2026-04-19.
- 12 Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001. doi:10.1016/S0377-2217(00)00100-4.
- 13 Pierre Hansen, Nenad Mladenović, Raca Todosijejić, and Saïd Hanafi. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454, 2017. doi:10.1007/s13675-016-0075-x.
- 14 Edward He, Natasha Boland, George Nemhauser, and Martin Savelsbergh. An exact algorithm for the service network design problem with hub capacity constraints. *Networks*, 80(4):572–596, 2022. doi:10.1002/net.22128.
- 15 R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. doi:10.1002/net.1975.5.1.45.
- 16 Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2019. doi:10.1162/evco_a_00242.
- 17 Jari Kytöjoki, Teemu Nuortio, Olli Bräysy, and Michel Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757, 2007. doi:10.1016/j.cor.2005.10.010.
- 18 Sara Lebow. Delivery has sped up, but US retailers still eat amazon’s dust. <https://www.emarketer.com/content/delivery-sped-up-us-retailers-amazon>, 2023. Accessed: 2026-04-19.

- 19 David Niemeier. Global parcel market 2025: Growth at scale, power in few hands. <https://ecdb.com/blog/global-parcel-market-2025-growth-at-scale-power-in-few-hands/5168>, 2025. Accessed: 2026-04-19.
- 20 Michel Gendreau Teodor Gabriel Crainic and Bernard Gendron. *Network design with applications to transportation and logistics*. Springer Cham, 2021. doi:10.1007/978-3-030-64018-7.
- 21 Haotian Wu, Ian Herszterg, Martin Savelsbergh, and Yixiao Huang. Service network design for same-day delivery with hub capacity constraints. *Transportation Science*, 57(1):273–287, 2023. doi:10.1287/trsc.2022.1155.
- 22 Yiyong Xiao and Abdullah Konak. A variable neighborhood search for the network design problem with relays. *Journal of Heuristics*, 23(2):137–164, 2017. doi:10.1007/S10732-017-9332-2.

A Constraints

We provide a formal description of our constraints below. These are not intended as a mathematical programming formulation, but rather as a precise definition of what each constraint enforces. Constraints C4 and C5 depend on other network factors (truck scheduling simulation, path definitions) not described here. Constraints relating variables are also out of scope. In what follows, we assume that operations over the time buckets $t \in T$ work in a rolling horizon. In other words, if t^* denotes the last bucket in T , $t = t^* + 1 = 1 \in T$ denotes the first period.

Constraint C1 - Shift opening hours

Let $O_i^+ \subset T$ be the set of buckets when site $i \in V$ is open for inbound and $O_i^- \subset T$ the set of buckets when it is open for outbound. The shift constraint can then be written as:

$$\begin{aligned} x_{ij}^t &= 0, & \forall (i, j) \in A, t \in T, t' = t + a_{ij} \in T \setminus O_j^+ \\ x_{ij}^t &= 0, & \forall (i, j) \in A, t \in T \setminus O_i^- \end{aligned}$$

where a_{ij} denotes the travel time (as a multiple of 15 minutes) in arc $(i, j) \in A$.

Constraint C2 - Minimum time difference between two FTDs on the same arc

Let L_{ij} denote the minimum time difference, in 15-minutes buckets, required between two FTDs placed on arc $(i, j) \in A$. The constraint then is written as:

$$x_{ij}^t + x_{ij}^{t'} \leq 1, \forall (i, j) \in A, t \in T, t' \in [t + 1, t + L_{ij}], t \neq t'$$

Constraint C3 - Maximum FTDs departure and arrival per hour

Let H_i^+ denote the maximum FTD inbound per hour at site $i \in V$ and H_i^- the maximum FTD outbound per hour at i .

$$\begin{aligned} \sum_{(i,j) \in A} \sum_{t'=t}^{t+4} x_{ij}^{t'} &\leq H_i^-, & \forall i \in V, t \in T \\ \sum_{(j,i) \in A} \sum_{t'=t-a_{ij}}^{t-a_{ij}+4} x_{ij}^{t'} &\leq H_i^+, & \forall i \in V, t \in T \end{aligned}$$

where here we look into windows of 4 buckets (1 hour).

Constraint C4 - Labor capacity at SCs and DCs

Labor capacity at SCs and DCs is our most complex constraint, requiring two simulations: (1) simulating truck schedules based on volume assigned to an FTD under the current network state σ (since ad-hoc trucks are scheduled before FTDs to spread volume arrivals over time); and (2) simulating sortation at each site for the inbound truck volume, which depends on how much labor capacity is available.

Building the ad-hoc truck schedule is out of scope for this paper and is done so using a black-box simulation that maps an FTD to a complete truck schedule based on the amount of volume assigned to the FTD. Briefly, we use historical volume accumulation curves to estimate how many packages are available for ad-hoc trucks departing before the FTD truck.

Let u_i^t denote total inbound volume at time t at site $i \in S \cup D$ (summed over all truck arrivals), b_i^t the backlog of unprocessed packages at site i and time t , c_i^t the maximum processing capacity at site i and time t , and $[t_i^{start}, t_i^{end}]$ the sortation window within which all packages must be sorted. We assume that $u_i^t \geq 0$, $t \in [t_i^{start}, t_i^{end}]$ and $u_i^t = 0$ otherwise. Volume that arrives earlier is aggregated at t_i^{start} and volume should not be injected later than t_i^{end} . We then define our labor constraints as:

$$\begin{aligned} b_i^{t^{start}} &= u_i^{t^{start}}, & \forall i \in S \cup D \\ b_i^t &= \max\{0, b_i^{t-1} - c_i^{t-1} + u_i^t\}, & \forall i \in S \cup D, t \in [t_i^{start} + 1, t_i^{end}] \\ b_i^{t^{end}} &= 0, & \forall i \in S \cup D \end{aligned}$$

Constraint C5 - Sort capacity at FCs

Let P_i denote the maximum number of sorts that FC $i \in F$ can process. We count each destination connected via a sort or presort as one resource consumption for simplicity.

$$\sum_{j \in V} z_{ij} \leq P_i, \forall i \in F$$

Computing the Cost of Stability in Periodic Microscopic Railway Timetabling via Logic-Based Benders Decomposition

Tongcheng Ouyang ✉ 

Institute for Transport Planning and Systems, ETH Zurich, Switzerland

Florian Flükiger-Fuchs ✉ 

Institute for Transport Planning and Systems, ETH Zurich, Switzerland

Bernardo Martin-Iradi ✉ 

Institute for Transport Planning and Systems, ETH Zurich, Switzerland

Francesco Corman ✉ 

Institute for Transport Planning and Systems, ETH Zurich, Switzerland

Abstract

Periodic railway timetables require stability in operation, but the cost of increasing stability in terms of total travel time remains unclear. We study this cost of stability in microscopic periodic timetabling using the realisable cycle time as a stability measure. To implement stability, we compress the timetable to a shorter realisable cycle time that remains feasible at the microscopic level, and then rescale it back to the nominal period used in regular operation. The difference between this nominal period and the realisable cycle time becomes the time supplement available for absorbing delays. We formulate an optimization problem that jointly considers realisable cycle time and total travel time, and solve it with Logic-Based Benders Decomposition. The master problem optimizes commercial timing variables, while the subproblem verifies microscopic feasibility with routing, periodicity, and resource-conflict constraints. Infeasible subproblems yield activity cuts that constrain the master. Experiments on two Swiss railway case studies show that the proposed decomposition substantially improves computational performance on the larger instance and reveals nonlinear cost-of-stability frontiers with topology-dependent infeasible cycle-time regions and exceptionally high marginal costs at certain stability levels.

2012 ACM Subject Classification Applied computing → Transportation; Mathematics of computing → Discrete optimization

Keywords and phrases Railway timetabling, periodic stability, microscopic timetabling, logic-based Benders decomposition

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.16

Category Short Paper

Funding This research received funding from the ETH Mobility Initiative, project RaDiCa.

1 Introduction

Periodic timetables are a central planning instrument in passenger railways because they provide a regular and memorable public service offer and are widely used in European passenger railway planning [2]. Their operational value, however, depends not only on feasibility in the nominal plan but also on stability: the ability of delayed trains to return to their scheduled train paths by using the available time supplement [5].

A useful stability proxy for periodic timetables is the minimum realisable cycle time. Given a nominal period T , a timetable whose minimum realisable cycle time ω satisfies $\omega < T$ has positive cycle-time slack which implies finite delay settling under realisability



© Tongcheng Ouyang, Florian Flükiger-Fuchs, Bernardo Martin-Iradi, and Francesco Corman; licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 16; pp. 16:1–16:6

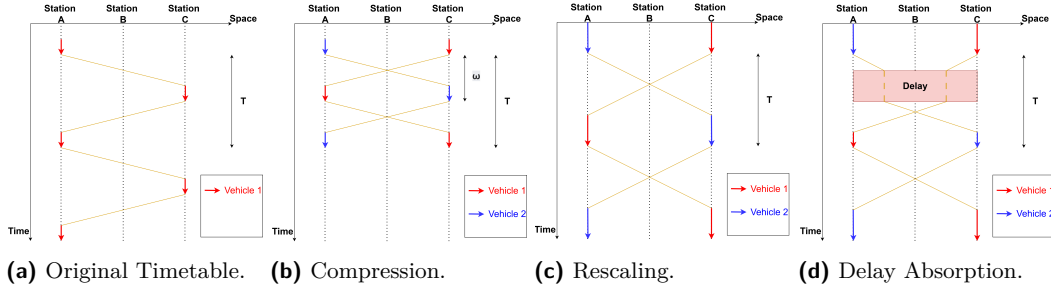


OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

assumptions [8]. A stable periodic timetable can be generated conceptually in two steps: compression and rescaling. In the compression step, one searches for a timetable structure that can be operated with a realisable cycle time $\omega < T$. In the rescaling step, this compressed timetable is rescaled back to the nominal period T for regular operation. The time difference $T - \omega$ is then available as time supplement [8]. Supplement could in principle be allocated in different ways, and this paper follows the established uniform rescaling construction, which guarantees the supplement in all the critical cycles after rescaling [8]. Fuchs et al. [4] extend the stability implementation method to single-track railway networks and prove the necessity of microscopic infrastructure modelling where resource conflicts dominate.

Figure 1 demonstrates an example of stability implementation and delay absorption. In an original timetable (Figure 1a) with no stability consideration, trains run at maximum speed between station A and C. In the compression step (Figure 1b), we alter the timetable structure by allowing trains to cross at B and optimize the realisable cycle time. The timetable is uniformly rescaled in Figure 1c. When a delay event occurs (Figure 1d), the timetable can adjust to absorb delay in the following train run and return to the normal schedule without intervention.



■ **Figure 1** An example of implementing timetable stability.

Increasing stability does not come for free. Existing approaches show that obtaining stability may require relaxing service frequencies, regularity requirements, or running-time bounds [1]. Here, we focus on the travel-time cost induced by the compression and rescaling. We define travel time as the sum of time durations of travel activities, including running between stations and dwelling at stations. When the compressed timetable is uniformly rescaled from cycle time ω to the nominal period T , each duration Δt is multiplied by the factor T/ω . With A^{com} denoting the set of commercial activities a in the timetable that contribute to the travel time, the total travel time of the rescaled timetable TT^r is

$$TT^r = \frac{T}{\omega} \cdot \sum_{a \in A^{\text{com}}} \Delta t_a, \quad (1)$$

The realisable cycle time ω can affect the total travel time in two aspects: Reducing ω increases the rescaling factor T/ω , which increases the rescaled travel time if the sum of Δt_a is the same. The effect can be observed between Figure 1c and Figure 1a where trains have longer running times after rescaling. At the same time, the combination and duration of Δt_a depend on the timetable structure solved under the chosen cycle time ω , including crossings, waiting-time allocations, route choices, and resource-conflict resolutions. The second effect can only be observed in a more complex network. The net effect of ω on travel time is therefore not evident a priori. This motivates the research question: For a periodic timetable solved with the stable timetable generation process, including compression and uniform rescaling, how much more travel time must be paid as costs for additional stability?

The paper makes three contributions. First, we propose a microscopic periodic timetabling model that jointly optimizes the stability and total travel time and quantifies the trade-off. Second, we extend an LBB approach with activity cuts that exploit the timetable's periodic structure and accelerate the solution process in the larger case study. Third, we conduct experiments on Swiss railway networks and reveal nontrivial properties of the trade-off frontier between stability and travel time.

2 Problem Setting and Stability Measure

We consider a given line plan, fixed train frequencies, and a microscopic infrastructure model. The underlying feasibility model follows the periodic microscopic timetabling framework of Fuchs et al. [4]. It combines a train-flow representation for route choices with an event-activity network for timing decisions. Events $e \in E$ have continuous times t_e , activities $a = (i, j) \in A$ impose lower and upper duration bounds (adjusted consistently for the compressed and rescaled timetables), and selected train movements occupy microscopic resource paths. Resource conflicts require a precedence decision and a minimum separation time. Periodic constraints impose that each event repeats after a cycle time ω .

We quantify the trade-off between two objectives: stability and travel time. Stability is represented by the realisable cycle time ω . Lower ω indicates more time supplement.

The total travel time is defined as the sum of durations of all commercial activities including running between stations and dwelling at stations. Let $A^{\text{com}} \subseteq A$ be the set of commercial activities in the timetable, whose durations contribute to the total travel time. Following the generation steps of timetable compression with realisable cycle time ω and rescaling, the total travel time of the generated stable timetable is

$$TT^r = \frac{T}{\omega} \cdot \sum_{a=(i,j) \in A^{\text{com}}} (t_j - t_i) \quad (2)$$

For design purposes, the two objectives can be combined in the weighted scalarization

$$\min \quad \lambda \cdot c_\omega \cdot \omega + (1 - \lambda) \cdot c_{TT} \cdot TT^r \quad (3)$$

where $\lambda \in [0, 1]$ is the weight, and $c_\omega, c_{TT} > 0$ normalize the two terms. In the computational evaluation below, we use an alternative fixed-stability formulation. For each sampled cycle time $\bar{\omega}$, we fix $\omega = \bar{\omega}$ and solve $f(\bar{\omega}) = \min TT^r$ subject to the same microscopic feasibility constraints. This measures the travel-time cost required for a prescribed stability level.

3 Logic-Based Benders Decomposition with Activity Cuts

3.1 Decomposition Structure

The microscopic stability model is a mixed-integer programming (MIP) model with continuous event times and binary decisions for routing, resource activation, and train precedence on conflicting resource paths [4]. Solving the full MIP directly is computationally challenging because microscopic feasibility is governed by a large set of routing and resource-conflict decisions, while the objective depends only on a smaller set of commercial timing variables. This structure motivates a decomposition.

We extend the microscopic-timetabling LBB framework of Leutwiler and Corman [7] with the activity cuts introduced below. The original problem is decomposed into a master problem, which contains only commercial event times, cycle time ω and accumulated cuts, and a subproblem, which verifies routing and resource-conflict feasibility.

For a candidate master solution (ω^*, t^*) , the subproblem fixes the variables from the master and verifies all the routing, periodicity, and resource-separation constraints. The subproblem is a pure feasibility problem that can be solved with high efficiency [7]. If the subproblem is feasible, the candidate solution is optimal for the original problem. If the subproblem is infeasible, a cut is generated and added to the master for the next iteration.

3.2 Positive-Cycle Certificates and Activity Cuts

The subproblem can be represented as a graph of difference constraints. A difference constraint $t_j - t_i \geq \delta_{ij}$ is represented by a directed arc from i to j with a weight δ_{ij} . A set of selected routing and precedence decisions therefore induces a weighted directed graph. The subproblem is infeasible if the corresponding graph contains a positive cycle. Let a cycle be decomposed into master-induced arcs S_m^D and activated subproblem arcs S_s^D , where D denotes a routing and precedence decision pattern that induces the cycle. Summing the difference constraints around the cycle cancels the event times and requires

$$0 \geq \sum_{(i,j) \in S_m^D} \delta_{ij} + \sum_{(k,l) \in S_s^D} \delta_{kl}. \quad (4)$$

A positive right-hand side violates inequality (4) and therefore certifies infeasibility. To avoid this infeasibility, the following activity cut derived from inequality (4) is added to the master:

$$\bigvee_{D \in \mathcal{C}} \left(\sum_{(i,j) \in S_m^D} (t_j - t_i) \leq -\Delta_D \right) \quad (5)$$

where $\Delta_D = \sum_{(k,l) \in S_s^D} \delta_{kl}$ is the constant aggregate contribution of the subproblem arcs for a fixed decision pattern D and $\mathcal{C}$ is the set of cycle-inducing decision patterns.

The activity cut is particularly effective in periodic timetabling, where infeasibility often arises from aggregation of multiple periodicity links. If the master-induced arcs in a certificate contain only periodicity links, each such link can be written in the form of $t_j - t_i = k_{ij}\omega$, $k_{ij} \in \mathbb{Z}$. The aggregate master term in the cut (5) then collapses to a direct period cut. With $K_D = \sum_{(i,j) \in S_m^D} k_{ij}$, the cut becomes

$$K_D \omega \leq -\Delta_D \quad (6)$$

The cut allows the LBBD scheme to use periodic infeasibility certificates to constrain ω directly, rather than only excluding local timing patterns.

3.3 Algorithmic Outline

The LBBD scheme applies to both formulations. For the weighted formulation in Equation (3), the master jointly minimizes ω and TT^r . For the fixed-stability instances used in the experiments, the same scheme is applied with the additional master constraint $\omega = \bar{\omega}$ and the single objective $\min TT^r$.

1. Solve the master problem with commercial constraints, ω , and accumulated cuts.
2. Fix the resulting commercial event times and ω in the microscopic subproblem.
3. If the subproblem is feasible, store (ω, TT^r) and the corresponding timetable.
4. If the subproblem is infeasible, extract a positive-cycle certificate and add the corresponding activity cut to the master.
5. Repeat until the current instance is solved or proved infeasible.

4 Computational Experiments

4.1 Case Studies

We study two cases on the Swiss Rhaetian Railway: CHUR-AROSA, a single-track corridor with passing loops, and CHUR-CORE, a mostly double-track network. For each sampled cycle time $\bar{\omega}$, we solve a fixed-stability instance with $\omega = \bar{\omega}$, nominal cycle time $T = 3600$ s, and the objective of minimizing TT^r . We compare the benchmark MIP and the LBB method on these instances; Table 1 reports the instance characteristics and runtime statistics. All experiments ran on an AMD Ryzen 7 PRO 8840U with 32 GB RAM, using OpenBus [3] in Python and gurobipy 13.0.1 [6], and terminated at a 0.5% optimality gap or after 1800 s.

Table 1 Case-study statistics.

Case	Topology	# of Nodes	# of Links	# of Conflicts	Avg. # of cuts	Avg. MIP runtime	Avg. LBB runtime
CHUR_AROSA	Single-track	3 169	5 032	21 640	10.79	1.45 s	2.93 s
CHUR_CORE	Double-track	58 945	83 635	607 338	3.6	*	148.36 s

* The MIP did not converge within the time limit.

4.2 Trade-Off Frontiers

Figure 2 shows the cost-of-stability frontier and the marginal costs of stability for both case studies. The plotted frontier uses cycle-time intervals of 36 seconds, equal to 1% of the nominal 3600-second period. Infeasible grid points are shown as dotted segments. For a solution at cycle time ω , the marginal cost is computed as the increase in rescaled total travel time when moving from the neighboring sampled solution at $\omega + 36$ s to ω .

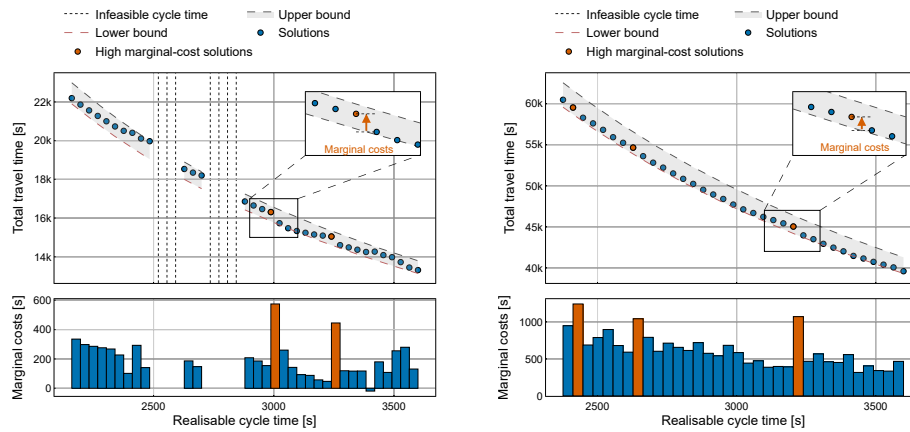


Figure 2 Trade-off between realisable cycle time ω and rescaled total travel time for CHUR-AROSA (left) and CHUR-CORE (right). Solutions with high marginal costs are highlighted.

The frontier shows that within some cycle-time ranges, the realisable cycle time can be reduced with moderate increases in rescaled total travel time, indicating moderate marginal costs. The costs reflect the combined effect of reducing cycle time: the rescaling increases travel time, while travel-time optimization can alleviate the marginal costs. Marginal costs at certain cycle times are exceptionally high, e.g., at 3024 s in CHUR-AROSA and 3204 s

in CHUR-CORE. This shows that stability should not be optimized only toward a single prescribed level: for operational planning, selecting a stability level without inspecting the frontier may therefore lead to unnecessarily high travel-time costs.

The two cases differ mainly in the infeasible cycle-time regions. The single-track infrastructure in CHUR-AROSA excludes certain sampled cycle-time levels and creates discontinuous feasible regions, whereas the mostly double-track infrastructure in CHUR-CORE allows a smoother adjustment of the timetable structure. The benefit of LBBD is most visible in the larger CHUR-CORE instance: the benchmark MIP does not converge within the 1800 s time limit, whereas LBBD solves the instances in 148.36 s on average.

5 Conclusion

We quantified the travel-time cost of periodic stability in microscopic railway timetabling through the trade-off between realisable cycle time and total travel time. We used Logic-Based Benders Decomposition, where commercial timing decisions are handled in the master problem and detailed routing, periodicity, and resource conflicts are checked in the subproblem. Infeasible subproblems with positive-cycle certificates yield activity cuts for the master. Experiments on two Swiss networks show that the frontier is topology-dependent: moderate stability gains can often be obtained at low travel-time cost, while certain stability requirements can be increasingly expensive or infeasible. Further work should extend the analysis to other networks, interpret infeasibility certificates in physical infrastructure terms, and develop stronger cut aggregation for LBBD.

References

- 1 Nikola Bešinović, Egidio Quaglietta, and Rob M. P. Goverde. Resolving instability in railway timetabling problems. *EURO Journal on Transportation and Logistics*, 8(5):833–861, 2019. doi:10.1007/s13676-019-00148-3.
- 2 Gabrio Caimi, Leo Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning & Management*, 6(4):285–312, 2017. doi:10.1016/j.jrtpm.2016.11.002.
- 3 Florian Fuchs and Francesco Corman. An open toolbox for integrated optimization of public transport. In *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–7. IEEE, 2019. doi:10.1109/MTITS.2019.8883335.
- 4 Florian Fuchs, Bernardo Martin-Iradi, and Francesco Corman. Optimizing periodic stability in railway timetables: A microscopic model for networks with a macroscopic comparison. *Journal of Rail Transport Planning & Management*, 36:100555, 2025. doi:10.1016/j.jrtpm.2025.100555.
- 5 Rob M. P. Goverde and Ingo A. Hansen. Performance indicators for railway timetables. In *2013 IEEE International Conference on Intelligent Rail Transportation Proceedings*, pages 301–306. IEEE, 2013. doi:10.1109/ICIRT.2013.6696312.
- 6 Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, 2026. URL: <https://www.gurobi.com>.
- 7 Florin Leutwiler and Francesco Corman. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research*, 303(2):525–540, 2022. doi:10.1016/j.ejor.2022.02.043.
- 8 Daniel Sparing and Rob M. P. Goverde. A cycle time optimization model for generating stable periodic railway timetables. *Transportation Research Part B: Methodological*, 98:198–223, 2017. doi:10.1016/j.trb.2016.12.020.

Integrated Location-Routing-Scheduling for Multi-Trip Mobile Parcel Lockers with Time Windows

Salma Essouaied ✉ 

CIRRELT and Faculté des Sciences de l'Administration, Université Laval, Québec, Canada
Univ. Polytechnique Hauts-de-France, LAMIH, CNRS, UMR 8201, Valenciennes F-59313, France
INSA Hauts-de-France, Valenciennes F-59313, France

Leandro C. Coelho ✉ 

CIRRELT and Faculté des Sciences de l'Administration, Université Laval, Québec, Canada
Canada Research Chair in Integrated Logistics, Université Laval, Québec, Canada

Christophe Wilbaut ✉ 

Univ. Polytechnique Hauts-de-France, LAMIH, CNRS, UMR 8201, Valenciennes F-59313, France
INSA Hauts-de-France, Valenciennes F-59313, France

Raca Todosijević ✉ 

Univ. Polytechnique Hauts-de-France, LAMIH, CNRS, UMR 8201, Valenciennes F-59313, France
INSA Hauts-de-France, Valenciennes F-59313, France

Abstract

Mobile parcel lockers can reposition delivery capacity during the day, but this flexibility requires decisions that are tightly linked: where lockers stop, which customers are assigned to each stop, how lockers move, and when service takes place. We introduce the multi-trip mobile parcel locker problem with time windows (MT-MPLP-TW), which integrates locker deployment, customer assignment, parking selection, routing, scheduling, parking capacity, walking limits, and depot replenishment. We present a compact mixed-integer programming formulation and a logic-based Benders decomposition (LBBD) that separates route-and-assignment decisions from detailed time-expanded certification. Computational results on generated instances and on the single-trip mobile parcel locker benchmark show that the compact formulation is effective on small and medium instances, while LBBD gives stronger performance on larger and more constrained cases.

2012 ACM Subject Classification Applied computing → Transportation; Theory of computation → Integer programming; Mathematics of computing → Combinatorial optimization

Keywords and phrases Mobile parcel lockers, City logistics, Routing and scheduling, Mixed-integer programming, Logic-based Benders decomposition

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.17

Category Short Paper

1 Introduction

Parcel lockers reduce failed deliveries and consolidate last-mile demand, but fixed lockers require permanent infrastructure and cannot easily adapt to demand that varies over space and time. This limitation has motivated mobile-locker concepts, in which locker capacity can be repositioned during the day [4]. Mobile parcel lockers (MPLs) can be temporarily parked close to customers, replenished at the depot, and reused over multiple trips.

This paper introduces the multi-trip mobile parcel locker problem with time windows (MT-MPLP-TW). The problem extends single-trip mobile parcel locker planning by allowing each deployed locker to perform several depot-to-depot trips during the planning horizon. It asks which MPLs should be deployed, which parking locations they should visit, which



© Salma Essouaied, Leandro C. Coelho, Christophe Wilbaut, and Raca Todosijević;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 17; pp. 17:1–17:6



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

customers should be assigned to those visits, and how the resulting trips should be scheduled. These decisions must respect customer availability, parking availability, parking capacity, locker capacity, walking limits, travel times, and depot replenishment. The problem fills a modeling gap because it treats locker location as an operational routing and scheduling decision rather than as a fixed network-design decision.

We make three contributions. First, we propose a compact mixed-integer programming formulation for MT-MPLP-TW. Second, we develop an LBBD algorithm in which the master problem selects a route-and-assignment structure and a certification subproblem checks the detailed schedule. Third, we compare the compact formulation and LBBD on generated instances and on the public ST-MPLP benchmark, showing that the decomposition is more scalable when assignment, routing, and timing are strongly coupled.

2 Problem Description and Formulation

We consider customers C , parking locations P , and a fleet M of MPLs. Each locker may perform several trips $r \in R$ from a depot 0, where a trip is a depot-to-depot tour that can visit several parking locations before returning for possible replenishment. Thus, the index r denotes the trip position of locker m during the planning horizon; trips are not generated in advance. Let $N = \{0\} \cup P$ and $A = \{(i, j) \in N \times N : i \neq j\}$. Each customer c has demand η_c , a maximum walking distance e_c , possible pickup locations L_c , and service durations θ_{cl} . The service duration is the time during which the locker must remain available at the selected parking location so that customer c can complete pickup from location l . Customer availability windows are associated with customer locations, while parking windows describe when a parking location can host an MPL. For each feasible triple (c, l, p) , the set W_{clp} contains service windows obtained by intersecting customer and parking availability. Triples without a feasible service window, or violating the walking limit, are removed from F . Here, B is a sufficiently large constant used to deactivate timing constraints when the associated binary decisions are not selected.

The binary variables are x_{ijmr} for arc use, y_{pmr} for parking visits, z_{cmr} for assigning customer c to locker m and trip r , w_{clp} for customer-location-parking assignment, o_{clpw} for service-window selection, ψ_{pkmr} for parking-window selection, and u_{mr} for active trips. The variable u_{m0} indicates whether MPL m is deployed. Continuous variables are departure times t_{mr} , parking arrival times ϕ_{pmr} , stay durations s_{pmr} , and service start times γ_{clmr} .

$$\min \quad C_R \sum_{(i,j) \in A} \sum_{m \in M} \sum_{r \in R} d_{ij} x_{ijmr} + C_W \sum_{(c,l,p) \in F} e_{cl}^p w_{clp} + C_F \sum_{m \in M} u_{m0} \quad (1)$$

$$\text{s.t.} \quad \sum_{\substack{j \in N \\ j \neq p}} x_{pjm r} = y_{pmr} \quad \forall p \in P, m \in M, r \in R \quad (2)$$

$$\sum_{\substack{i \in N \\ i \neq p}} x_{ipmr} = y_{pmr} \quad \forall p \in P, m \in M, r \in R \quad (3)$$

$$\sum_{j \in P} x_{0jmr} = u_{mr} \quad \forall m \in M, r \in R \quad (4)$$

$$\sum_{i \in P} x_{i0mr} = u_{mr} \quad \forall m \in M, r \in R \quad (5)$$

$$\sum_{\substack{l \in L_c: \\ (c,l,p) \in F}} w_{clp} \leq \sum_{m \in M} \sum_{r \in R} y_{pmr} \quad \forall c \in C, p \in P \quad (6)$$

$$\sum_{(l,p):(c,l,p) \in F} w_{clp} = 1 \quad \forall c \in C \quad (7)$$

$$\sum_{m \in M} \sum_{r \in R} z_{cmr} = 1 \quad \forall c \in C \quad (8)$$

$$y_{pmr} \geq z_{cmr} + \sum_{\substack{l \in L_c: \\ (c,l,p) \in F}} w_{clp} - 1 \quad \forall c \in C, p \in P, m \in M, r \in R \quad (9)$$

$$z_{cmr} \leq u_{mr} \quad \forall c \in C, m \in M, r \in R \quad (10)$$

$$\sum_{w \in W_{clp}} o_{clpw} = w_{clp} \quad \forall (c,l,p) \in F \quad (11)$$

$$\gamma_{clmr} \geq a_{clp}^w - B(2 - o_{clpw} - z_{cmr}) \quad \forall (c,l,p) \in F, w \in W_{clp}, m \in M, r \in R \quad (12)$$

$$\gamma_{clmr} \leq b_{clp}^w - \theta_{cl} + B(2 - o_{clpw} - z_{cmr}) \quad \forall (c,l,p) \in F, w \in W_{clp}, m \in M, r \in R \quad (13)$$

$$\sum_{k \in K_p} \psi_{pkmr} = y_{pmr} \quad \forall p \in P, m \in M, r \in R \quad (14)$$

$$\phi_{pmr} \geq a_p^k - B(1 - \psi_{pkmr}) \quad \forall p \in P, k \in K_p, m \in M, r \in R \quad (15)$$

$$\phi_{pmr} + s_{pmr} \leq b_p^k + B(1 - \psi_{pkmr}) \quad \forall p \in P, k \in K_p, m \in M, r \in R \quad (16)$$

$$\phi_{jmr} \geq \phi_{imr} + s_{imr} + \frac{d_{ij}}{v} - B(1 - x_{ijmr}) \quad \forall i, j \in P, i \neq j, m \in M, r \in R \quad (17)$$

$$\gamma_{clmr} \geq \phi_{pmr} - B(2 - w_{clp} - z_{cmr}) \quad \forall (c,l,p) \in F, m \in M, r \in R \quad (18)$$

$$\gamma_{clmr} + \theta_{cl} \leq \phi_{pmr} + s_{pmr} + B(2 - w_{clp} - z_{cmr}) \quad \forall (c,l,p) \in F, m \in M, r \in R \quad (19)$$

$$u_{mr} \leq u_{m,r-1} \quad \forall m \in M, r = 2, \dots, |R| \quad (20)$$

$$t_{mr} \geq \phi_{p,m,r-1} + s_{p,m,r-1} + \frac{d_{p0}}{v} - B(1 - x_{p0,m,r-1}) \quad \forall p \in P, m \in M, r = 2, \dots, |R| \quad (21)$$

$$\sum_{(l,p):(c,l,p) \in F} e_{cl}^p w_{clp} \leq e_c \quad \forall c \in C \quad (22)$$

$$\sum_{c \in C} \eta_c z_{cmr} \leq q_m u_{mr} \quad \forall m \in M, r \in R \quad (23)$$

$$s_{pmr} \geq \theta_{cl}(w_{clp} + z_{cmr} - 1) \quad \forall (c,l,p) \in F, m \in M, r \in R \quad (24)$$

$$\sum_{m \in M} \sum_{r \in R} \psi_{pkmr} \leq b_p \quad \forall p \in P, k \in K_p \quad (25)$$

$$\phi_{pmr} \leq t_{mr} + \frac{d_{0p}}{v} + B(1 - x_{0pmr}) \quad \forall p \in P, m \in M, r \in R \quad (26)$$

$$\phi_{pmr} \geq t_{mr} + \frac{d_{0p}}{v} - B(1 - x_{0pmr}) \quad \forall p \in P, m \in M, r \in R \quad (27)$$

$$\phi_{pmr} \geq t_{mr} + \frac{d_{0p}}{v} - B(1 - y_{pmr}) \quad \forall p \in P, m \in M, r \in R \quad (28)$$

$$x_{ijmr}, y_{pmr}, u_{mr}, z_{cmr}, w_{clp}, o_{clpw}, \psi_{pkmr} \in \{0, 1\} \quad \forall (i,j) \in A, p \in P, c \in C, m \in M, r \in R \quad (29)$$

$$t_{mr}, \gamma_{clmr}, \phi_{pmr}, s_{pmr} \geq 0 \quad \forall (c,l,p) \in F, m \in M, r \in R \quad (30)$$

The objective balances locker travel, customer walking, and deployment costs. Constraints (2)–(5) define the trip routes. Constraints (6)–(10) assign customers, link them to visited parking locations, and activate trips. Constraints (11)–(16) select compatible customer and parking windows. Constraints (17)–(19) propagate route timing and synchronize customer service with locker presence. Constraints (20) and (21) impose the multi-trip structure. The remaining constraints enforce walking limits, locker capacity, parking capacity, stay duration, first-arrival timing, and variable domains. In the compact formulation, trips are optimized through the arc variables x_{ijmr} ; the index r only limits the maximum number of depot-to-depot tours each MPL may perform.

The formulation is strengthened with standard inequalities that remove weak or symmetric solutions: active trips must serve at least one customer, service-window variables can be selected only when the corresponding assignment is selected, homogeneous MPLs are ordered, trips of the same MPL are ordered, and the first departure is fixed to break symmetry. The model specializes to the ST-MPLP of [4] by imposing a single trip, full-day parking availability, and the corresponding benchmark cost structure.

3 Logic-Based Benders Decomposition

Benders decomposition separates a problem into a master problem and a subproblem [1]. Logic-based Benders decomposition extends this idea to settings where the subproblem has discrete or logical structure and cuts are derived from inference rather than linear duality [3, 2]. We use this principle to separate route-and-assignment decisions from detailed time-expanded scheduling.

The compact formulation becomes difficult because routing, assignment, and continuous timing are optimized together. The LBB master problem keeps the structural part of the model: active trips, deployed lockers, parking visits, routing arcs, customer-to-trip assignments, customer-location-parking assignments, and coarse window choices. It contains the flow, assignment, linking, walking, locker-capacity, parking-capacity, and symmetry-breaking constraints of the compact formulation, together with static compatibility inequalities. The detailed continuous timing constraints are not fully enforced in the master. They are checked only after the master proposes an integer route-and-assignment structure.

For a master solution $\bar{\Omega}$, the certification subproblem fixes the selected discrete structure. It extracts the selected trips, the ordered parking visits of each trip, the customers assigned to each trip, the selected customer-location-parking choices, and the parking and service windows selected by the master. The subproblem then searches for feasible departure times, parking arrival times, stay durations, customer service times, and trip-to-locker assignments. It verifies travel-time propagation along each trip, parking-window feasibility, customer-window feasibility, synchronization between customer service and locker presence, depot return, and replenishment between consecutive trips of the same MPL.

If the subproblem is feasible, the candidate solution is certified. The resulting objective value is compared with the incumbent, and the incumbent is updated if the certified solution is better. If the subproblem is infeasible, the selected structure cannot appear in any feasible solution of the original compact model, and a lazy feasibility cut is added to the master.

Let ξ be the vector of binary master decisions defining a candidate structure. For an infeasible structure $\bar{\xi}$, define $S^1(\bar{\xi}) = \{a : \bar{\xi}_a = 1\}$ and $S^0(\bar{\xi}) = \{a : \bar{\xi}_a = 0\}$. The fallback no-good cut is

$$\sum_{a \in S^1(\bar{\xi})} (1 - \xi_a) + \sum_{a \in S^0(\bar{\xi})} \xi_a \geq 1. \quad (31)$$

This cut removes the exact infeasible master structure. It is always valid, but it is weak because it forbids only one binary pattern.

The implementation therefore tries to identify smaller infeasible explanations. When the certification model proves that only a subset of selected decisions is responsible for infeasibility, the cut is written on that subset instead of on the full master solution. For example, if a subset $C' \subseteq C$ of customers cannot be scheduled together on trip (m, r) , the following customer-core cut is added:

$$\sum_{c \in C'} z_{cmr} \leq |C'| - 1. \quad (32)$$

Similarly, if a subset $P' \subseteq P$ of parking visits cannot be jointly scheduled on trip (m, r) , we add

$$\sum_{p \in P'} y_{pmr} \leq |P'| - 1. \quad (33)$$

If infeasibility is caused by a selected route order, represented by a set of arcs $A' \subseteq A$, the route-core cut is

$$\sum_{(i,j) \in A'} x_{ijmr} \leq |A'| - 1. \quad (34)$$

These cuts are generated from the infeasibility information returned by the certification subproblem. When an irreducible infeasible subsystem is available, its constraints are mapped back to the master decisions that activated them. Otherwise, a deletion-filtering procedure removes selected customers, parking visits, or arcs one at a time and keeps only the decisions that are necessary to preserve infeasibility.

The algorithm is implemented in a branch-and-cut framework. Integer master solutions are intercepted through callbacks, certified by the subproblem, and either accepted as incumbents or cut off. A cache stores previously evaluated route-and-assignment structures, avoiding repeated certification of identical candidates. This is important because the master may rediscover the same structure through different fractional paths in the branch-and-bound tree. The overall scheme remains exact because every accepted incumbent is certified by the time-expanded subproblem, and every infeasible cut excludes only structures that cannot be scheduled.

4 Computational Results

All experiments were conducted on a workstation equipped with an Intel Core Ultra 9 285K processor with 24 cores and 62 GB of RAM, running Ubuntu 24.04.3 LTS. Unless stated otherwise, each run used five solver threads. The time limits were 3600 seconds for generated MT-MPLP-TW instances and 1800 seconds for the ST-MPLP benchmark.

Generated instances. The generated set contains small, medium, and large instances with different operational structures. Table 1 focuses on the large instances, where scalability differences are clearest. LBBD improves the average gap in all three families and is especially effective on the multi-trip-tight family.

■ **Table 1** Average performance on large generated MT-MPLP-TW instances.

Family	Method	Incumbent	Lower bound	Gap (%)	Time (s)
Homogeneous	Compact	6069.57	5816.04	4.69	2955.79
Homogeneous	LBBD	6011.80	5952.00	0.94	1952.74
Capacity buffer	Compact	5846.67	5725.35	1.91	2007.94
Capacity buffer	LBBD	5824.32	5824.25	0.00	855.38
Multi-trip tight	Compact	7444.87	6761.89	9.66	3282.85
Multi-trip tight	LBBD	7144.27	7010.33	1.99	2796.34

ST-MPLP benchmark. For the ST-MPLP benchmark, we use the public benchmark instances of [4] and implement the formulation described in their paper. We compare this implementation with our compact formulation and LBBD under the same computational

setting. Table 2 shows that the compact model is competitive on small instances, but its gap grows with the number of customers. LBBD gives the best average gaps on the two largest groups.

■ **Table 2** Average results for ST-MPLP instance groups.

Group	ST-MPLP		Compact MIP		LBBD	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
5 customers	0.000	9.44	0.000	0.83	0.000	0.36
15 customers	4.803	467.80	4.808	405.28	0.000	17.21
30 customers	43.992	1801.20	39.487	1800.28	5.284	1139.05
50 customers	11.382	1800.57	27.200	1800.41	1.508	1650.77

5 Conclusion

We introduced the MT-MPLP-TW, an integrated model for multi-trip mobile parcel locker planning with time windows. The compact formulation captures locker deployment, assignment, parking selection, routing, scheduling, parking capacity, walking limits, and replenishment. The same formulation can be specialized to the single-trip mobile parcel locker benchmark by limiting each deployed locker to one trip.

The experiments show that the compact formulation is effective for small and medium instances, while LBBD improves performance on large and more constrained cases. The main computational difficulty is the coupling between customer assignment, route structure, and time feasibility. Future work should strengthen the master problem with sharper incompatibility cuts and extend the model to stochastic or dynamic demand.

References

- 1 Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962. doi:10.1007/BF01386316.
- 2 John N. Hooker. *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley & Sons, 2011.
- 3 John N. Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003. doi:10.1007/s10107-003-0375-9.
- 4 Stefan Schwerdfeger and Nils Boysen. Who moves the locker? A benchmark study of alternative mobile parcel locker concepts. *Transportation Research Part C: Emerging Technologies*, 142:103780, 2022. doi:10.1016/j.trc.2022.103780.