

Lecture Notes in Operations Research

Frank Phillipson *Editor*

Optimisation in Synchromodal Logistics

From Theory to Practice

 Springer

Lecture Notes in Operations Research

Editorial Board Members

Ana Paula Barbosa-Povoa, University of Lisbon, LISBOA, Portugal

Adiel Teixeira de Almeida , Federal University of Pernambuco, Recife, Brazil

Noah Gans, The Wharton School, University of Pennsylvania, Philadelphia, USA

Jatinder N. D. Gupta, University of Alabama in Huntsville, Huntsville, USA

Gregory R. Heim, Mays Business School, Texas A&M University, College Station, USA

Guowei Hua, Beijing Jiaotong University, Beijing, China

Alf Kimms, University of Duisburg-Essen, Duisburg, Germany

Xiang Li, Beijing University of Chemical Technology, Beijing, China

Hatem Masri, University of Bahrain, Sakhir, Bahrain

Stefan Nickel, Karlsruhe Institute of Technology, Karlsruhe, Germany

Robin Qiu, Pennsylvania State University, Malvern, USA

Ravi Shankar, Indian Institute of Technology, New Delhi, India

Roman Slowiński, Poznań University of Technology, Poznan, Poland

Christopher S. Tang, Anderson School, University of California Los Angeles, Los Angeles, USA

Yuzhe Wu, Zhejiang University, Hangzhou, China

Joe Zhu, Foisie Business School, Worcester Polytechnic Institute, Worcester, USA

Constantin Zopounidis, Technical University of Crete, Chania, Greece

Lecture Notes in Operations Research is an interdisciplinary book series which provides a platform for the cutting-edge research and developments in both operations research and operations management field. The purview of this series is global, encompassing all nations and areas of the world.

It comprises for instance, mathematical optimization, mathematical modeling, statistical analysis, queueing theory and other stochastic-process models, Markov decision processes, econometric methods, data envelopment analysis, decision analysis, supply chain management, transportation logistics, process design, operations strategy, facilities planning, production planning and inventory control.

LNOR publishes edited conference proceedings, contributed volumes that present firsthand information on the latest research results and pioneering innovations as well as new perspectives on classical fields. The target audience of LNOR consists of students, researchers as well as industry professionals.

Frank Phillipson
Editor


Optimisation in Synchronodal Logistics

From Theory to Practice



Springer

Editor

Frank Phillipson 

TNO

The Hague, The Netherlands

This work was supported by TKI DINALOG

ISSN 2731-040X

ISSN 2731-0418 (electronic)

Lecture Notes in Operations Research

ISBN 978-3-031-15654-0

ISBN 978-3-031-15655-7 (eBook)

<https://doi.org/10.1007/978-3-031-15655-7>

© Netherlands Organisation for Applied Scientific Research 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This book is the result of the work of (for the most part) MSc thesis interns, as part of the Comet-PS project that was active from 2017 until 2021. Partners in this project were VU University Amsterdam, TU Delft, University Twente, University of Amsterdam, Combi Terminal Twente (CTT), Air Cargo Netherlands (ACN) and the Port of Amsterdam.

The work is based on the following master's theses:

1. 'Optimising routing in an agent-centric synchromodal network with shared information', M. De Juncker, Applied Mathematics, Eindhoven University of Technology, 2017 [5].
2. 'General methods for synchromodal planning of freight containers and transports', D. Huizing, Applied Mathematics, Delft University of Technology, 2017 [8].
3. 'Container-to-mode assignment on a synchromodal transportations network: A multi-objective approach', M. Ortega Del Vecchy, Applied Mathematics, Delft University of Technology, 2017 [14].
4. 'Intermodal Transport: Routing Vehicles and Scheduling Containers', K. Kalicharan, Applied Mathematics, Delft University of Technology, 2018 [9].
5. 'A robust optimization approach to synchromodal container transportation', I. Chiscop, Applied Mathematics, Delft University of Technology, 2018 [4].
6. 'Optimisation of synchromodal transportation using user equilibria', L.A.M. Bruijns, Applied Mathematics, Delft University of Technology, 2018 [2].
7. 'A simulation based approach to synchromodal container transport', W.J. de Koning, Applied Mathematics, Delft University of Technology, 2019 [12].

Most chapters are based on papers distilled from these theses that were published earlier:

1. 'Optimising and Recognising 2-Stage Delivery Chains with Time Windows', F. Phillipson, M.R. Ortega del Vecchy, B. van Ginkel, D. Huizing and A. Sangers, 8th International Conference on Computational Logistics (Springer), Southampton (United Kingdom), 2017 [16].

2. ‘Framework of synchromodal transportation problems’, M.A.M. De Juncker, D. Huizing, M.R. Ortega del Vecchyo, F. Phillipson and A. Sangers, 8th International Conference on Computational Logistics (Springer), Southampton (United Kingdom), 2017 [6].
3. ‘Optimising routing in an agent-centric synchromodal network with shared information’, M.A.M. De Juncker, F. Phillipson, L.A.M. Bruijns and A. Sangers, 9th International Conference on Computational Logistics (Springer), Salerno (Italy), 2018 [7].
4. ‘Alternative Performance Indicators for Optimizing Container Assignment in a Synchromodal Transportation Network’, M.R. Ortega del Vecchyo, F. Phillipson and A. Sangers, 9th International Conference on Computational Logistics (Springer), Salerno (Italy), 2018 [15].
5. ‘User Equilibrium in a Transportation Space-Time Network’, L.A.M. Bruijns, F. Phillipson and A. Sangers, 6th International Physical Internet Conference (IPIC), London (UK), 2019 [1].
6. ‘Reduction of Variables in a Logistic Flow Problem’, K. Kalicharan, F. Phillipson, A. Sangers and M. De Juncker, 6th International Physical Internet Conference (IPIC), London (UK), 2019 [11].
7. ‘Decision-Making in a Dynamic Transportation Network: A Multi-objective Approach’, M.R. Ortega del Vecchyo, F. Phillipson and A. Sangers, 6th International Physical Internet Conference (IPIC), London (UK), 2019 [17].
8. ‘Cutting Planes for Solving Logistic Flow Problems’, K. Kalicharan, F. Phillipson and A. Sangers, 11th International Conference on Computational Logistics (Springer), Enschede (the Netherlands), 2020 [10].
9. ‘Simulation Approach for Container Assignment under Uncertainty’, W. de Koning, F. Phillipson and I. Chiscop, 11th International Conference on Computational Logistics (Springer), Enschede (the Netherlands), 2020 [13].
10. ‘Fair User Equilibrium in a Transportation Space-Time Network’, L.A.M. Bruijns, F. Phillipson and A. Sangers, 11th International Conference on Computational Logistics (Springer), Enschede (the Netherlands), 2020 [3].

Part of the supervision of the interns was done by Alex Sangers and Irina Chiscop. Other interns that have contributed to the project are Iris Meester, Bart van Ginkel and Sanne van Alebeek. Many thanks to all who contributed to the outcome of this project.

The Hague, The Netherlands
December 2021

Frank Phillipson

References

1. Bruijns, L., Phillipson, F., & Sangers, A. (2019). User equilibrium in a transportation space-time network. In *6th International Physical Internet Conference (IPIC)*.
2. Bruijns, L. A. (2018). Optimization of user equilibrium container transportation problems using toll pricing. Master's Thesis, TU Delft.
3. Bruijns, L. A., Phillipson, F., & Sangers, A. (2020). Fair user equilibrium in a transportation space-time network. In *International Conference on Computational Logistics* (pp. 682–697). Springer.
4. Chiscop, I. (2018). A robust optimization approach to synchromodal container transportation. Master's Thesis, TU Delft.
5. De Juncker, M. (2017). Optimising routing in an agent-centric synchromodal network with shared information. Master's Thesis, Eindhoven University of Technology.
6. De Juncker, M. A., Huizing, D., del Vecchyo, M. O., Phillipson, F., & Sangers, A. (2017). Framework of synchromodal transportation problems. In *International Conference on Computational Logistics* (pp. 383–403). Springer.
7. De Juncker, M. A., Phillipson, F., Bruijns, L. A., & Sangers, A. (2018). Optimising routing in an agent-centric synchromodal network with shared information. In *International Conference on Computational Logistics* (pp. 316–330). Springer.
8. Huizing, D. (2017). General methods for synchromodal planning of freight containers and transports. Master's Thesis, TU Delft.
9. Kalicharan, K. (2018). Intermodal transport: Routing vehicles and scheduling containers. Master's thesis, TU Delft (2018)
10. Kalicharan, K., Phillipson, F., & Sangers, A. (2020). Cutting planes for solving logistic flow problems. In *International Conference on Computational Logistics*, (pp. 569–583). Springer.
11. Kalicharan, K., Phillipson, F., Sangers, A., & Juncker, M. D. (2019). Reduction of variables for solving logistic flow problems. In *6th International Physical Internet Conference (IPIC)*.
12. de Koning, W. J. (2019). A simulation based approach to synchromodal container transport. Master's Thesis, Delft University of Technology.
13. de Koning, W. J., Phillipson, & F., Chiscop, I. (2020). Simulation approach for container assignment under uncertainty. In *International Conference on Computational Logistics* (pp. 616–630). Springer.
14. Ortega del Vecchyo, M. R. (2017). Container-to-mode assignment on a synchromodal transportation network: A multi-objective approach. Master's Thesis, TU Delft.

15. Ortega Del Vecchy, M. R., Phillipson, F., & Sangers, A. (2017). Alternative performance indicators for optimizing container assignment in a synchromodal transportation network. In *International Conference on Computational Logistics - ICCL2018* (pp. 222–235). Springer.
16. Phillipson, F., del Vecchy, M. O., van Ginkel, B., Huizing, D., & Sangers, A. (2017). Optimising and recognising 2-stage delivery chains with time windows. In *International Conference on Computational Logistics* (pp. 366–380). Springer.
17. Ortega del Vecchy, M., Phillipson, F., & Sangers, A. (2019). Decision making in a dynamic transportation network: a multi-objective approach. In *6th International Physical Internet Conference (IPIC)*.

Acknowledgements

This work has been carried out within the project ‘Complexity Methods for Predictive Synchronicity’ (Comet-PS), supported and funded by NWO (the Netherlands Organisation for Scientific Research), TKI-Dinalog (Top Consortium Knowledge and Innovation), Combi Terminal Twente (CTT) and the Early Research Program ‘Grip on Complexity’ of TNO (the Netherlands Organisation for Applied Scientific Research).

Contents

Part I Introduction

1	Categorisations of Optimisation Problems in Sychromodal Logistics	3
	Frank Phillipson	
2	Framework of Sychromodal Transportation Problems	17
	M. A. M. De Juncker, D. Huizing, and M. R. Ortega del Vecchyó	

Part II Solving MCMCF Problems

3	Deterministic Container-to-Mode Assignment	41
	D. Huizing	
4	Stochastic Container-to-Mode Assignment	59
	D. Huizing	
5	Deterministic Operational Freight Planning	89
	D. Huizing	
6	Alternative Performance Indicators for Optimising Container Assignment in a Sychromodal Transportation Network	119
	M. R. Ortega del Vecchyó	
7	Decision Making in a Dynamic Transportation Network: A Multi-Objective Approach	133
	M. R. Ortega del Vecchyó	
8	Reduction of Variables for Solving Logistic Flow Problems	143
	K. Kalicharan	
9	Cutting Planes for Solving Logistic Flow Problems	157
	K. Kalicharan	

Part III Synchronodal Logistics as Selfish Systems

10 Optimising Routing in an Agent-Centric Synchronodal Network with Shared Information..... 171
M. A. M. De Juncker

11 User Equilibrium in a Transportation Space-Time Network..... 187
L. A. M. Bruijns

12 Fair User Equilibrium in a Transportation Space-Time Network 201
L. A. M. Bruijns

Part IV Applications

13 Simulation Approach for Container Assignment under Uncertainty..... 219
W. J. de Koning

14 Optimising and Recognising 2-Stage Delivery Chains with Time Windows 235
F. Phillipson

15 Two-Step Approach for the Multi-Objective Container Assignment Problem with Barge Scheduling..... 251
F. Phillipson

16 A Robust Optimisation Approach to Synchronodal Container Transportation 263
I. Chiscop

Index..... 295

Acronyms

α -PFI	α -Pessimistic Future Iteration
AARC	Affinely adjustable robust counterpart
AIS	Automatic identification system
ARC	Adjustable robust counterpart
CC-heuristic	Compatibility clustering heuristic
CMCND	Capacitated multi-commodity network design
CTW	Controlled time window
CVRP	Capacitated vehicle routing problem
CVRP-TW	Capacitated vehicle routing problem with time windows
D	Demand elements
D2R	Demand-to-resource allocation
DARP	Dial-A-ride-problem
DD	Demand destination
DDD	Demand due date
DM	Decision maker
DO	Demand origin
DP	Demand penalty
DRD	Demand release date
DTA	Dynamic traffic assignment
DTN	Space time network
DV	Demand volume
EDT	Earliest delivery time
EFI	Expected future iteration
EFI	Expected future iteration
EPU	Earliest pick-up time
FIFO	First in first out
FPTAS	Fully polynomial-time approximation scheme
FTL	Full truck load
GG-heuristic	Greedy gain heuristic
GPS	Global positioning system
ILP	Integer linear programming problem

KPI	Key performance indicator
LDT	Latest delivery time
LP	Linear programming problem
LPU	Latest pick-up time
LSP	Logistic service provider
MCMCF	Minimum cost multi-commodity flow problem
MCND	Multi-commodity network design
MDP	Markov decision process
MILP	Mixed-integer linear programming problem
MIP	Mixed-integer programming
MPTW	Multi-period time window
OD	Origin-destination pair
OpEx	Operating expenditure
PTAS	Polynomial-time approximation scheme
R	Resource elements
RC	Resource capacity
RC	Robust counterpart
RD	Resource destination
RDT	Resource departure time
RO	Resource origin
ROP	Robust optimisation problem
RP	Resource price
RTT	Resource travel time
SO	System optimal
SOP	Stochastic optimisation problem
SPTW	Single period time window
TCP	Transmission control protocol
TEU	Twenty-foot equivalent unit
TH	Terminal handling time
UDP	User datagram protocol
UE	User equilibrium
VRP	Vehicle routing problem
VRP-PD	Vehicle routing problem with pickup and delivery
VRP-TW	Vehicle routing problem with time windows

Part I

Introduction

In this first part, we give an introduction to synchronodal logistics. In Chap. 1, a view is given on optimisation problems within this type of logistics. We can distinguish a number of categories in which these problems can be divided, which will sketch the global framework we use throughout this book. Also, a link to self-optimisation and complexity theory in logistic networks is given.

In Chap. 2, we dive deeper into the topic and give a broad literature overview on synchronodality and the relation to network design problems and multimodal and intermodal logistics. We introduce a general framework to classify synchronodal logistic optimisation problems that should help researchers and developers to find solution methodologies that are commonly used in their problem instance and to grasp characteristics of the models and cases in a compact way.

Chapter 1

Categorisations of Optimisation Problems in Sychromodal Logistics



Frank Phillipson

Abstract In this chapter, a view is given on optimisation of sychromodal transportation. For this, a framework is presented to distinguish four quadrants, based on local or global information available, combined with a local or global optimisation goal. We discuss how shifts can be made in this framework and how self-organisation can play a role in it. Next, a second way to distinguish between sychromodal planning problems is presented, based on the presence of uncertainty and the degree of freedom in service network design.

Introduction

In freight transportation logistics, there are various concepts. First there are multimodal and intermodal logistics. A freight network is called multimodal if the transportation of goods can be made via different modes, where a mode is a mean of transportation, such as a barge or truck. In an intermodal network, the goods are transported through a standardised unit of transportation, usually a container. In the last few years also, the concept of sychromodal transportation was introduced. Here the flexible deployment of modes, the possibility of continuous changes of the planning, and a central Logistic Service Provider (LSP), who offers integrated transport to its clients, are introduced.

The presence of such a central LSP suggests that there is a strong control of the system. However, even a big LSP only controls a small part of the total transportation system and might use parts of the transportation system that are out of his direct control. Next to this, the flexible deployment of modes in combination with the continuous changes is often placed in the direction of self-organisation. It is often assumed that it will be too complex, or complicated, for the LSP to control this system. In this chapter, we give some thoughts on the optimisation of

F. Phillipson (✉)
TNO, The Hague, The Netherlands
e-mail: frank.phillipson@tno.nl

a synchronodal transportation system and the role of the LSP in it. We will be touching the complexity of the system and the use and role of self-organisation in it. To elaborate on this, first we will introduce and use a framework that recognises four areas, based on the level of optimisation and the level of information within the logistic system. Next we look at a categorisation using the scope of the problem and the presence of uncertainty in it.

The organisation of this chapter is as follows. First, in section “[Context of Synchronodal Logistics](#)”, we introduce synchronodality. Then, in section “[Literature](#)”, we refer to some related chapters in the domain of modelling (complex) intermodal and synchronodal logistics, synchronodal optimisation opportunities, and self-organisation in logistics. In section “[Optimisation Framework](#)”, we sketch a framework for synchronodal transportation systems based on the level of information and the level of control or optimisation. How an LSP can move its system through this framework by adding information or control is described in section “[Changing Position in the Framework](#)”. In section “[Complexity and Self-Organisation](#)”, we elaborate on the complexity of, and the role of self-organisation in, such systems. Next, a second way to distinguish between synchronodal planning problems is presented, based on the presence of uncertainty and the degree of freedom in service network design. We end with some conclusions.

Context of Synchronodal Logistics

Freight transportation plays an essential role in supply chains by providing the efficient movement of feedstock, goods, and finished products between producers and consumers. In the European Union (EU) particularly, freight transport accounts for almost 4.5% of the gross domestic product (GDP), while the shipping carries 90% of the EU’s foreign trade [2]. However, freight transport also raises a number of issues such as pollutant emissions, noise, and congestion, which are mainly due to the road transport. A few figures illustrate this assertion. In 2014, about 49% of the total freight transportation in EU countries was done via road, 11.7% via rail, 4.3% via inland waterways, and 31.8% by sea¹ [13]. In terms of pollution, 72.9% of the greenhouse gas (GHG) emissions are due to road transport, 12.8% to maritime, and 0.5% due to railways [11]. To address both the issues of congestion and polluting emissions, a modal shift has become desirable [9]. In order to explain this concept, we will briefly review the existing transport modes.

Nowadays freight transport is mostly carried out using containers of standardised dimensions. These can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another (container ships, rail transport flatcars, and semi-trailer trucks) without being opened. The handling system is completely mechanised such that all handling is done with

¹ There is a certain amount of freight transport carried out by cargo aircrafts. However, this is not relevant for the scope of this chapter.

cranes and special forklift trucks. All containers have their own identification number and are tracked using computerised systems. These aspects make containers a preferable choice for goods transportation. The transportation chain of such containers is partitioned in three different segments [35]: *pre-haul* (first mile for the pickup process at the customer’s warehouse for instance), *long-haul* (transit of containers between different ports), and *end-haul* (last mile for the delivery process at the distribution centre). In most cases, the origin or destination of containers is located in the hinterland, and therefore, the pre-haul and end-haul transportation is carried out by road. For the long-haul, however, multiple transportation modes are available such as road, rail, and waterways. In this scenario, we distinguish several types of transportation whose terminology is well-established in the literature. We distinguish between unimodal transportation (transporting load by means of only one transportation mode) and multimodal transportation (using multiple modes). We further elaborate on different types of multimodal transportation (Table 1.1). In *intermodal* freight transportation, a load is transported from origin to destination in one transportation unit without handling the goods themselves when changing modes [35]. The three-segment container transport chain previously described is an example of intermodal transport. *Co-modal* transportation, as defined in [39], is the intelligent use of two or more modes of transport by a (group of) shipper(s) in a distribution system, either on their own or in combination, in order to obtain the best benefit from each mode, in terms of overall sustainability. Sychromodal freight transportation is the next step in terms of development, based on an efficient combination of intermodal and co-modal transportation. The Platform Sychromodality provides the following definition: “Sychromodality is the optimally flexible and sustainable deployment of different modes of transport in a network under the direction of a logistics service provider, so that the customer (shipper or forwarder) is offered an integrated solution for his (inland) transport.” [29]. Sychromodality emphasises the following aspects: the usage of various transport modes available in parallel to provide a flexible transport solution, the entrustment of the logistics service provider with the choice of transportation mode, and the possibility to switch in between transportation modes in real time [1].

Table 1.1 Intermodal, co-modal, and sychromodal transport [39]

Kind of transport	Multimodal transport (general term)	
Level of coordination shippers	Use of different modes in one transport from A to B	Use of different modes in a network
No operational logistics coordination between shippers: 1-to-1 link (chain) between user and provider of multimodal transport	Intermodal transport	Co-modal transport
Operational logistics coordination between shippers: many-to-many link (network) between users and providers of multimodal transport	Sychromodal transport	

In view of the existing types of transportation, the modal shift previously mentioned refers to reducing the number of containers transported by road in the long haul by dispatching them on barges or sea vessels in a smart and efficient way based on the cooperation of shippers. In other words, it is a transition from unimodal transport to either intermodal, co-modal, or synchromodal transportation, depending on the resources and the cooperation of the agents in the transportation network. The necessity of this shift has also been recognised by some port authorities [9]. In [30], the Port of Rotterdam Authority presented their goal to reduce the total number of containers transported by truck between the terminals in Rotterdam and inland destinations in North-West Europe from 55% in 2010 to 35% by 2035. For this purpose, a synchromodal network of rail and inland waterway connecting The Netherlands, Belgium, and Germany was initiated by a consortium led by the Europe Container Terminals (ECT) in Rotterdam [39]. The Extended Gate Services (EGS) network is based on the partnership between shipping lines and inland terminals [12]. The inland terminals of Amsterdam, Duisburg, Venlo, Moerdijk, and Willebroek act as virtual extensions of the Rotterdam-based deep sea terminal, in such a way that containers are trans-shipped in minimal time from the deep sea terminal in Rotterdam to the inland terminals.

Literature

In this section, we describe the main papers that give an overview of the problems in modelling (complex) intermodal and synchromodal logistics, the optimisation opportunities, the key factors needed for efficient transportation, and the first attempts on self-organisation in logistics.

Bestas and Crainic [8] describe the players in an intermodal network and the challenges that they face. They look at the shipper's perspectives on intermodal transportation, who has to decide on a certain transportation mode, and at the carrier's perspective, who has to provide an efficient and cost-effective service to the customer.

In [37], Tavasszy et al. give an introduction to synchromodality. The authors discuss the current position and evolution of intermodal transportation, the main elements of the synchromodal transport chain, and the innovations that are necessary to arrive at synchromodal transportation systems. Changes that they suggest that have to be made to the network in order to create a synchromodal system are, among others, the need for an integrated network and service design, an integrated operation and control, contracts that allow synchronised transport, a stronger collaboration, and a mind shift in planning and control. In the following sections, we will put this in a broader context.

The work by Riessen et al. [32] gives an overview of research opportunities in synchromodal container transportation in the case of the hinterland network of European Gateway Services. Their main topics are: optimisation of integral network

planning, methods for real-time decision-making, and the creation of flexibility in the network planning problem.

The paper by Pfoser et al. [28] determines the critical success factors of synchronomodality. They come up with a list of seven factors, which will be discussed later on in this chapter.

The papers [4–6, 14–16, 41] introduce self-organisation in the complex logistic networks, where logistics can be broader than only transportation and much of their focus is on supply networks and manufacturing. In this chapter, we combine some of the insights from their papers within in the domain of synchronomodal logistics.

Optimisation Framework

If we want to optimise a synchronomodal transportation system, we propose to look at the level of control in combination with the scope of the optimisation and at the information that is available to the LSP, or other decision makers, for making their decisions on modality choice or assignments. The first view is on the information aspects. Information can be available locally, where only (own) information about the direct neighbourhood is available, or globally, where information about the total system is available. The other view on synchronomodal transportation systems is the degree of control and optimisation. Here also a global and local view can be taken. There is a global view when everybody in the system tries to reach, if possible given the level of control, a global optimum. It is local when every decision maker is trying to optimise his own local goal.

In a simple view, as depicted in Fig. 1.1, this can be clustered into four quadrants:

- Limited: information local and optimisation local

Fig. 1.1 Optimisation framework used in this book



- Selfish: information global and optimisation local
- Cooperative: information local and optimisation global
- Social: information global and optimisation global

Each of these quadrants can be realised, and most of them can be found in practice and in the literature. In the literature, not much is written about the Limited case. Reason for this is that a Limited case is not novel, and from certain perspective, this case can be seen as Social, as explained further in section “[Changing Position in the Framework](#)”. An example of Selfish can be found in [38], and an example of Cooperative can be found in [20]. Examples of Social are plenty: [7, 17, 22, 23, 27, 33], and [43]. Most practical cases that are described are also Social: Case Rotterdam–Moerdijk–Tilburg, Synchronomodality, Case Synchronodal Control Tower, and Case Synchronodale Cool Port control [29]. Lean and Green Synchronodal [29] can be seen as a Selfish case.

Changing Position in the Framework

Not all positions in the framework of Fig. 1.1 are as rewarding for the LSP. A shift from one quadrant to another could be interesting. First note that in [28], seven critical success factors of synchronomodality are discussed:

1. Network, collaboration, and trust
2. Awareness and mental shift
3. Legal and political framework
4. Pricing/cost/service
5. ICT/ITS technologies
6. Sophisticated planning
7. Physical infrastructure

The question now is whether these factors can influence the position of the LSP in the framework, or, what changes in these factors are needed to make a shift between quadrants in the framework?

If we look at Fig. 1.1, most LSPs start in a “Limited position” from a macro view. This means, viewed from the outside, considering the whole logistic system. The LSP only uses his own information and tries to optimise his own business, not bothering (too) much about the world around him. However, the LSP might see this as a social case, as for the system that he controls, he has all the information and he has the total system under control. This was one of the reasons the Limited area is not described much in the literature; from the limited perspective of this LSP, this looks like a social environment. Think, for example, of an LSP that controls trucks, barges, and trains in his own network. In this network, the LSP acts as social. However, the used roads, railroads, ports, and other infrastructure are also used (and controlled) by other parties, making it a limited system from the macroview. When

the LSP wants to shift to a Selfish or Cooperative system, he has to add some of the critical success factors to the system.

In the shift from Limited to Selfish (vertical step), there is a need for global information. This starts by using the freely available information about traffic and other resources, offered by road or port authorities. This is a step that contains no natural barriers, and an LSP is expected to make this shift, where it will improve his information position and thus, in expectation, the quality of his decisions, reaching a better solution. This step requires a good information system from the authorities, ICT/ITS connections, and the ability to use the information (automatically) in his planning process. Further information, from the logistic chain, to use infrastructure or modalities from other commercial, competing parties, gives rise to the need for trust (to share information), awareness, mental shift, and, again, good technology and planning capability.

The step from Limited to Cooperative (horizontal) is less natural. This requires collaboration, trust, a legal framework, and good technology. Again, there is an expected gain for the total system due to the cooperation, or, given incentives, reaching a system optimal solution. However, sharing these benefits is a tricky one, as it requires a mental shift to receive the willingness of being controlled. Legal agreements and a lot of trust are needed. The expected gain is motivated by Roughgarden and Tardos [34]. They show that Selfish, here meaning locally optimising, systems have their price: they prove that travel times induced by selfish agents might be the same as the total travel time incurred by optimally routing twice as much traffic and indicate, as in [36], that adding central control or incentives gives an overall improvement of the system. However, in networks with high load, the performance might not suffer too much, as can be found in [26].

The shift from Selfish to Social (horizontal) asks the same or even more trust, collaboration, and sharing as the previous step. Here again all parties have to obey (one single) authority. Here also the sophisticated planning is needed and some mechanism to share the benefits of the total optimisation.

The shift from Cooperative to Social (vertical) is again expected to be easier where there will be a natural intention to gather available information to be used in the planning. Again, the sharing of information between the commercial parties and/or within the logistic chain is harder to organise.

We can conclude that the horizontal shifts are quite hard to make, where this asks the willingness of being controlled and the trust in sharing the benefits of the shift. The vertical shifts are ambivalent. It is a natural step to gather information to use in the own planning, so gathering the information available from road and port authorities and other open data sources is expected. Going a step further and sharing information within the logistic chain and between competitors will be harder. This gives rise to two additions to Fig. 1.2, first to add subclasses between the vertical classes, where public information is used. Next, we introduce a third dimension, the total wealth, that indicates the gain to be realised by making the shifts, motivated by [34]. This is shown in Fig. 1.3. Limited is expected to realise the lowest value, and then Selfish, Cooperative and Social the highest value.

Fig. 1.2 Enhanced framework

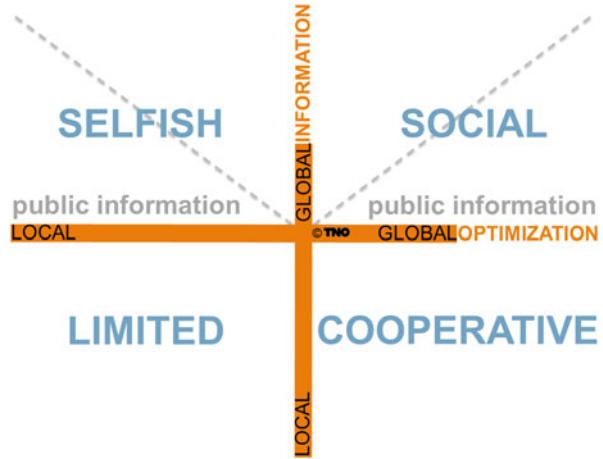


Fig. 1.3 Adding a third dimension: objective values



Complexity and Self-Organisation

Where in the case of the Social state a sophisticated planning is needed, from a mathematical point of view, this could be the easiest state. Information is available globally; the agents are controllable and obey some global authority. The Selfish and the Cooperative states, however, have properties that bring them in the context of complex adaptive systems. A description of complex adaptive systems is found in the work by Arthur et al. [3], who identify six properties that characterise any economy: dispersed interaction, the absence of a global controller, cross-cutting hierarchical organisation, continual adaptation, perpetual novelty, and far-from-equilibrium dynamics. However, where they speak of any economy, they point out that these features apply as well to any complex adaptive system [19].

In the Selfish system, the complexity is most obvious, where there is a lack of a global controller, dispersed interaction, and continual adaption of behaviour of the individual agents on the observed state of the system. Describing the system would be a first step to identify possible incentives to steer the system, perhaps unconsciously, in the direction of the Social system.

The Cooperative state looks less complex on first sight. Here we have local controllers who can take care of some level of organisation. However, the absence of global information will cause strong adaptive reactions on decisions of others, making it less predictable as a whole.

What role can self-organisation play here? The work in [5] indicates that “Minimal data requirements” and “Adaptivity” are expected in systems that use self-organisation, what makes both the Selfish and the Cooperative state a logical application area for Self-organisation. Self-organisation is also known under other terms such as autonomous cooperation and control, self-management, and self-regulation. This is defined by Windt and Hülsmann [40] as: “decentralised decision-making in hierarchical structures, presuming interacting elements in non-deterministic systems, which possess the capability and possibility to render decisions”. There is a trend in calling the logistic system a Physical Internet [24], and the comparison is made with the, apparent, self-organisation of the Internet. Then self-organisation is thought of as the solution for logistic systems, making it a goal in itself. The containers will flow through the Physical Internet as data packets through the Internet. But is the Internet really self-organising? Actually not, the data packets do not make a decision themselves. They are controlled by the routers that have some basic rules on routing schemes. Not really adaptive also, the packets do not interact, which are managed by the sender of the packets, where a TCP protocol, Transmission Control Protocol [31], waits for an acknowledgement and sends the packet again, when it takes too much time. Or, the UDP protocol, User Datagram Protocol [31], that sends it once, assuming that the arrival of a number of packets less will not be of impact on the experienced performance of the receiver. Not really comparable to the flow of containers. The Internet can be considered self-healing in some way, however. Within the framework of the previous section, the Internet can be placed in (as expected) the cooperative part. A single controller, router on the Internet, has no global view and takes decisions based on, mostly static, routing rules. The routing rules try to realise a global optimal solution.

This means that self-organisation and decentralised decision-making do not necessarily mean having smart, selfish, entities on the lowest level. Then the selfish behaviour will lead to poor overall network performance. It can mean smart decomposition, decentralising, or distributing of decision power, but keeping it as high in the hierarchy as possible, and not using more information than strictly necessary. This kind of self-organising networks will end, in Fig. 1.1, somewhere on the border between Social and Cooperative. It is important not to see self-organisation on the lowest hierarchical level as the ultimate goal. First the general goal has to be identified. If that goal is an adaptive, scalable, or robust transportation schedule, then control on the lowest level is not the only medicine, also control on higher levels, and robust, or disruption tolerant, planning can be useful.

Uncertainty and Scope of Optimisation

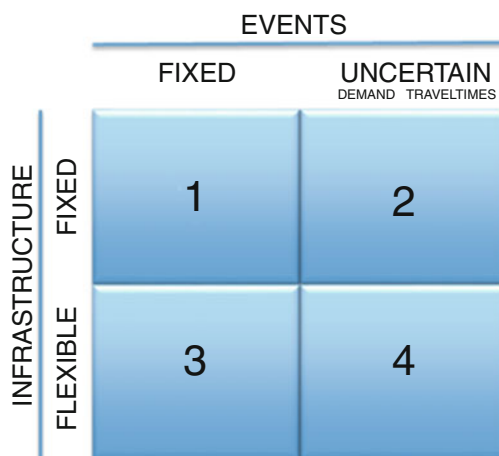
Next to level of optimisation, availability of information, and possible complexity, there are two other important aspects that make a logistic problem synchronomodal. These aspects are the presence of uncertainty and the scope of the optimisation.

The uncertainty can be in many parts of the logistic system, as shown in [21]. Uncertainty can be in demand, supply, or arrival of goods at the client, availability of resources and within the transportation process, think of travel times, failures in equipment, etc.

For the scope of the problem, both assignments of goods to modalities can be considered, as the operations of the vehicles (routes, departure times, etc.). The latter part is often known as service network design [10], which has a part that is not flexible at all, think of the location of (rail) roads and water ways, and more flexible parts such as timetables and routing. The latter part can also be taken into account at the (tactical) service network design phase, but especially in synchronomodal logistic problems, this is often taken into account during the (more) operational planning phase.

In Fig. 1.4, these elements are combined into 4 regions of problems. In the first region, the events or orders to be assigned are not uncertain (fixed) and the infrastructure (vehicles) has fixed schedules. These are common assignment or planning problems; examples can be found in [20, 22, 23, 38]. In problem 2, uncertainty or stochasticity is added, making it a, more complicated, problem of assignment or planning under uncertainty, as shown in [18, 42]. In the third problem both the orders and the infrastructure needs to be planned. This gives a high degree of freedom, resulting in a larger problem to be solved. Examples of this approach can be found in [7, 25, 33]. The fourth problem brings uncertainty to the third problem. This problem is discussed in [27, 43].

Fig. 1.4 Four types of problems



Conclusion

When optimising a sychromodal transportation system, we proposed to look at the level of control, in combination with the scope of the optimisation, and at the information that is available to the decision makers. This resulted in a framework with four main areas. We showed to expect that there is a natural drive to reach the top right area, where the total expected wealth will be maximal. However, allowing total control and sharing information will be a big hurdle to reach this state. Staying at other states, Cooperative or Selfish, some level of decentral decision-making is expected. We argued that self-organisation, meaning putting control at the lowest level, should not be the ultimate goal. Keeping the control as high as possible, smart decomposition, using the available information and robust planning, is expected to realise better results. A second way to classify problems in sychromodal logistics, we looked at uncertainty and the scope of the problem. Both ways of classifying will be used throughout this book, and we try to give an overview of various approaches to span both types of classifications.

References

1. Agbo, A. A., & Zhang, Y. (2017). Sustainable freight transport optimisation through sychromodal networks. *Cogent Engineering*, 4(1), 1421005.
2. Anonymous. (2016). EUROPA—EU transport policy. Retrieved Feb 7, 2018 from https://europa.eu/european-union/topics/transport_en
3. Arthur, W. B., Durlauf, S. N., & Lane, D. A. (1997). *The economy as an evolving complex system II* (Vol. 28). Addison-Wesley Reading.
4. Bartholdi, J. J., & Eisenstein, D. D. (2012) A self-coördinating bus route to resist bus bunching. *Transportation Research Part B: Methodological*, 46(4), 481–491.
5. Bartholdi, J. J., Eisenstein, D. D., & Lim, Y. F. (2010). Self-organizing logistics systems. *Annual Reviews in Control*, 34(1), 111–117.
6. Bartholdi III, J. J., & Eisenstein, D. D. (1996). A production line that balances itself. *Operations Research*, 44(1), 21–34.
7. Behdani, B., Fan, Y., Wiegmans, B., & Zuidwijk, R. (2014). Multimodal schedule design for sychromodal freight transport systems. *European Journal of Transport & Infrastructure Research*, 16(3), 424–444.
8. Bektas, T., & Crainic, T. (2007). A brief overview of intermodal transportation. CIRRELT.
9. Van den Berg, R., & De Langen, P. W. (2014). An exploratory analysis of the effects of modal split obligations in terminal concession contracts. *International Journal of Shipping and Transport Logistics*, 6(6), 571–592.
10. Crainic, T. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2), 272–288.
11. EU. Greenhouse gas emissions from transport. Retrieved Feb 7, 2018 from <https://www.eea.europa.eu/data-and-maps/indicators/transport-emissions-of-greenhouse-gases/transport-emissions-of-greenhouse-gases-10/>
12. European Gateway Services. (2022). Retrieved Feb 8, 2018 from <https://www.europeangatewayservices.com/>
13. Européenne, U., & Européenne, C. (2016). EU transport in figures 2016. Publications Office of the European Union, Luxembourg (2016). OCLC: 960914234

14. Hülsmann, M., Grapp, J., & Li, Y. (2008). Strategic adaptivity in global supply chains—competitive advantage by autonomous cooperation. *International Journal of Production Economics*, 114(1), 14–26.
15. Hülsmann, M., Kopfer, H., Cordes, P., & Bloos, M. (2009). Collaborative transportation planning in complex adaptive logistics systems: a complexity science-based analysis of decision-making problems of “groupage systems”. In *International Conference on Complex Sciences* (pp. 1160–1166). Springer.
16. Hülsmann, M., & Windt, K. (2007). *Understanding autonomous cooperation and control in logistics: The impact of autonomy on management, information, communication and material flow*. Springer.
17. Kooiman, K. (2015). A classification framework for time stamp stochastic assignment problems and an application to inland container shipping. TNO Internal Documentation.
18. Kooiman, K., Phillipson, F., & Sangers, A. (2016). Planning inland container shipping: A stochastic assignment problem. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications* (pp. 179–192). Springer.
19. Levin, S. A. (1998). Ecosystems and the biosphere as complex adaptive systems. *Ecosystems*, 1(5), 431–436.
20. Li, L., Negenborn, R. R., & De Schutter, B. (2017). Distributed model predictive control for cooperative synchromodal freight transport. *Transportation Research Part E*, 105, 240–260. <https://doi.org/10.1016/j.tre.2016.08.006>
21. Li, L., & Schulze, L. (2011). Uncertainty in logistics network design: A review. In *Proceedings of the International Multiconference of Engineers and Computer Scientists* (Vol. 2).
22. Lin, X., Negenborn, R. R., & Lodewijks, G. (2016). Towards quality-aware control of perishable goods in synchromodal transport networks. *IFAC-PapersOnLine*, 49(16), 132–137.
23. Mes, M., & Iacob, M. (2016). Synchromodal transport planning at a logistics service provider. In *Logistics and supply chain innovation* (pp. 23–36). Springer.
24. Montreuil, B. (2011). Towards a physical internet: Meeting the global logistics sustainability grand challenge. *Logistics Research*, 3(2–3), 71–87.
25. Nabais, J. L., Negenborn, R. R., Benitez, R. B. C., & Botto, M. A. (2013). A constrained MPC heuristic to achieve a desired transport modal split at intermodal hubs. In *2013 16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)* (pp. 714–719). IEEE.
26. Peeta, S., & Mahmassani, H. S. (1995). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, 60(1), 81–113.
27. Pérez Rivera, A., & Mes, M. (2016). Service and transfer selection for freights in a synchromodal network. *Lecture Notes in Computer Science*, 9855, 227–242.
28. Pfoser, S., Treiblmaier, H., & Schauer, O. (2016). Critical success factors of synchromodality: Results from a case study and literature review. *Transportation Research Procedia*, 14, 1463–1471.
29. PlatformSynchromodaliteit. (2017). Synchromodality. Retrieved Feb 7, 2018 from www.synchromodaliteit.nl/
30. Port of Rotterdam: Port Vision 2030. Retrieved Feb 6, 2018 from <https://www.portofrotterdam.com/sites/default/files/upload/Port-Vision/Port-Vision-2030.pdf>
31. Postel, J. (1980). User datagram protocol. Tech. rep., RFC.
32. Riessen, B. V., Negenborn, R. R., & Dekker, R. (2015). Synchromodal container transportation: An overview of current topics and research opportunities. In *International Conference on Computational Logistics* (pp. 386–397). Springer.
33. Riessen, B. V., Negenborn, R. R., Dekker, R., & Lodewijks, G. (2013). Service network design for an intermodal container network with flexible due dates/times and the possibility of using subcontracted transport. *International Journal of Shipping and Transport Logistics*, 7(4), 457–478.
34. Roughgarden, T., & Tardos, E. (2002). How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2), 236–259.

35. SteadieSeifi, M., Dellaert, N., Nuijten, W., Woensel, T.V., & Raoufi, R. (2014). Multi-modal freight transportation planning: A literature review. *European Journal of Operational Research*, 233(1), 1–15.
36. Swamy, C. (2007). The effectiveness of Stackelberg strategies and tolls for network congestion games. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1133–1142). Society for Industrial and Applied Mathematics.
37. Tavasszy, L., Behdani, B., & Konings, R. (2015). Intermodality and sychromodality. SSRN.com
38. Theys, C., Dullaert, W., & Notteboom, T. (2008). Analyzing cooperative networks in intermodal transportation: a game-theoretic approach. In *Nectar logistics and freight cluster meeting* (pp. 1–37). Delft, The Netherlands.
39. Verweij, K. (2011). *Synchronic modalities—critical success factors*. Logistics Handbook 2011.
40. Windt, K., & Hülsmann, M. (2007). Changing paradigms in logistics—understanding the shift from conventional control to autonomous cooperation and control. In *Understanding autonomous cooperation and control in logistics* (pp. 1–16). Springer.
41. Wycisk, C., McKelvey, B., & Hülsmann, M. (2008). “Smart parts” supply networks as complex adaptive systems: analysis and implications. *International Journal of Physical Distribution & Logistics Management*, 38(2), 108–125.
42. Xu, Y., Cao, C., Jia, B., & Zang, G. (2015). Model and algorithm for container allocation problem with random freight demands in sychromodal transportation. *Mathematical Problems in Engineering*, 2015 (2015). <https://doi.org/10.1155/2015/986152>
43. Zhang, M., & Pei, A. (2016). Sychromodal hinterland freight transport: Model study for the port of Rotterdam. *Journal of Transport Geography*, 52, 1–10.

Chapter 2

Framework of Sychromodal Transportation Problems



M. A. M. De Juncker, D. Huizing, and M. R. Ortega del Vecchyo

Abstract Problem statements and solution methods in mathematical sychromodal transportation problems depend greatly on a set of model choices for which no rule of thumb exists. In this chapter, a framework is introduced with which the model choices in sychromodal transportation problems can be classified, based on the literature. This framework should help researchers and developers to find solution methodologies that are commonly used in their problem instance and to grasp characteristics of the models and cases in a compact way, enabling easy classification, comparison, and insight in complexity. It is shown that this classification can help steer a modeller towards appropriate solution methods.

Introduction

The first introduction to sychromodal logistics was given in the previous chapter. Sychromodal freight transport is viewed here as intermodal freight transport with an increased focus on at least one of the following two aspects:

1. Transport planning is done using real-time data, allowing for online changes in the planning [21, 26, 31, 34].
2. Different parties share their real-time information, transportation resources, or transportation demands and may even entrust decisions to a central operator or logistics service provider (LSP). In some cases, clients may make an *a-modal* booking, agreeing with an LSP that their goods will be delivered at a set time and place against a set price and leaving it up to the LSP by what modes this is done [21, 26, 29, 34, 45].

Though other important developments exist within intermodal transport [41], sychromodality only concerns synchronising real-time data collection with real-time planning and synchronising the transportation flows and requirements among

M. A. M. De Juncker (✉) · D. Huizing · M. R. Ortega del Vecchyo
TNO, The Hague, The Netherlands

different parties. The goal of aspect 1 is to increase flexibility and reliability, that is to say, to become able to deal with disturbances in the system more effectively and to more effectively optimise against unknowns. The goal of aspect 2 is to increase efficiency and sustainability, by facilitating *full truck load consolidation* (FTL consolidation), in other words, letting one small order wait at a terminal so it can be combined with some other order [43]. Aspect 2 also facilitates smarter *equipment repositioning*, for example, by moving leftover empty containers directly to a nearby terminal where they are needed instead of through a depot [28].

Interest in synchronomodality has increased, due to improvements in data technology, an increased focus on the more complicated hinterland transport, and the ever-growing need for efficiency. However, synchronomodality faces several challenges that keep it from being adopted in practice. The challenges come from several sources. In [26], seven critical success factors of synchronomodality are discussed:

1. Network, collaboration, and trust
2. Awareness and mental shift
3. Legal and political framework
4. Pricing/cost/service
5. ICT/ITS technologies
6. Sophisticated planning
7. Physical infrastructure

Roughly, it can be argued that the first and second factors are mainly social problems, the third is a political problem, the fourth is a mathematical, social, and political problem, the fifth is a technological problem, the sixth is a mathematical problem, and the seventh is a technological and constructional problem.

Each of these factors is currently being addressed by different initiatives. Also in mathematics (applied in logistics) a lot of work has been done that can be used in synchronomodality. Mathematical planning problems are often divided into three main categories: strategical, tactical, and operational, so is the case with mathematical synchronomodal problems (Fig. 2.1). These problems are related in a pyramidal-like structure in the following sense: tactical problems are usually engaged where a specific strategical instance is given, and operational problems are frequently solved where strategical and tactical structures are fixed, although sometimes problems in two consecutive levels are solved simultaneously: for instance, in [2], the frequency of a resource is determined along with the flow of freight (that is, part of the schedules to resource and the freight to resources are solved at once).

Mathematical synchronomodal transportation problems on a tactical or operational level are usually represented via tools from graph theory and optimisation [33]. However, more often than not, the similarities end there: most of the models used to analyse a synchronomodal transportation network are targeted to a specific real problem of interest [33], and knowledge and methods of other branches such as statistics, stochastic processes, or systems and control are often used. The models emphasise on what is most important for the given circumstances. Consequently,

mathematical sychromodal transportation problems on a tactical or operational level have been engaged with approaches that may differ in many aspects:

- The exhaustiveness of the elements considered varies, e.g., weather or traffic conditions are considered in some models (such as the one presented in [17]) but not all.
- The elements that can be manipulated and controlled may vary, e.g., the departure time of some transportation means may be altered if suitable (as it happens in the model of [2]) or it may be that all transportation schedules are fixed.
- The amount of information relevant to the behaviour of the network may vary, and if a lack of information is considered, the way to model this situation may also vary [25].
- Whether some other stakeholders with authority in the network are in the model, and if so, how their behaviour is modelled.

A model is not necessarily improved by making it increasingly exhaustive. As it happens with most model making, accuracy comes with a trade-off, in this case, computational power. This computational burden is an intrinsic property of operational sychromodal problems [42] and one that is of the utmost importance given the real-time nature of operational problems: new information is constantly fed and it should be processed on time.

There is no rule of thumb for making the decisions above; also, each of the decisions mentioned above will shape the model and likely stir its solution methods to a specific direction. Though literature reviews of sychromodal transportation exist [33, 42], no generalised mathematical model for sychromodal transportation problems has been found yet, nor a way of categorising the existing literature by their modelling approaches. The framework for mathematical sychromodal transportation problems on a tactical or operational level presented in this chapter aims to capture the essential model-making decisions done in the model built to represent the problem. When no such model is specified, it shows the model-making decisions likely to be done in that case, which makes classification partly subjective. This is done in an attempt to grasp the characteristics of the model/case in a compact way, enabling easy classification and comparison between models and cases, as well as a way to see the complexity of a specific case at a glance. Also, it provides perspective to better relate new problems with previous ones, thus identifying used methodologies for the problem at hand.

In the remainder of this chapter, section “[Literature](#)” gives an overview of the relevant literature. Section “[Framework Identifiers and Elements](#)” introduces the classification framework, and section “[Notation](#)” two short-hand notations for this framework. In section “[Examples](#)”, some examples are provided. Based on these examples, common solution methods are mapped in section “[Solution Method Mapping](#)”, and the relationship with VRP terminology is discussed in section “[Relationship to VRP Terminology](#)”. In section “[Discussion](#)”, the examples are used to discuss strengths and weaknesses of the framework.

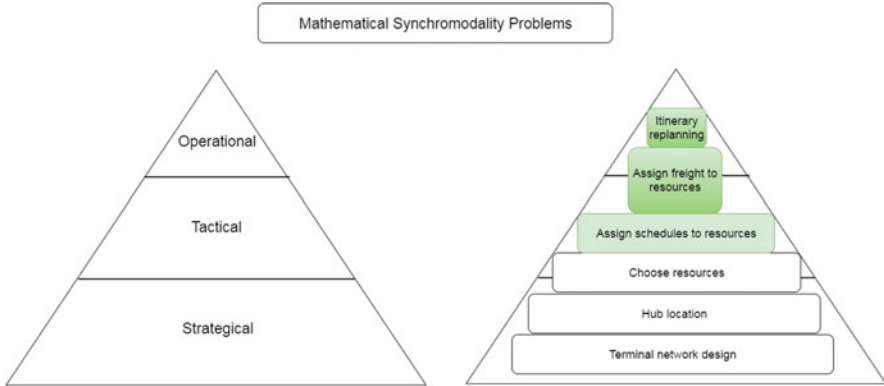


Fig. 2.1 Mathematical synchronodal problems

Literature

Synchronodal planning problems exist in both the tactical and operational area. The tactical planning problem is quite extensive. One needs to select and schedule the services to operate, allocate the capacity and equipment, and look at the routing of the goods. Together, this is also called *Service Network Design*. The review paper of Crainic [7] gives an extensive review of these problems, their formulations, and their solution frameworks. They also give a classification of these problems. In the literature, these problems are mostly modelled as *Fixed-Cost Capacitated Multicommodity Network Design Problems*. The paper by Min [23] develops a chance-constrained goal programming model that has multiple aspects in the objective function.

Papers in this area that explicitly deal with synchronodality are [2, 5, 30]. The paper by Puettmann and Stadler [30] mentions the importance of coordination of plans and operation of independent service providers in an intermodal transportation chain. They present a coordination scheme that will lead to reductions in overall transportation costs. They include stochastic demand in their calculation of the overall costs. Another paper by Caris, Macharis and Janssens [5] also looks at cooperation between inland terminals. In the paper, they develop a service network design model for intermodal barge transport and apply it to the hinterland network of the port of Antwerp. They simulate cooperation schemes to attain economies of scale. The paper by Behdani et al. [2] develops a mathematical model for a synchronodal service schedule on a single origin–destination corridor. Taking into account the frequency and capacity of different modalities, it determines the optimal schedule and timing of services for all transport modes. The assignment of containers to services is also determined by the model.

In operational planning problems, problems are regarded that deal with the day-to-day problems in a logistic network. This means that all these problems deal with

uncertainty and stochasticity, which make these problems complex. The decisions depend on the current information and an estimation of the future events. Issues here are:

- Reliability of a network: dealing with disruptions [6, 13, 22, 27] and resilience measures [6, 22]
- Resource management: empty unit repositioning problems [8–10] and allocation and positioning of the operating fleet [1, 32, 36–40]
- Replanning and online allocation [4, 11, 15]

Papers in the operational area within the sychromodal context are [21, 25, 45]. Zhang and Pel [45] developed a model that captures relevant dynamics in freight transport demand and supply, flexible multimodal routing with transfers, and transshipments. It consists of a demand generator (random sampling from historic data), an infrastructure and service network processor (which generates the resource schedule), a schedule-based assignment module (which assigns the demand to resources), and a performance evaluator. The model can be used to compare intermodal and sychromodal transportation from different perspectives: economic, social, and environmental. The authors use their model for a case study regarding the Rotterdam hinterland container transport, and they show that sychromodality will likely improve service level, capacity utilisation, and modal shift, but not reduce delivery cost.

The paper by Mes and Iacob [21] searches for the k -shortest paths through an intermodal network. They present a sychromodal planning algorithm that takes into account time windows, schedules for trains and barges, and closing times of hubs and minimises costs, delays, and CO₂ emissions. The k -shortest paths are then presented to a human planner, which can choose the best fitting path for an order by filtering these paths. Their approach consists of offline steps and online steps. In the offline steps, the network is reduced by eliminating paths that are too far from the route. In the online steps, an order is assigned to paths, by iterating over the number of main legs. A main leg in this chapter is a certain train or barge. The assumption they make is that a cost-efficient route consists of as few legs as possible. The online steps can be done after a disruption to make a new planning.

The paper by Rivera and Mes [25] looks at the problem of selecting services and transfers in a sychromodal network over a multi-period horizon. They take into account the fact that an order can be rerouted at any given moment. The orders become known gradually, but the planner has probabilistic knowledge about their arrival. The objective is to minimise expected costs over the entire horizon. They propose a *Markov Decision Process model* and a heuristic approach based on approximate dynamic programming.

Framework Identifiers and Elements

In this section, the framework is introduced. Within the framework, *demand* and *resources* are considered. In synchromodal transportation models, demand will likely be containers that need to be shipped from a certain origin to a destination. Resources can, for example, be: trucks, train, and barges. However, the framework allows for a broader interpretation of these terms. In repositioning problems, empty containers can be regarded as resources, where the demand items are bulks of cargo that need to be put in a container.

The framework has two main parts. The first part consists of the *identifiers*; these are specific questions one can answer about the model that depict the general structure of the model. The other is a list of *elements*; these elements are used to depict in more detail what the nature is of the different entities of the synchromodal transportation problem. Note that the notation presented does not include the optimisation objective. Within a specific model, there is of course an option to look at different optimisation objectives. This framework is developed in collaboration with multiple parties that study synchromodal systems. However, for certain specific problems, one might want to extend the framework. We think this is easily done in the same way as we set up the framework.

Identifiers

First we will elaborate on the identifiers of the framework. These identifiers are questions about the model. They identify the number of authorities, i.e., how many agents are in control of elements within the model. They will also identify the nature of different elements within the model. The list of elements will be discussed in detail in section “[Elements](#)”, but they are used to determine which components in the model are under control, which are fixed, which are dynamic, and which are stochastic. For instance, the departure time of a barge may be a control element, but it could also be fixed upfront, or modelled as stochastic. Some of the questions address how the information is shared between different agents and if the optimisation objective is aimed at global optimisation or local optimisation. All the answers on these questions together present an overview of the model, which can then be easily interpreted by others or compared to models from the literature.

The identifiers that describe the behaviour of the model in more detail are discussed below. Note that “resources” most often refer to transport vehicles and “demand items” most often refer to freight containers: however, demand items could also be empty containers with no specific destination in equipment repositioning problems. Therefore, a degree of generality is necessary in these identifiers:

1. *Are there other authorities (i.e., agents that make decisions)?*

Here it is identified if there is one global controller that steers all agents in the network or that there are multiple agents that make decisions on their own.

- *If there are other authorities, how is their behaviour modelled: One turn only, Equilibrium, or Isolated?*

If the previous question is answered with yes, i.e., there are multiple agents that make decisions, one needs to specify how these authorities react to each other. Three different ways for modelling the behaviour of multiple authorities in a sychromodal network are distinguished:

- *One turn only*: This means that each agent gets a turn to make a decision. After the decision is made, the agent will not switch again. For instance, in the case of three agents *A*, *B*, and *C*, agent *A* will first make a decision, then agent *B* and then agent *C*. The modelling ends here, since agent *A* will not differ from its first decision.
- *Equilibrium*: The difference between “one turn only” and “equilibrium” is that after each agent has decided, agents can alter their decision with this new knowledge. In the same example: agents *A*, *B*, and *C* make a decision, but then agent *A* changes its decision based on the decisions of *B* and *C*. If nobody wants to alter their decision anymore, the modelling ends and an equilibrium is reached between the specific agents.
- *Isolated*: If the behaviour of the multiple authorities is isolated, it means that from the perspective of one of the authorities only limited information is available about the decisions of the other agents. For instance: agent *C* needs to make a decision. It is not known what agents *A* and *B* have chosen or will choose, but agent *C* knows historic data on the decisions of agents *A* and *B*. Agent *C* can then use this information to make an educated guess on the behaviour of agents *A* and *B*.

2. *Is information within the network: global or local?*

This identifies if the information within the network is available globally or locally. If the information is locally available, it means that only the agents themselves know for example where they are or what their status is at a certain time. If the information is global, the network operator and/or all other agents know all this information as well.

3. *Is the optimisation objective: global or local?*

The same can hold for the optimisation objective. If all agents need to be individually optimised, the optimisation objective is local. If the optimisation objective is global, we want the best alternative for the entire network.

4. *Which elements do you control?*

Since we want to model a decision problem, at least one element of the system must be in control and must take decisions. For example: if one wants to model which containers will be transported by a certain mode in a sychromodal network, we have control over the *demand-to-resource allocation*. If we want to model which trains will depart on which time at certain locations, we have control of the *resource departure time*. An extensive list of elements is given in section “[Elements](#)”.

Of course the controllable element can have constraints: for instance, we can influence the departure times of trains, but they cannot depart before a certain

time in the morning. This is still a controllable element. We thus consider an element a controllable element if a certain part of it can be controlled.

5. *What is the nature of the other elements (fixed, dynamic, stochastic, or irrelevant)?*

The other elements within the network can also have different behaviours. We distinguish four:

- *Fixed*: A fixed element does not change within the scope of the problem.
- *Dynamic*: A dynamic element might change over time or due to a change in the state of the system (e.g., the amount of containers changes the travel time), but this change is known or computable beforehand.
- *Stochastic*: A stochastic element is not necessarily known beforehand. For instance, it is not known when orders will arrive, but it is a Poisson process. It might also occur that the time the order is placed is known, but the amount of containers for a certain order follows a normal distribution.
- *Irrelevant*: The list we propose in section “[Elements](#)” is quite extensive. It might occur that for certain problems not all elements are taken into consideration to model the system. Then these elements are irrelevant,

6. *What is the optimisation objective?*

This identifier is for the optimisation objective. One can look at the exact same system but still want to minimise a different function. One could think of travel times and CO₂ emissions. It is also possible to identify a much more specific optimisation objective. Examples of optimisation objectives are in section “[Examples](#)”.

Elements

Having defined the identifiers of the framework, now a list of elements is presented, which are expected to exist in most synchronodal transportation problems. They are divided into two parts: *resource elements* and *demand elements*. The resource elements are all elements related to the resources, which are mostly barges, trains, and trucks. However, for compactness, we also view a terminal as a resource. The demand elements are all elements related to the demand, which are most of the time freight or empty containers. Most elements mentioned in this list are straightforward, and small clarifications are mentioned where necessary:

- Resource elements:
 - *Resource Type*: Different modalities can be modelled as different resource types. Another way to use this element is for owned and subcontracted resources.
 - *Resource Features*: These features can be appointed to the different resource types or can have the same nature for the different types. For instance, it may be that there are barges and trains in the problem, but their schedules are both

fixed, thus making the nature of the resource features *fixed* for both resource types:

- *Resource Origin (RO)*.
 - *Resource Destination (RD)*.
 - *Resource Capacity (RC)*: Indication of how much demand the different resources can handle.
 - *Resource Departure Time (RDT)*.
 - *Resource Travel Time (RTT)*: Time it takes to travel from the origin to the destination (in the case of a moving resource).
 - *Resource Price (RP)*: This can be per barge/train/truck/... or per container.
- *Terminal Handling time (TH)*: Time it takes to handle the different types of modes at the terminal. This can again be per barge/train/truck/... or per container.
- Demand elements:
 - *Demand Type*: One can also think of different types of demand. For instance, larger and smaller containers or bulk.
 - *Demand-to-Resource Allocation (D2R)*: The assignment of the demand to the resources.
 - *Demand Features*:
 - *Demand Origin (DO)*.
 - *Demand Destination (DD)*.
 - *Demand Volume (DV)*: It might be that different customers have different amount of containers that are being transported. (Note that the demand element in this case will always be 1 container, since each container can have its own assignment.)
 - *Demand Release Date (DRD)*: The release date is the date at which the container is available for transportation.
 - *Demand Due Date (DDD)*: Latest date that the container should be at its destination, which is not necessarily a hard deadline.
 - *Demand Penalty (DP)*: Costs that are incurred when the due date is not met or when the container is transported before the release date (this is sometimes possible with coordination with the customers).

Notation

In this section, two types of notations are introduced, which will make it easier to quickly compare different models. Obviously, it is hard to make a compact notation and still incorporate all aspects of a sychromodal system. Therefore, the notation was made as compact as possible, and some of the details are left out. When

comparing models in detail, it is easier to look at all answers to the identifiers mentioned in section “[Identifiers](#)”. Our six-field notation was built to resemble Kendall’s notation for classification of queue types [14] and the notation of theoretic scheduling problems proposed by Graham, Lawler, Lenstra and Rinnooy Kan [12].

Six-Field Notation

A synchronodal transportation model can be described by the notation:

$$C|S|D|I|Y|B.$$

The letters denote the following things:

- *C*: controlled elements
- *S*: stochastic elements
- *D*: dynamic elements
- *I*: irrelevant elements
- *Y*: system characteristics
- *B*: behaviour of other authorities, if any

The first four entries in the notation can be filled with all elements mentioned in the list in section “[Elements](#)”. If any of the elements is not mentioned in these four fields, it is assumed to be fixed. If all unmentioned resource elements should default to stochastic instead, an *R* can be written in the second field: the same goes for defaulting to controlled, dynamic, or irrelevant elements. Analogously, a *D* can be written in any of the first four fields to set a default for the demand elements.

For the system characteristics, a notation is proposed that gives an answer to questions 1, 2, and 3 of the identifiers. Thus: are there other authorities, is the information global or local, and is optimisation global or local? The notation is based on the categorisation of the previous chapter. In a similar way, the four options for the field *system characteristics* in the notation are:

- *Selfish*: information global and optimisation local
- *Social*: information global and optimisation global
- *Cooperative*: information local and optimisation global
- *Limited*: information local and optimisation local

The four options for the final field are *one turn only*, *equilibrium*, *isolated*, and *I*: the first three are explained in section “[Identifiers](#)”, and the final option denotes that there are no other decision-making authorities in the system.

Two-Column Notation

Though the proposed six-field notation is a relatively compact way to describe a complex system, it comes with two downsides: it requires a degree of memorisation, and if new natures other than controlled, fixed, stochastic, dynamic, or irrelevant are distinguished, there is no place for this in the current notation. These problems are solved by using the two-column notation described in this section, at the cost of compactness.

A synchronodal transportation model can also be described by the notation:

Controlled elements	C , written out
Fixed elements	Fixed elements, written out
Stochastic elements	S , written out
Dynamic elements	D , written out
Irrelevant elements	I , written out
System characteristics	Y
Behaviour of other authorities	B

If there are no stochastic elements in a problem, that row can be left out: the same goes for the other natures. If a new nature is distinguished, a row can be easily added for this. In the six-field notation, any unmentioned element was considered fixed, unless an R or D was placed in one of the fields to set the default to that nature. This is again possible here: an R and a D should always be placed in one of the rows to set the default nature of the resource elements and demand elements, respectively.

On the Two Notations

In neither notation, the optimisation objective is included: these are considered to be too distinct among different problems to merit classification. As discussed earlier, the two-column notation is much less compact than the six-field notation but requires less memorisation and lends itself better to change when new natures are distinguished. Our advice is to employ the two-column notation at first, but to switch to the six-field notation when the framework starts gaining familiarity: this familiarity should make the memorisation easier, and this adoption time should suffice to discover any truly important new natures. This chapter will largely use the six-field notation for the sake of compactness, seeing how reminders are readily available within this chapter.

Examples

As discussed earlier, one of the ideas of the framework is that, when starting work on a new problem, one can first classify the assumptions this model would need and then investigate papers that have similar classification. Therefore, a number of classification examples are presented for both the existing models and the new problems. First, we answer the framework questions for the Kooiman pickup case [15] in Table 2.2 and show how this can be written in our compressed notation. Afterwards, Table 2.3 shows compressed notation of some other problems described in papers, such that the interested reader can study more examples of our framework classification. Then, using Table 2.4, we examine some real-life cases and classify how we would choose to model these problems. To clarify: these problems do not yet have an explicitly described model, so this classification is based on how we would approach and model these practical problems, but other modellers may make other modelling decisions. Finally, the given examples will be used as input for discussion. In the Kooiman pickup case [15], a barge makes a round trip along terminals in a fixed schedule to pick up containers to bring back to the main terminal; however, the arrival times of the containers at the terminals are stochastic. At each terminal, a decision has to be made of how many containers to load onto the barge, and a guess has to be made of how much capacity will be needed for later terminals, all while minimising the amount of late containers. The actual time of residing at the terminal is disregarded. We refer to Table 2.2 for the answering of the framework questions. We refer to Table 2.1 for a reminder of the framework element abbreviations.

Note that only barges are taken into consideration as resources, not trucks. It would have been possible to describe trucks as resources as well, but we have chosen to classify these as part of the lateness penalty, because there is no decision-making in how the trucks are used. Also, it may seem strange to speak of global or local information and optimisation when there are no other decision-making authorities. The information is considered global because the only decision-making authority knows “everything” that happens in the network; the optimisation is considered

Table 2.1 Abbreviations of the framework elements used in the compressed notation

<i>R</i> : Unmentioned resource elements	<i>D</i> : Unmentioned demand elements
<i>RO</i> : Resource origin	<i>DO</i> : Demand origin
<i>RD</i> : Resource destination	<i>DD</i> : Demand destination
<i>RC</i> : Resource capacity	<i>DV</i> : Demand volume
<i>RDT</i> : Resource departure time	<i>DRD</i> : Demand release date
<i>RTT</i> : Resource travel time	<i>DDD</i> : Demand due date
<i>RP</i> : Resource price	<i>DP</i> : Demand penalty
<i>TH</i> : Terminal handling time	<i>D2R</i> : Demand-to-resource allocation

Table 2.2 The framework applied on the Kooiman pickup case [15]

Other authorities	No
Information global/local	Global
Optimisation global/local	Global
Resource elements	Resource type: barges Controlled resource elements: none Resource features: fixed, except TH (irrelevant)
Demand elements	Demand type: freight containers Controlled demand elements: $D2R$ Demand features: fixed, except DRD (stochastic)
Optimisation objective	Maximal percentage of containers that travel by barge instead of truck

global because the decision-maker wants to optimise the performance over all demand in the network put together, not over some individual piece or pieces of freight.

Using the six-field notation, most of Table 2.2 can be summarised as follows:

$$D2R|DRD| \cdot |TH|social|1.$$

It could also be represented in the two-column notation, as follows:

Controlled elements	Demand-to-resource allocation
Fixed elements	R, D
Stochastic elements	Demand release date
Irrelevant elements	Terminal handling time
System characteristics	Social
Behaviour of other authorities	1

Here, the row for dynamic elements can be left out because the problem has no dynamic elements, and R and D are written in the row for fixed elements to indicate that any unmentioned resource element and any unmentioned demand element are fixed by default.

Only the optimisation objective and type specifications are lost in this process. In Table 2.3, we apply the framework to more problems from academic papers. In this table, we include the optimisation objective to illustrate the wide range of optimisation possibilities. It is not actually necessary to describe the optimisation objective when using the compressed problem notation. In some cases, especially practical problem descriptions, optimisation objectives may not yet be explicitly known. Therefore, Table 2.4 leaves them out. In that table, we review some practical problem descriptions and apply the framework to them.

Table 2.3 Selected papers in the synchromodal framework

Behdani [2]: $D2R, RDT \cdot \cdot \cdot social 1$
Objective: minimal transportation costs and waiting penalties
Kooiman [15]: $D2R DRD \cdot TH social 1$
Objective: maximal percentage of containers by barge instead of truck
Le Li [18]: $D2R \cdot DV RDT, DRD, DDD cooperative equilibrium$
Objective: with self-optimising subnetworks, total minimal cost in union
Lin [20]: $D2R \cdot RC RP social 1$
Objective: minimal total quality loss of perishable goods
Mes [21]: $D2R \cdot RP RC social 1$
Objective: best modality paths against different balances of objectives
Nabais [24]: $D2R \cdot RC, RTT, RP, DV, DP TH social 1$
Objective: sustainable transport modality split that retains client satisfaction
van Riessen [31]: $D2R, RDT \cdot RC, RTT, RP, TH, DP \cdot social 1$
Objective: minimise transport and transfer cost, penalty for late delivery and cost of use of owned transportation
Rivera [25]: $D2R D R \cdot social 1$
Objective: minimal expected transportation costs
Theys [35]: $RP, D2R, DP \cdot \cdot RDT, DRD, DDD selfish equilibrium$
Objective: fairest allocation of individual costs
Xu [44]: $D2R, RC RP, DV, DP \cdot RDT, RTT, TH, DRD, DDD social 1$
Objective: maximised expected profit during tactical planning
Zhang [45]: $D2R D \cdot \cdot social 1$
Objective: maximised balance of governmental goals

Table 2.4 Selected use cases in the synchromodal framework

Lean and Green Synchromodal [28]: $D2R \cdot \cdot \cdot selfish 1$
Rotterdam–Moerdijk–Tilburg [28]: $D2R RTT, TH \cdot \cdot social 1$
Synchromodality [28]: $D2R, RDT D \cdot \cdot social 1$
Synchromodal Control Tower [28]: $D2R, RC, DV RP, RTT, TH \cdot \cdot social 1$
Synchromodale Cool Port control [28]: $D2R, RDT RTT DDD, DP \cdot social 1$

Another example we reviewed is the modelling of an agent-centric synchromodal network. Here all agents want to be at their destination as fast as possible, but everyone does share the information about where they are and where they are going with everybody else in the network. Table 2.5 shows the answer on the questions of the framework. In the short notation, this problem is:

$$D2R|D \cdot |DP|selfish|equilibrium.$$

Table 2.5 The framework for an agent-centric sychromodal network

Other authorities	Yes
Information global/local	Global
Optimisation global/local	Local
Resource elements	Resource type: barges, trains, and trucks
	Controlled resource elements: none
	Resource features: fixed
Demand elements	Demand type: containers
	Controlled demand elements: $D2R$
	Demand features: stochastic, except DP (irrelevant)
Optimisation objective	Minimise travel times

Solution Method Mapping

In the previous section, a number of papers on sychromodal transport problems and solution methods were studied. Some of the choices in solution methods are similar between papers and can be partially recognised from their framework notation. Here, we group the papers on solution method with remarks on complexity issues and insightful framework similarities:

- *Shortest path algorithms*: In [21], $D2R$ is to be performed under the absence of capacity constraints. Mes et al. rightfully note that, in the absence of capacity constraints, the best modality paths can be found simply by using shortest path algorithms, which are known to run in polynomial time in the input size. Whenever capacity is included, this brings computational difficulties, as dividing flow over capacitated arcs is related to the NP-hard multi-knapsack problem. In [45], this is handled by a sequential shortest path algorithm: whenever a demand item comes in, assign it to the cheapest path with remaining capacity and repeat this until everything is assigned. Though this, is an efficient method, one can imagine it yielding sub-optimal results, especially under the stochastic release dates. However, if $D2R$ is the only control element, a sequential shortest path algorithm is recognised as a computationally efficient option: in the absence of capacity constraints, stochastic elements, and control-based dynamic elements, it is likely to yield the optimal solution.
- *Two-stage stochastic programming*: In [44], $D2R$ must again be performed. RC is technically a control element as well, but the challenge lies mainly in the $D2R$ control. Now, the stochasticity is dealt with by means of two-stage stochastic programming. The studied model may lend itself well to stochastic programming because no intermediary nodes are recognised between the one origin and the set of destinations. Even so, Xu et al. propose a meta-heuristic to deal with the computational intensity incurred by large sets of freight types, destinations, transportation modes, or scenarios.

- *Approximate dynamic programming*: In [15] and [25], Markov decision process models are presented but argued to be too computationally expensive. Instead, they solve $D2R$ with stochastic elements by making tentative decisions, simulating the potential results of this decision and their incurred costs, and then taking the tentative decision with the lowest simulated expected cost. This is recognised as a computationally reasonable alternative to solving $D2R$ with stochastic elements.
- *Systems and control theory*: In [18], a cooperative $D2R$ equilibrium problem is studied rather than a social problem without other authorities. In [20] and [24], $D2R$ is performed, while dynamic elements play an important role. Finding a good equilibrium with the other authorities, or settling on a good equilibrium between the control elements and the dynamic parameters that depend on control, is understandably modelled using systems and control theory. In two out of these three papers, model predictive control is employed. However, the similarities between these three papers could also be explained by their shared authors.
- *Multi-control integer linear programming*: In both [2] and [31], not only $D2R$ is controlled, but RDT as well, as a form of partial resource schedule control. Both papers resort to using integer linear programs to find an optimal solution. As many of the variables in these programs are indexed on three sets, these methods are expected to scale poorly to larger instances. Efficient solution methods to problems where not only $D2R$ is controlled but the resource schedules as well appear to be an open problem: though the Vehicle Routing Problem (VRP) comes to mind, section “[Relationship to VRP Terminology](#)” will address the challenges that synchronomodality introduce to the VRP.
- *Game theory*: In [35], fair pricing must be determined in a system with selfish decision-makers. Understandably, steering this selfish behaviour is attempted by using game theory. Theys et al. note that the proposed techniques work for limited systems, but that moderately advanced synchronomodal systems require advanced game-theoretical techniques.

One could put this the other way round and wonder, given a problem classification, what solution methods could be suitable, and what complexity issues arise. To this, we give the following answer. Selfish problems have been investigated with game theory, but only moderately advanced synchronomodal systems already seem to require advanced game theory. Cooperative problems have been studied using model predictive control, for which commercial solvers exist. Social $D2R$ problems could be solved using sequential shortest path algorithms. These are efficient methods, but only optimal under the absence of capacity constraints, stochasticity, and control-based dynamic elements. Under the presence of capacity constraints, $D2R$ problems are likely to be NP-hard due to their similarity to the multi-knapsack problem. To solve $D2R$ with stochastic elements, two-stage stochastic programming and Markov Decision Processes have been examined but proposed to be computationally too expensive. Approximate dynamic programming and Xu’s meta-heuristic are proposed as efficient alternatives. To solve $D2R$ with dynamic elements, model predictive control and other systems and control theory techniques are proposed.

To solve social $D2R$ and RDT simultaneously, only large-scale integer linear programs have been proposed in the examined literature.

This is far from a complete mapping from framework classification to solution method. Components that are not described by the framework may be critical to the viability of a solution method, like the absence of intermediary locations in [44] facilitating two-stage stochastic programming. However, we believe that worthwhile relationships have been and can be drawn between framework classifications and potential solution methods.

Relationship to VRP Terminology

When optimising the transport of freight using several vehicles, thus simultaneously determining $D2R$ and resource schedules, the vehicle routing problem (VRP) immediately comes to mind. The VRP is a widely studied transport problem. In a sense, a framework for the classification of different VRP variants exists in the form of consensus: the Capacitated Vehicle Routing Problem (CVRP), the Vehicle Routing Problem with Pickup and Delivery (VRPPD), the Vehicle Routing Problem with Time Windows (VRPTW), subvariants, and combinations of these variants are well-known and their definitions largely agreed upon [16, 19]. However, none of the papers investigated in section “[Examples](#)” seem to involve themselves explicitly with VRP models. This can be explained and recognised by applying the developed framework on VRP variants.

The VRP, in its most classical sense, is the problem of minimising transport costs when dispatching m vehicles from some depot node to service all other nodes exactly once. A synchromodal version of this is quite imaginable. The real-time flexibility aspect of synchromodality would mean that re-evaluations may occur where the vehicles “start” at their current destination but must still return to the depot, and the already visited nodes are taken out of the problem. The information sharing aspect of synchromodality can be assumed to already be part of the problem: the resources and demands can be assumed to be pooled from several parties and put under the control of a central operator. Under these minor assumptions, the synchromodal VRP lends itself to the following classification:

$$D2R, RD| \cdot | \cdot | RC, RDT, RTT, TH, DV, DRD, DDD, DP|social|1.$$

The decision-maker must simultaneously decide which service nodes are visited by which vehicle and in which order. Time and capacity constraints are not present, and all related elements are irrelevant. Only the total “price” of these routes is minimised: though this price may equal the travel time, the actual element of time does not influence the decision space, as long as release time, due times, and time windows are absent. When adding vehicle capacities, the RC and DV become fixed

rather than irrelevant, so the synchronodal CVRP is denoted by

$$D2R, RD | \cdot | \cdot | RDT, RTT, TH, DRD, DDD, DP |_{social} | 1.$$

When time windows are added, the RDT becomes a control element, and the RTT , DRD , DDD , DP , and sometimes the TH become relevant. Note that soft and hard time windows are not necessarily classified differently: the demand penalty could be an arbitrarily high constant to simulate hard deadlines, but soft due dates may also come with fixed penalties that are not arbitrarily high. As such, the synchronodal Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) could be classified as, depending on whether or not terminal handling times are observed,

$$D2R, RD, RDT | \cdot | \cdot | TH |_{social} | 1 \quad \text{or} \quad D2R, RD, RDT | \cdot | \cdot | \cdot |_{social} | 1.$$

If separate pickup and delivery locations are specified, this would still mean that each demand item has a fixed DO and DD , so the Capacitated Vehicle Routing Problem with Time Windows and Pickup and Delivery (CVRPTWPD) would be classified the same way as the CVRPTW.

One of the most important differences between synchronodal VRP variants and the problems examined in section “[Examples](#)” is laid bare by the framework notation: all synchronodal VRP variants have the resource destination as a control element, while none of the studied papers do. In fact, having the RD as a control element is largely synonymous with having the responsibility of routing.

While this definitely helps in recognising the absence of vehicle routing in the studied papers, it does not yet explain it. The following explanations for the absence of vehicle routing in the studied papers are proposed:

- Papers with more control elements than just $D2R$ tend to resort to using large ILPs, making inclusion of the RD as a control element computationally challenging.
- In many of the papers, the routes were already predetermined in a strategic/tactical phase, and only the day-by-day assignment remained as a problem on the operational level, possibly due to this computational intensity and the real-world implications of planning vehicles routes.
- Most Multiple Travelling Salesman Problem (mTSP)-based models, including most VRP variants, do not lend themselves to the concept of intermodality, thus synchronodality: while intermodal transport encourages that different vehicles take care of different parts of a container’s journey, most mTSP-based models encourage that the entire voyage of one container is taken care of by one vehicle only [3].

We conclude that the class of synchronodal transport problems differs significantly from the classical VRP variants: as such, they require a classification scheme of their own.

Discussion

The examples in section “[Examples](#)” show some strengths and limitations of the classification framework, which are discussed in this section.

One of the goals of this framework was to offer guidance when tackling a new problem: as an example, if the problem from the sychromodality [28] case is modelled in a non-stochastic way, we can now see that it may be worthwhile to study the solution method presented by Behdani [2], because they then have a very similar compressed framework classification: in particular, the sychromodality case involves the same control elements. If such a record is kept of papers and models, this could greatly improve the efficiency of developments in sychromodal transport. This would fulfil the second goal of the framework: to collect literature on sychromodal transportation within a meaningful order.

The final goal of this framework was to expose and compare relationships between seemingly different problems: for example, we can now see that the problems described by Le Li [18] and Theys [35] have similarities, in that they investigate negotiation between parties and do not focus on timeliness of deliveries. Similarly, we can see that the model assumption Mes [21] makes in disregarding resource capacity is an uncommon decision. In section “[Solution Method Mapping](#)”, it was argued that such similarities and dissimilarities can help explain the effectiveness of certain solution methods.

In the sychromodality case [28], our interpretation of the problem implies that the demand features are stochastic. However, the problem could also be approached in a deterministic way, depending on choices that the modeller and contractor make based on the scope of the problem, the requirements on the solution, and the available information. This shows the most important limitation of the classification framework: what classification to assign to a problem or model remains dependent on modelling choices, as well as interpretation of problem descriptions. Even without the framework, however, modelling choices will always introduce subjective elements into how a real-world problem is solved. This framework can be used to consistently communicate these underlying model assumptions.

A second limitation of the framework is that, because of the large amount of elements described in it, two similar problems are relatively unlikely to fall in the exact same space in the framework because of their minor differences. Therefore, one should not only look for problems with the exact same classification, but also problems with a classification that is only slightly different. In a more general sense, solution methods may apply to far more than one of these very specific framework classes. If two problems have the exact same controlled elements, it is imaginable that their models and solution methodologies may largely apply to the other. As a point of future research, it could be interesting to further investigate which classification similarities are likely to imply solution similarities, which may also be a steppingstone towards a general solution methodology.

As a final limitation, the compressed notation does not reveal that the paper by Lin [20] and the “Sychromodale Cool Port control” [28] case both focus on

perishable goods. This shared focus is not only cosmetic: mathematically, it may imply objective functions and constraints not focused on in other cases. To combat this limitation, we advise anyone using the framework to offer both a compressed and an extended description of their problem or model.

References

1. Bandeira, D., Becker, J., & Borenstein, D. (2009). A DSS for integrated distribution of empty and full containers. *Decision Support Systems*, 47(4), 383–397.
2. Behdani, B., Fan, Y., Wiegmans, B., & Zuidwijk, R. (2016). Multimodal schedule design for synchromodal freight transport systems. *European Journal of Transport & Infrastructure Research*, 16(3), 424–444.
3. Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3), 209–219.
4. Bock, S. (2010). Real-time control of freight forwarder transportation networks by integrating multimodal transport chains. *European Journal of Operational Research*, 200(3), 733–746.
5. Caris, A., Macharis, C., & Janssens, G. (2012). Corridor network design in hinterland transportation systems. *Flexible Services and Manufacturing Journal*, 24(3), 294–319.
6. Chen, L., & Miller-Hooks, E. (2012). Resilience: An indicator of recovery capability in intermodal freight transport. *Transportation Science*, 46(1), 109–123.
7. Crainic, T. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2), 272–288.
8. Crainic, T., Gendreau, M., & Dejax, P. (1993). Dynamic and stochastic models for the allocation of empty containers. *Operations Research*, 41(1), 102–126.
9. Di Francesco, M., Lai, M., & Zuddas, P. (2013). Maritime repositioning of empty containers under uncertain port disruptions. *Computers & Industrial Engineering*, 64(3), 827–837.
10. Erera, A., Morales, J., & Savelsbergh, M. (2005). Global intermodal tank container management for the chemical industry. *Transportation Research Part E: Logistics and Transportation Review*, 41(6), 551–566.
11. Goel, A. (2010). The value of in-transit visibility for supply chains with multiple modes of transport. *International Journal of Logistics: Research and Applications*, 13(6), 475–492.
12. Graham, R., Lawler, E., Lenstra, J., & Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5, 287–326.
13. Huang, M., Hu, X., & Zhang, L. (2011). A decision method for disruption management problems in intermodal freight transport. In *Intelligent decision technologies* (pp. 13–21). Springer.
14. Kendall, D. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. *The Annals of Mathematical Statistics*, 24(3), 338–354.
15. Kooiman, K., Phillipson, F., & Sangers, A. (2016). Planning inland container shipping: A stochastic assignment problem. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications* (pp. 179–192). Springer.
16. Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3), 345–358.
17. Li, L. (2016). Coordinated model predictive control of synchromodal freight transport systems. Ph.D. Thesis, Delft University of Technology TRAIL thesis series.
18. Li, L., Negenborn, R. R., & De Schutter, B. (2017). Distributed model predictive control for cooperative synchromodal freight transport. *Transport. Res. Part E*, 105, 240–260 (2017). <https://doi.org/10.1016/j.tre.2016.08.006>.

19. Lin, C., Choy, K. L., Ho, G. T., Chung, S. H., & Lam, H. (2014). Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, 41(4), 1118–1138
20. Lin, X., Negenborn, R. R., & Lodewijks, G. (2016). Towards quality-aware control of perishable goods in sychromodal transport networks. *IFAC-PapersOnLine*, 49(16), 132–137.
21. Mes, M., & Iacob, M. (2016). Sychromodal transport planning at a logistics service provider. In *Logistics and supply chain innovation* (pp. 23–36). Springer.
22. Miller-Hooks, E., Zhang, X., & Faturechi, R. (2012). Measuring and maximizing resilience of freight transportation networks. *Computers & Operations Research*, 39(7), 1633–1643.
23. Min, H. (1991). International intermodal choices via chance-constrained goal programming. *Transportation Research Part A: General*, 25(6), 351–362.
24. Nabais, J. L., Negenborn, R. R., Benitez, R. B. C., & Botto, M. A. (2013). A constrained MPC heuristic to achieve a desired transport modal split at intermodal hubs. In *2013 16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)* (pp. 714–719). IEEE.
25. Pérez Rivera, A., & Mes, M. (2016). Service and transfer selection for freights in a sychromodal network. *Lecture Notes in Computer Science*, 9855, 227–242.
26. Pfoser, S., Treiblmaier, H., & Schauer, O. (2016). Critical success factors of sychromodality: Results from a case study and literature review. *Transportation Research Procedia*, 14, 1463–1471.
27. Phillipson, F. (2015). Creating timetables in case for disturbances in simulation of railroad traffic. In *Proceedings of the 45th International Conference on Computers & Industrial Engineering (CIE45)* (pp. 1–8).
28. PlatformSychromodaliteit. (2017). Sychromodality. Retrieved Feb 7, 2018 from www.sychromodaliteit.nl/
29. Pleszko, J. (2012). Multi-variant configurations of supply chains in the context of sychromodal transport. *LogForum*, 8(4), 287–295.
30. Puettmann, C., & Stadler, H. (2010). A collaborative planning approach for intermodal freight transportation. *OR spectrum*, 32(3), 809–830.
31. Riessen, B. V., Negenborn, R. R., Dekker, R., & Lodewijks, G. (2013). Service network design for an intermodal container network with flexible due dates/times and the possibility of using subcontracted transport. *International Journal of Shipping and Transport Logistics*, 7(4), 457–478.
32. Song, D., & Dong, J. (2012). Cargo routing and empty container repositioning in multiple shipping service routes. *Transportation Research Part B: Methodological*, 46(10), 1556–1575.
33. SteadieSeifi, M., Dellaert, N., Nuijten, W., Woensel, T. V., & Raoufi, R. (2014). Multi-modal freight transportation planning: A literature review. *European Journal of Operational Research*, 233(1), 1–15.
34. Tavasszy, L., Behdani, B., & Konings, R. (2015). Intermodality and sychromodality. SSRN.com
35. Theys, C., Dullaert, W., & Notteboom, T. (2008). Analyzing cooperative networks in intermodal transportation: A game-theoretic approach. In *Nectar logistics and freight cluster meeting* (pp. 1–37). Delft, The Netherlands.
36. Topaloglu, H. (2006). A parallelizable dynamic fleet management model with random travel times. *European Journal of Operational Research*, 175(2), 782–805.
37. Topaloglu, H. (2007). A parallelizable and approximate dynamic programming-based dynamic fleet management model with random travel times and multiple vehicle types. In *Dynamic fleet management* (pp. 65–93). Springer.
38. Topaloglu, H., & Powell, W. (2005). A distributed decision-making structure for dynamic resource allocation using nonlinear functional approximations. *Operations Research*, 53(2), 281–297.
39. Topaloglu, H., & Powell, W. (2006). Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing*, 18(1), 31–42.

40. Topaloglu, H., & Powell, W. (2007). Sensitivity analysis of a dynamic fleet management model using approximate dynamic programming. *Operations Research*, 55(2), 319–331.
41. Van Binsbergen, A., Konings, R., Tavasszy, L., & Van Duin, J. (2014). Innovations in intermodal freight transport: Lessons from Europe. In *Papers of the meeting of the transportation research board*, Washington (USA). Revised paper. TRB.
42. Van Riessen, B., Negenborn, R. R., & Dekker, R. (2015). Synchronodal container transportation: An overview of current topics and research opportunities. *Computational Logistics*, 9335, 386–397.
43. Vinke, P. (2016). Dynamic consolidation decisions in a synchronodal environment: Improving the synchronodal control tower. Master's Thesis, University of Twente.
44. Xu, Y., Cao, C., Jia, B., & Zang, G. (2015). Model and algorithm for container allocation problem with random freight demands in synchronodal transportation. *Mathematical Problems in Engineering*, 2015, 986152. <https://doi.org/10.1155/2015/986152>
45. Zhang, M., & Pel, A. (2016). Synchronodal hinterland freight transport: Model study for the port of Rotterdam. *Journal of Transport Geography*, 52, 1–10.

Part II

Solving MCMCF Problems

In this part, we focus on the top-right quadrant of Fig. 1.2, where we want a global optimal solution for the synchromodal system, based on global information. The size of the problem, the amount of information, and the dynamic nature of synchromodal problems urge for solution techniques that are fast. The problem will be modelled as a minimum-cost multi-commodity flow problem on a space–time network throughout this part. We start, in Chaps. 3–5, by proposing methods to solve this problem for quadrants 1, 2, and 3 of Fig. 1.4: In the first problem, vehicle schedules are assumed to be already fixed, and one only has to decide how to assign containers to modality paths, to get each container where it needs to be against minimum total cost. In the second problem, the challenge is again only to assign containers to modality paths; however, many elements, such as travel times, can be stochastic. In the third problem, there are no more random elements, the decision-maker must simultaneously determine vehicle timetables and container-to-mode assignment; in other words, the decision-maker must tell barges and other vehicles where to go, when, and what containers to take with them.

In the next two chapters, Chaps. 6 and 7, we propose alternative performance indicators, next to costs, to use as optimisation criterion: robustness, flexibility, and customer satisfaction. Based on these three new performance indicators, we suggest an interactive approach based on multi-objective optimisation to use these performance indicators in practice.

In the last two chapters of this part, we dive further into solving the minimum-cost multi-commodity flow problem on a space–time network efficiently. For this, various variable reduction and cutting plane techniques are proposed to reduce the size and complexity of the problem.

Chapter 3

Deterministic Container-to-Mode Assignment



D. Huizing

Abstract This chapter presents a social optimisation problem (following Fig. 1.2), where both the events and the infrastructure are fixed (following Fig. 1.4). It is assumed that the transportation vehicles have fixed time tables, and one only has to decide on a container-to-mode assignment, so by what modality paths all containers reach their destination against minimal total cost. The containers have release times and deadlines. A model that also allows soft due dates is developed. Moreover, the option of using trucks or other “infinite resources” to help fulfil requests is added. With appropriate graph reductions, this problem can be solved to optimality in little time by solving the minimum-cost multi-commodity flow problem on an appropriate space–time network.

Introduction

We start here in the first quadrant of Fig. 1.4 and pose the question how a low-cost net-centric container-to-transport assignment can be found fast enough for online use if everything is known beforehand. With *deterministic container-to-mode assignment* (Problem 1), the $\bar{R}|\bar{D}, [D2R]|social(1)$ -problem (following the framework presented by Chap. 2) is meant: to assign freight containers to transports, such that the containers reach their destinations before a deadline against minimum total cost, given that the transports have fixed given schedules and all features of the problem are deterministic. An example is given in Fig. 3.1. Here, 18 containers at location A are due at location B at 14.00 h. There are two available transits: the top one has departure time 09.30 h, arrival time 13.30 h, price 523, and capacity 20; the bottom one has departure time 11.00 h, arrival time 12.30 h, price 498, and capacity 10. The cheapest feasible container-to-mode assignment is to send 10 containers over the bottom transit and 8 over the top transit.

D. Huizing (✉)
TNO, The Hague, The Netherlands

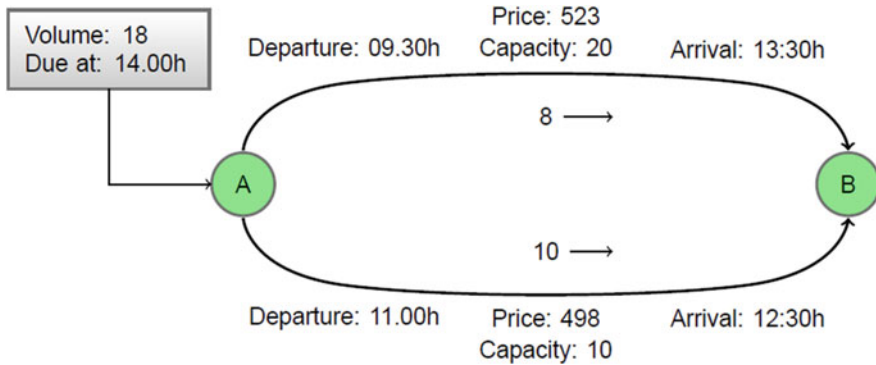


Fig. 3.1 An example of deterministic container-to-mode assignment, or Problem 1

Deterministic container-to-mode assignment has several direct and indirect applications. The studied one is of planning container flows on an operational level, given some transports with fixed schedule and no stochastic elements. However, it is also reinterpretable to intermodal service network design on the tactical level, for example to determine weekly freight flows: this is merely a difference in time scale and whether time “loops” or not. Even when considering stochastic elements, being able to solve deterministic container-to-mode assignment is useful for comparing what could have been possible if the future was known or predicted well enough. As a final use, Chap. 5 will contain stochastic container-to-mode subproblems that reduce to deterministic ones.

Modelling the Problem as a MCMC Flow Problem on a Space–Time Network

The network represented in Fig. 3.1 differs from classical graphs in that the edges have time restrictions attached to them. Though time-dependent graphs have been studied as such [3], this chapter uses the common technique of rewriting the graph to a space–time network [1, 5]. Solving Problem 1 can then be done by solving the non-negative integral minimum-cost multi-commodity flow problem on such a space–time network. Both concepts are elaborated on in this section, as well as extensions with which to allow some lateness of deliveries and calling on trucks in times of need.

Space–Time Networks

Graph problems with time restrictions may benefit from being written as a *space–time network*. Such a representation can make the time restrictions explicit in the

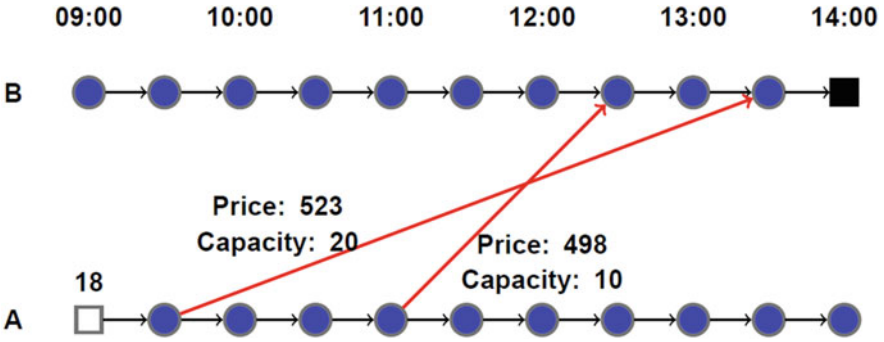


Fig. 3.2 The network described in Fig. 3.1, translated to a space–time network without trucks. The black arcs represent “waiting arcs” with infinite capacity and cost zero. The 18 containers are situated at the white square node in space–time and need to go to the black square node in space–time

graph structure. A space–time network is a digraph where a node does not represent a physical place, but rather a physical place at a certain time step. If $D = (V, A)$ is a digraph where the arcs have departure and arrival times, a space–time network \mathcal{S} can be based on it in the following way:

- Pick a number of time steps T , based on the desired time window length and time discretisation fineness.
- For $v = 1, \dots, |V|, t = 0, 1, \dots, T$, define space–time node $s_{v,t}$.
- Let \mathcal{S} have as node set the “grid” $\{s_{v,t} | v \in V, t \in \{0, 1, \dots, T\}\}$.
- Initialise the arc set of \mathcal{S} as all “waiting arcs,” in other words, all arcs in the set $\{(s_{v,t}, s_{v,t+1}) | v \in V, t \in \{0, 1, \dots, T - 1\}\}$ with weight 0 and capacity ∞ .
- For every arc a in A , translate it to an arc in \mathcal{S} by finding the space–time nodes corresponding to the start and end points of a and then connecting them with the same weight and capacity as a . Add the resulting arc to the arc set of \mathcal{S} .
- Add “truck arcs” to the arc set of \mathcal{S} , depending on the modelling choice for trucks that can be called on at any time. Some of the possible choices will be presented in section “[Infinite Resource Models and the Corresponding Graph Reductions](#)”.

In Fig. 3.2, the network of Fig. 3.1 is translated to a space–time network. This space–time network assumes that there are no trucks, thus no truck arcs. Indeed, the time restrictions described in Fig. 3.1 are now embedded explicitly in the graph structure of the graph in Fig. 3.2.

Minimum-Cost Multi-Commodity Flow

In the previous section, it was shown how the studied problem can be represented in a space–time network, thus how the time dependencies can be made explicit in the graph structure. A question that was not answered, however, was how to

optimally move the container demands from their origins in space-time to their due destinations in space-time. This can be done by solving the *non-negative integral minimum-cost multi-commodity flow problem* (MCMCF) on the space–time network.

The MCMCF is the problem of moving flow of different types as cheaply as possible through a network where arcs have weights and may have limited capacity. More specifically, denote for some digraph $D = (V, A)$ and a list of commodities K the following variables and parameters:

- The variables $x_{i,j}^k$ for the amount of flow of commodity k that is being sent over arc (i, j)
- The parameters $c_{i,j}$ for the capacity of arc (i, j) , so the maximal total amount of flow that can be sent over this arc
- The parameters $f_{i,j}$ for the cost of sending one unit of flow of any commodity over arc (i, j)
- The parameters s_k for the source node from which the flow of commodity k emanates, and t_k for the sink node where all the flow of commodity k should go
- The parameters d_k for the amount of flow of commodity k that emanates from s_k and needs to go t_k

Then the MCMCF can be written as the following minimisation problem, in integer linear programming (ILP) form:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} \sum_{k \in K} f_{i,j} x_{i,j}^k \\ \text{s.t.} \quad & \sum_{k \in K} x_{i,j}^k \leq c_{i,j} \quad \forall (i, j) \in A \end{aligned} \quad (3.1)$$

$$\sum_{(s_k, j) \in A} x_{s_k, j}^k = d_k \quad \forall k \in K \quad (3.2)$$

$$\sum_{(i, t_k) \in A} x_{i, t_k}^k = d_k \quad \forall k \in K \quad (3.3)$$

$$\sum_{(i, v) \in A} x_{i, v}^k = \sum_{(v, j) \in A} x_{v, j}^k \quad (\forall k \in K)(\forall v \in V \setminus \{s_k, t_k\}) \quad (3.4)$$

$$x_{i,j}^k \in \mathbb{N} \quad (\forall (i, j) \in A)(\forall k \in K). \quad (3.5)$$

Inequality (3.1) states that the total flow on any arc cannot exceed the arc capacity. Equality (3.2) states that for every commodity k , a total of d_k flow of commodity k must leave the source node s_k . Similarly, equality (3.3) states that exactly d_k flow of commodity k must enter the sink node t_k . Equality (3.4) states that in all other nodes, there must be flow conservation: whatever flow of some commodity enters the node, must also leave it. Finally, (3.5) states that the amount of flow of any commodity that may traverse any arc must be a natural number. The latter is essential for modelling the problem at hand, as for the practicality of the model, it is not allowed to put a non-integral or negative amount of containers onto a transit.

Now, Problem 1 can be modelled and solved as follows. Given some transport network with some fixed mode schedules, one can translate this to a space–time network. Next, every *batch* of containers that is released at some location and

some time and that is due at some location and some time can be interpreted as a commodity with corresponding d_k (volume), s_k (source node in the space–time network), and t_k (sink node in the space–time network). Finally, the optimal assignment can be seen as an instance of the MCMCF, and the corresponding ILP (integer linear programming problem) can be solved using an ILP solver.

The MCMCF and other linear cost multi-commodity network flow problems are well studied, and the interested reader is referred to Kennington’s survey [8].

Allowing Lateness with Virtual Sinks

In the logistics business, deadlines are often considered to be soft [2, 13]. Henceforth, this report will refer to a *due date* as the time demand is supposed to be at its destination, which may be violated against some penalty. With *deadline*, a hard final date will be meant. A demand item is allowed to have both a due date and a deadline.

One may easily expand the given model to incorporate these due dates and deadlines with time-specific unit penalties. An example of this is given in Fig. 3.3. The idea is to, for every commodity k , replace the original sink space–time node, t_k ,

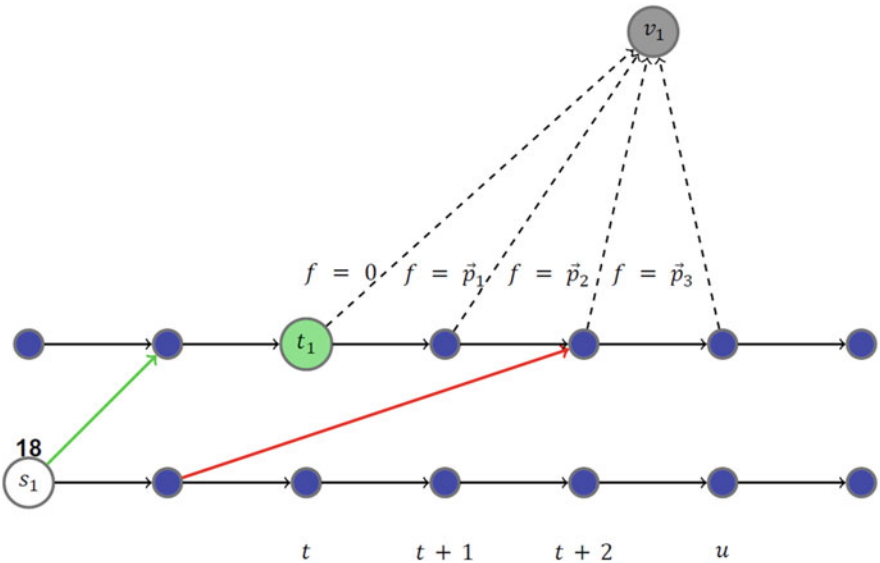


Fig. 3.3 18 containers should be delivered to t_1 , but they are allowed to be up to three time steps late, if they are made to be due at virtual node v_1 . If x containers are i time steps late, the penalty is $x \cdot \mathbf{p}_i$. If the green arc is very expensive and \mathbf{p}_2 is very low, it may be worthwhile to send containers over the red arc rather than the green arc. One may choose to connect the nodes beyond u to v_1 as well, against cost \mathbf{p}_3 or ∞ , in order to guarantee the existence of feasible paths at any time step

by a virtual sink node v_k floating outside of space-time. Suppose the due location is L , the due time is t , the deadline is u , and the time-dependent unit penalty is specified by some $(u - t)$ -dimensional vector \mathbf{p} . Then the first node space-time node “after” t_k , $s_{L,t+1}$, can be connected by an arc to v_k with the first entry of \mathbf{p} as its weight and with infinite capacity. The same can then be done at times $t + 2, t + 3, \dots, u$. Of course, an arc with infinite capacity and cost 0 should be drawn from t_k to v_k to account for goods delivered on time. One may choose to implement connections beyond time u as well, using the last entry of \mathbf{p} as weight; this is useful to guarantee feasibility in Chap. 5. The last entry of \mathbf{p} can then be made arbitrarily high to imply deadlines. If they are all made arbitrarily high, this implies that the due date equals the deadline, on other words, that no delay is allowed. It is worthwhile to note that the presented extension makes the model applicable to $\bar{R}|\bar{D}, [D2R], \widetilde{DP}|social(1)$ -problems where the delay penalty depends arbitrarily on time and linearly on amount.

Solving to Optimality

If there is only one batch of containers, the MCMCF reduces to a non-negative integral minimum-cost flow problem. This problem is known to be solvable in polynomial time and space because the continuous version is known to have an *integral solution polytope* [6]: in other words, every vertex of the solution polytope has integral coordinates. Every linear program has its optimal values in vertices of its solution polytope. LP (linear programming problem) solvers can find the optimal solution, thus the optimal vertex, in polynomial time and space, for example by using interior point methods. Therefore, the optimal solution of the LP relaxation of the non-negative integral minimum-cost flow problem can be found efficiently. This gives a lower bound on the non-negative integral minimum-cost flow problem. But because this optimum is in a vertex and all vertices have integral coordinates, this optimum has only integral flows, thus is a feasible solution to the non-negative integral minimum-cost flow problem that equals a lower bound on the problem, thus is an optimum. In other words: if the LP relaxation of a problem has an integral solution polytope, the integral optimum can be found simply by using an LP solver.

Finding a non-negative integral two-commodity flow on a directed graph, however, is proven by Even to be NP-complete [4]. Finding one with minimum cost and with at least two commodities must be at least as difficult. In some cases, it is possible to easily rewrite a MCMCF as a single minimum-cost flow problem, by creating a super source s' from which all sources receive their flow and a super sink t' where all sinks send their flow to. An example of this is given in Fig. 3.4. However, the optimal single minimum-cost flow cannot always be reinterpreted as a feasible multi-commodity flow, as shown in Fig. 3.5.

Surprisingly, however, the LP relaxation of the studied MCMCFs almost always already has an integral optimum. The implication is that it almost always suffices to solve the MCMCF by means of an LP solver, which is typically fast, instead of an

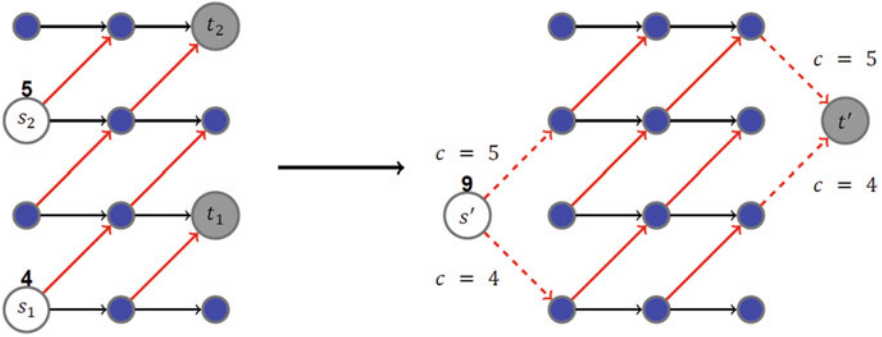


Fig. 3.4 On the left, a two-commodity minimum-cost flow problem on a space–time network, where all arcs have infinite capacity. On the right, an equivalent single-commodity minimum-cost flow problem, where the dashed arcs have indicated capacity and all the others have infinite capacity. Every flow that is feasible in the network on the right can be reinterpreted as a feasible flow in the network on the left because no flow from s_1 can ever reach t_2 and vice versa

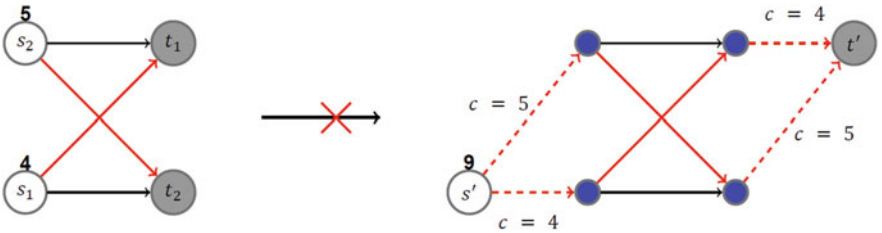


Fig. 3.5 On the left, a two-commodity minimum-cost flow problem on a space–time network, where all arcs have infinite capacity and the red arcs are more expensive than the black arcs. On the right, a single-commodity minimum-cost flow problem on the same network but with a super source and super sink, where the dashed arcs have indicated capacity and all the others have infinite capacity. The optimal flow in the problem on the right will send 4 containers over the black arcs and 1 over a red arc, as the black arcs are cheaper than the red arcs. However, this result cannot be reinterpreted as a feasible flow for the problem on the left

ILP solver, which is typically slow. In generating 500 random instances of MCMCF on space–time networks, all 500 had this immediate integrality. Ozdaglar noted that most of their non-integral optima were due to some form of perfect symmetry in a cycle [10], which could lead one to suspect that digraphs *without directed cycle* do have an integral solution polytope. This, unfortunately, is not always true: Fig. 3.6 shows a digraph without directed cycle that has no feasible non-negative integral two-commodity flow, but which does have a non-integral one. Figure 3.7 shows a space–time network based on this previous digraph. In this space–time network, two expensive truck arcs have been added to enable integral feasible flows. But even then, solving the LP relaxation on the instance in 3.7 gives a non-integral optimum, as it will try to squeeze itself through the cheap red network isomorphic to the one in Fig. 3.6.

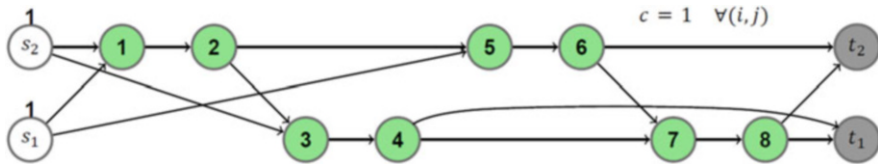


Fig. 3.6 A digraph without directed cycles. Assuming that each arc has capacity 1, it has no feasible non-negative integral two-commodity flow, but it does have a non-integral one: namely, send half a unit of commodity 1 over $(s_1, 1, 2, 3, 4, t_1)$ and the other half over $(s_1, 5, 6, 7, 8, t_1)$, while sending half a unit of commodity 2 over $(s_2, 1, 2, 5, 6, t_2)$ and the other half over $(s_2, 3, 4, 7, 8, t_2)$

What can be predicted, however, is that they are unlikely to influence computational time much. In the given examples, as soon as one of the non-integral flow variables is branched upon, the resulting branches do have integral LP-relaxation optima. This property is due to the fact that the conflicts caused by these entwinings are “resolved” by giving one of the paths priority. As branch-and-bound applications first check whether the LP relaxation has an integral optimum, the advice is to simply use ILP solvers: if their search trees do not have depth 1, they should have depth no more than the amount of commodities caught in each entwining, given that resolving entwinings do not create new entwinings.

A final note made on this problem is that the problem and its ILP may lend themselves well to Lagrange relaxation on the capacity constraints: in other words, to punish flows that exceed capacity, but not forbid them. Lagrange relaxations are known to give bounds at least as strong as LP relaxations [7], and the optimum of the Lagrange relaxation can probably be computed easily: without the capacity constraints, the problem reduces to a shortest path problem for each request. These Lagrange relaxations are irrelevant when the solution polytope is indeed integral but may give improvements for non-integral solution polytopes or help in developing shortest path-based heuristics.

Infinite Resource Models and the Corresponding Graph Reductions

An assumption that is sometimes made in the literature [9] is that freight can always be transported from any location to any other location by trucks with infinite capacity and a given speed, but that trucks are more expensive than other modalities. Instead of trucks, one could also model the option of subcontracted transport that ensures timely arrival against a price, without much mathematical difference. This method of modelling will prove useful in Chap. 6 to limit the sheer amount of vehicles to explicitly control. From this point, a distinction is made between *infinite*

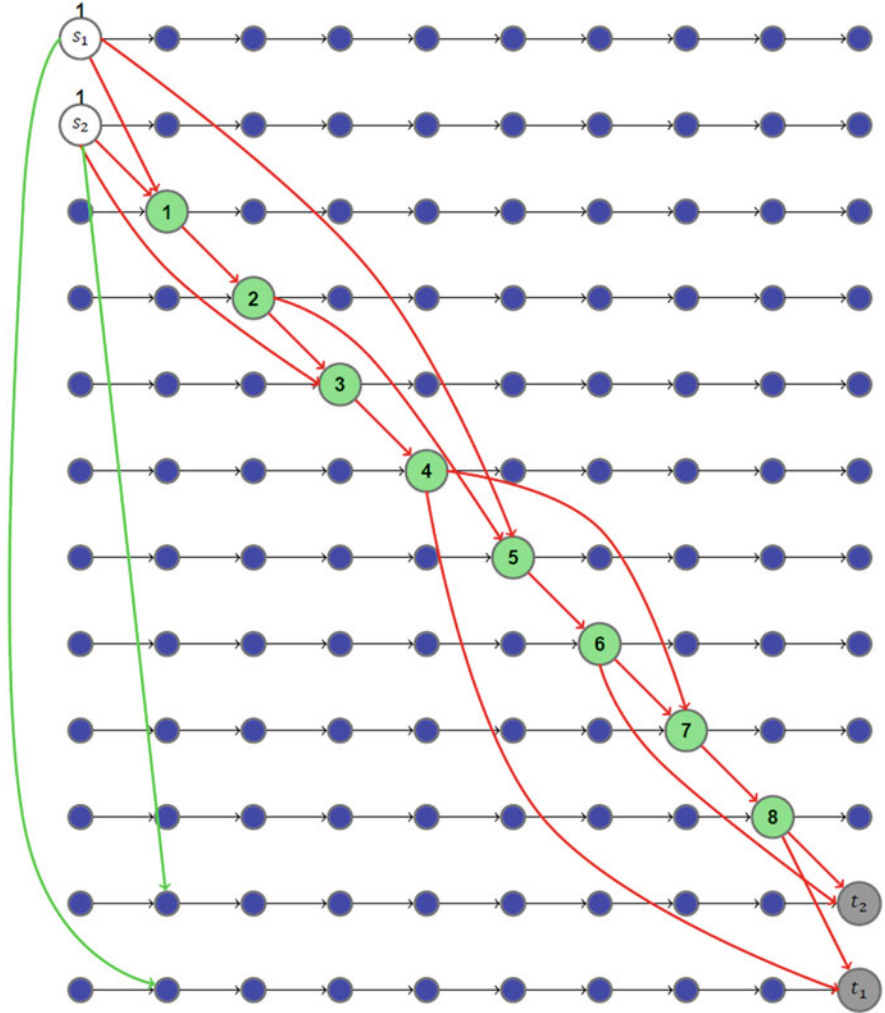


Fig. 3.7 An instance where the LP relaxation of the MCMCF has a non-integral optimum, given that the red arcs are cheap and have capacity 1, while the green arcs are expensive and have capacity 100. There is an integer feasible point, namely by sending all flow over the green arcs. However, it is possible and cheaper to send the flow only over red arcs; this subgraph is isomorphic to the one in Fig. 3.6

resources, such as a separate truck department or subcontracted transportation, and *vehicles*, such as controlled barges and trains. To steer intuition, the terms “trucks” and “infinite resources” will sometimes be used interchangeably throughout this report.

If infinite resources are modelled, so it is allowed to send containers from any place to any other place at any time against a high price by for example trucks, this would imply that a lot of truck arcs should be added to the previously discussed space–time networks to incorporate this flexibility. However, adding a truck arc to the space–time network from any location to any other location at any time comes at the cost of adding $n(n-1) \cdot (T-1)$ arcs, thus $n(n-1) \cdot (T-1) \cdot |K|$ variables, where n is the number of locations in the system, T is the final time step, and K is the set of observed commodities. Even though it was argued that ILP solvers should find an optimum in relatively little time, the underlying LP solvers do become significantly slower when an excessive amount of variables are added. The overall computational performance can be significantly improved by excluding unnecessary nodes and arcs in the space–time networks and conflating waiting arcs where possible. How such a reduction process can be done depends on the modelling choices for the trucks. Therefore, this section describes two possible models for trucks with a corresponding graph reduction.

Double Matrix Infinite Resources

Suppose that, indeed, the following assumptions are made:

- Trucks or other infinite resources can always be employed from anywhere to anywhere with infinite capacity.
- The infinite resources have a fixed travel time $M_{i,j}^1$ that depends only on origin–destination pair (i, j) .
- Transporting containers with an infinite resource from location i to location $j \neq i$ have a unit cost $M_{i,j}^2$ per container that depends only on origin–destination pair.
- Triangle inequality: trucking from A to B directly is always cheaper than through C.

Allowing such trucks has two important merits: it is now possible to send containers by truck if all other modality paths are infeasible in time or capacity, and under smart choices of the matrices M^1, M^2 , the model now more closely simulates the practice of dividing a journey up into pre-haul, long haul, and post-haul.

Figure 3.8 shows a space–time network where it is always possible to take a truck from any location to any other location in one time step. The truck arcs have high cost and infinite capacity. This space–time network has a lot of redundancy:

- Time step 0 is superfluous.
- Because of the triangle inequality, location A is useless.
- First trucking from C to D and then waiting one time step are equivalent to first waiting one time step and then trucking from C to D. Only one truck arc is needed to represent this option of trucking. In order to facilitate synchromodality, so to facilitate keeping options open until more has become known, one could choose to truck only at the last minute to go to t_k .

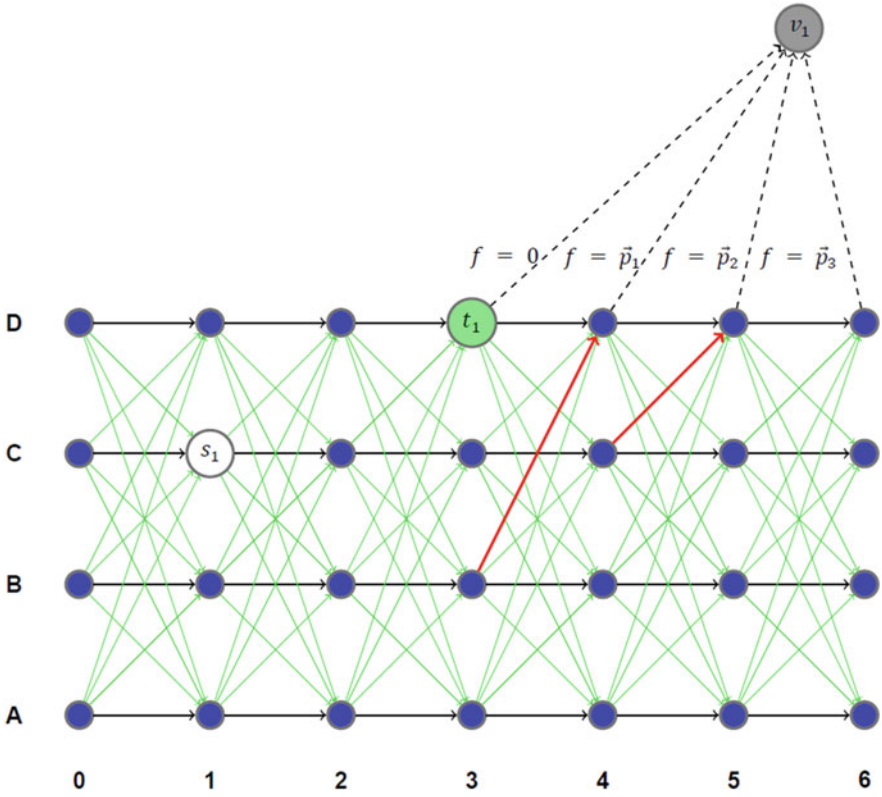


Fig. 3.8 An unreduced space–time network with double matrix infinite resources: it is possible to truck with infinite capacity from any location to any other location at any time against some high price. This price depends only on origin–destination pair. Regardless of origin–destination pair, the travel time is always one time step. The thin green arcs are these expensive truck arcs with infinite capacity. This network contains a lot of redundancy under these assumptions: trucking immediately from C to D and then waiting one time step are equivalent to the converse

- Similarly, if flow is to travel over some non-truck transit arc, one could truck only at the last minute to catch that transit.
- If the waiting arc from $s_{C,3}$ to $s_{C,4}$ is used by some flow, then that same flow will always use the waiting arc from $s_{C,4}$ to $s_{C,5}$, so they might as well be joined into one arc.

From the above observations, it becomes clear that many nodes and arcs can be removed from the network in Fig. 3.8 to obtain a reduced instance that has the same optimum but involves a lot less arcs, thus a lot less variables, thus a lot less computation to solve. Most importantly, many redundant paths can be removed by allowing trucking only at the last moment, either to go to the (non-virtual) sink or

to catch some transit. Rather than first generating a full space–time network and then removing arcs and nodes, it requires significantly less computation to already exclude all unnecessary arcs and nodes during generation. The employed way to generate such a reduced instance is described as Algorithm 1.

Using Algorithm 1, one could obtain a reduced instance to replace the instance shown in Fig. 3.8. This reduced instance is shown in Fig. 3.9. Aside from creating a reduced space–time network, one can also discard the flow variables for all but one commodity on the arcs that end in a virtual sink.

It can be seen that more reduction is possible still: for example, the node at $(B, 2)$ is useless. More importantly, under the current method, more and more nodes will remain preserved as time progresses because they could technically act as pre-sink nodes from which to depart to a virtual sink if indeed all nodes after the deadline are still connected for feasibility. However, one may see by inspection that the pre-

Algorithm 1 Generating a reduced instance of MCMCF on a space–time network, given that infinite resources are modelled as double matrix infinite resources

Require: List of locations, list of requests (volume, due time, deadline, due location, release time, release location, lateness penalty vector \mathbf{p}) and list of vehicle transits (capacity, unit price, departure time, departure location, arrival time, arrival location), infinite resource travel time matrix M^1 , infinite resource travel time matrix M^2)

Ensure: Instance of MCMCF on a reduced space-time network

- 1: Initialise an empty space-time network \mathcal{S}
 - 2: Determine first time step T_1 as first release time among requests
 - 3: Request a final time step T_2 , or generate it by some means
 - 4: Eliminate all locations that are not interesting, so that are not a request's source or sink node or a transit's departure or arrival location
 - 5: **for** request in list of requests **do**
 - 6: Add space-time source node to \mathcal{S} , if not already present
 - 7: Add virtual sink node to \mathcal{S}
 - 8: **for** τ in *due time*, *due time* + 1, ..., T_2 **do**
 - 9: Add space-time pre-sink node at (*due location*, τ) to \mathcal{S} , if not already present
 - 10: Add arc from space-time pre-sink node to virtual sink node with infinity capacity and weight 0 if $\tau = \textit{due time}$ and with weight dictated by \mathbf{p} otherwise
 - 11: Add a truck arc to \mathcal{S} , if not already present, from any ($i = \textit{location} \neq \textit{due location} = j, \textit{due time} - M_{i,j}^1$) to (*due location*, *due time*) with infinite capacity and weight dictated by $M_{i,j}^2$, adding nodes if necessary
 - 12: **for** transit in list of transits **do**
 - 13: **if** *departure time* $\geq T_1$ and *arrival time* $\leq T_2$ **then**
 - 14: Add arc to \mathcal{S} , with supplied weight and capacity, adding nodes if necessary
 - 15: Add a truck arc to \mathcal{S} , if not already present, from any ($i = \textit{location} \neq \textit{departure location} = j, \textit{departure time} - M_{i,j}^1$) to (*departure location*, *departure time*) with infinite capacity and weight dictated by $M_{i,j}^2$, adding nodes if necessary
 - 16: For every request with *due time* $\geq \textit{arrival time}$, add a truck arc to \mathcal{S} if not already present from ($i = \textit{arrival location}, \textit{arrival time}$) to ($j = \textit{due location}, \textit{arrival time} + M_{i,j}^1$) with cost $M_{i,j}^2$
 - 17: Add horizontal waiting arcs between space-time nodes
-

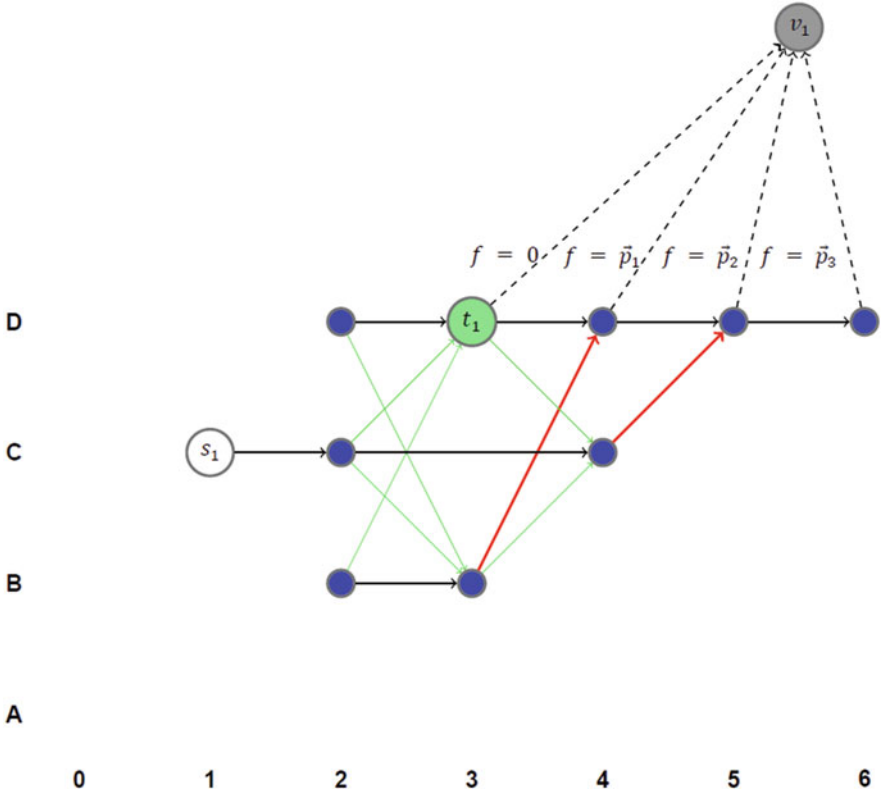


Fig. 3.9 A reduced version of the instance in Fig. 3.8, obtained using Algorithm 1. This reduced instance has only 19 arcs instead of the original 102. Further reduction should still be possible: for example, location A is useless here, as is the final pre-sink node at $(D, 6)$

sink node at $(D, 6)$ in Fig. 3.9 is useless, assuming that lateness penalties are non-decreasing. Aside this issue of more and more nodes being marked as relevant, there should also be opportunities in not always allowing trucking to catch some transit, but only if the added truck price and transit price are lower than the cost of immediate trucking to the transit’s destination.

But in spite of these possible improvements, the current algorithm already fares well in combating the growth of unnecessary truck arcs. The instance in Fig. 3.8 has 102 arcs, thus 102 variables as there is only one commodity, whereas the instance in Fig. 3.9 has only 19 arcs. While more ideas for instance reduction are imaginable, they will be left for further research.

Other Or No Infinite Resources

If there are no trucks in the model, then of course, no truck arcs need to be generated. However, one could also choose not to generate truck arcs in the fashion of section “[Double Matrix Infinite Resources](#)” because one of the key assumptions does not hold: perhaps trucks have limited capacity, or they cost more in the weekends than on weekdays, or the trucks have to be explicitly modelled to guarantee feasibility. Many truck models in this wider range of options could still be modelled by giving them space–time arcs as one would with any other modality. The necessity of generating a reduced instance would greatly depend on how many truck arcs are added this way, and the methodology of generating a reduced instance would greatly depend on the assumptions that do still hold. Therefore, further commentary on reduction methods is not provided here, and save two remarks. A reduction that should still be applicable in most cases is the conflation of waiting arcs: if a node has a waiting arc as its only incoming arc and a waiting arc as its only outgoing arc, the node may as well be removed. Additionally, one can always discard, for arcs going to a virtual sink, any flow variable that is not of the commodity that the sink belongs to. For the rest, the reader is advised to construct some reduction methodology, if necessary, by observing the given assumptions and Algorithm 1.

Numerical Results

In this chapter, it was shown that Problem 1 can be solved to optimality by solving a minimum-cost multi-commodity flow problem on a space–time network. It was argued that this can be done in relatively little time because of the rarity of instances where the LP relaxation has a non-integral optimum. Furthermore, a problem reduction procedure was shown in the form of Algorithm 1 that should decrease computation time. In this section, numerical results are presented to support this claim.

In the experiments, 10 random instances were generated in each of four test classes. These instances were made by generating lists of random transits and requests, given a time scale of the problem and a list of locations. The transits had random capacity, duration, departure time, and price, though the prices were always considerably below the standard truck price. The orders had random volume, release time, due time, and penalty vector, including instances where the penalty would become arbitrarily large to simulate hard deadlines. Each instance in test class $i = 1, 2, 3$ concerned $5i$ random requests and random transits over $5i$ locations, within a time scale of $5i + 1$ time steps. Test class 4 contained instances of “real life size”: based on data from a NWO-affiliated use case, instances in this class concerned 40 random requests and 60 random transits over 32 locations and 121 time steps. This was the minimum size specified to reflect operational use, where 121 time steps correspond to the 120h of a five-day working week. Though the instances in this class are scaled to the required size of the use case, the random

Table 3.1 Numerical results for deterministic container-to-mode assignment, or Problem 1. Class 4 contains instances of a size that was specified as useful within an operational use case

	Class 1	Class 2	Class 3	Class 4
Average running time with reduction (in seconds)	0.182	0.430	0.884	8.889
Variance of the above (in squared seconds)	$3.44 \cdot 10^{-4}$	$3.41 \cdot 10^{-3}$	$4.80 \cdot 10^{-3}$	0.948
Amount of instances with integral LP-relaxation optimum	10	10	10	10
Average running time without reduction (in seconds)	0.2972	1.046	3.061	36.33
Variance of the above (in squared seconds)	$1.07 \cdot 10^{-3}$	$4.90 \cdot 10^{-3}$	$3.07 \cdot 10^{-2}$	0.506
Amount of instances with integral LP-relaxation optimum	10	10	10	10

placement of the transit arcs does not reflect scheduling procedures from the use case: rather, they reflect the “pandemonium” a net operator may experience when supervising container-to-mode assignment over uncontrolled vehicles.

For each of these test classes, every instance was solved with and without the reduction from Algorithm 1, assuming double matrix infinite resources in both cases. The objective values, of course, were equal and optimal in both cases: rather, the solutions of two methods were compared in running time. When solving an instance, it was randomly determined which method to solve it with first, so as to mitigate cumulative slowdown bias. Also, for each of the 80 ILPs solved this way, it was monitored if indeed its LP relaxation had an integral optimum. The methods were implemented in Python 3.4, using the PuLP library and its built-in non-commercial MILP solver. The experiments were run on an Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz 2.80GHz processor. The results can be viewed in Table 3.1. These results support the following valuable conclusions:

- Algorithm 1 significantly reduces computation time.
- With or without reduction, this problem is solved to optimality fast, even in instances of “real life size.”
- The variation in computation time grows faster when using Algorithm 1, probably due to the variation in how much reduction is possible.
- Among the 80 ILPs, none was encountered of which the LP relaxation had a non-integral optimum.

Discussion

The methods described in this chapter have certain strong merits:

- An optimal solution can be found in surprisingly little time, given that the problem is NP-hard.
- The method naturally models the option of consolidation.

However, there are still points of attention that could be investigated in future work:

- The given methods may still be not fast enough when used in iterative methods in an online setting. A heuristic or further graph reduction may still be required.
- Possibilities for further graph reduction were mentioned, but not explored.
- There is currently no penalty in cost or time for containers switching modality, which may not model reality well.

Added Value

Solving this problem by solving a min-cost multi-commodity flow on a space–time network is not an entirely new idea [11, 12]. Where, then, lies the added value of this research?

- It has been proven that these methods can be translated well from tactical problems to operational problems and that they are fast enough for operational online use.
- The possibility to model soft deadlines was added.
- A significant reduction of computation time was achieved using Algorithm 1.
- A partial explanation for the low computation time was found by investigating graph theory, and the notion that an LP solver is sufficient rather than an MILP solver was shown to be true in 100% of the randomly generated instances.

Conclusion

This chapter sought to answer the following research question:

How can a low-cost net-centric container-to-transport assignment be found fast enough for online use if everything is known beforehand?

Deterministic container-to-mode assignment, or “Problem 1,” was defined to be the problem of assigning freight containers optimally to transport modalities with predetermined schedules, in the presence of time constraints and the absence of stochastic elements. It can be solved to optimality by solving the non-negative

integral min-cost multi-commodity flow problem on a space–time network. This model can be expanded to allow lateness, by introducing virtual sinks. Due to NP hardness, solving the problem requires the use of an ILP solver. In all randomly generated instances, however, the LP relaxation has an integral optimum. In a carefully constructed instance, this was not the case, but only one branching was required. Therefore, it was hypothesised that ILP solvers will find an optimal solution relatively quickly in almost all cases.

The option was added to truck from anywhere to anywhere with infinite capacity and an origin–destination-dependent unit cost. Algorithm 1 generates a reduced instance under this truck model, in order to exclude a large amount of superfluous arcs and variables. Further reductions are possible and recommended for future research.

Random instances of different sizes were generated to test this solution methodology on, with or without the reduction from Algorithm 1. They show that problems of a “real life size” can be solved to optimality in an average 8.889 s, even when using a non-commercial MILP solver on a single computer. This makes the method suitable for online use.

To answer the sub-question: by solving the min-cost multi-commodity flow problem on a space–time network with an MILP solver, an optimal assignment can be found in a matter of seconds for problems of “real life size.”

References

1. Andersen, J., Crainic, T., & Christiansen, M. (2009). Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, 193(2), 377–389.
2. Blecker, T., Kersten, W., & Gertz, C. (2008). Management in logistics networks and nodes.
3. Ding, B., Yu, J. X., & Qin, L. (2008). Finding time-dependent shortest paths over large graphs. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 205–216). ACM.
4. Even, S., Itai, A., & Shamir, A. (1975). On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science, 1975* (pp. 184–193). IEEE.
5. Helme, M. P. (1992). Reducing air traffic delay in a space-time network. In *IEEE International Conference on Systems, Man and Cybernetics, 1992* (pp. 236–242). IEEE.
6. Hoffman, A. J., & Kruskal, J. B. (2010). Integral boundary points of convex polyhedra. In *50 Years of Integer Programming 1958–2008* (pp. 49–76). Springer.
7. Kallenberg, L. (2005). *Besliskunde 4*. Universiteit Leiden.
8. Kennington, J. L. (1978). A survey of linear cost multicommodity network flows. *Operations Research*, 26(2), 209–236.
9. Kooiman, K. (2015). A classification framework for time stamp stochastic assignment problems and an application to inland container shipping. TNO Internal Documentation.

10. Ozdaglar, A. E., & Bertsekas, D. P. (2004). Optimal solution of integer multicommodity flow problems with application in optical networks. In *Frontiers in global optimization* (pp. 411–435). Springer.
11. Pedersen, M. B., Madsen, O. B., & Nielsen, O. A. (2005). Optimization models and solution methods for intermodal transportation. Ph.D. Thesis, Technical University of Denmark, Department of Transport, Traffic Modelling.
12. SteadieSeifi, M., Dellaert, N., Nuijten, W., Woensel, T.V., & Raoufi, R. (2014). Multi-modal freight transportation planning: A literature review. *European Journal of Operational Research*, 233(1), 1–15.
13. Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A Tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2), 170–186.

Chapter 4

Stochastic Container-to-Mode Assignment



D. Huizing

Abstract In this chapter, we introduce stochasticity in the events, going to quadrant 2 in Fig. 4.1. Here, the goal is the same as in the previous chapter, however, now almost any element can be stochastic: for instance, travel times and container release times could be given a discrete probability distribution rather than a fixed value. Rigorous definitions are formulated to capture the generalities in this stochasticity. Multistage stochastic optimisation and Markov Decision Processes are illustrated but advised against for their computing time: instead, Expected Future Iteration and 70%-Pessimistic Future Iteration are developed and shown to yield near-optimal results in a small amount of time in the simulated environment.

Introduction

We proceed here in the second quadrant of Fig. 4.1 and pose the question how a low-cost net-centric container-to-transport assignment can be found fast enough for online use if new data is still expected to come in. In the previous chapter, a method was developed to solve deterministic container-to-mode assignment relatively efficiently. One of the important classifiers of synchromodal transport is being able to adjust plannings in an online fashion as more information become available. A natural consequence of this is that decisions have to be made, while some parameters are still uncertain. This is also often the case in operational planning in practice: if one cheap modality is “probably” going to run into delays in the port area, and another modality is more expensive but also more likely to be on time, what decision should be made? Figure 4.1 shows how such a decision could be seen as a stochastic optimisation problem.

Though Fig. 4.1 shows a problem where only the travel time of a modality is stochastic, uncertainties can manifest in many components of the transport chain. In the literature, the travel time is often modelled as uncertain [11, 13],

D. Huizing (✉)
TNO, The Hague, The Netherlands

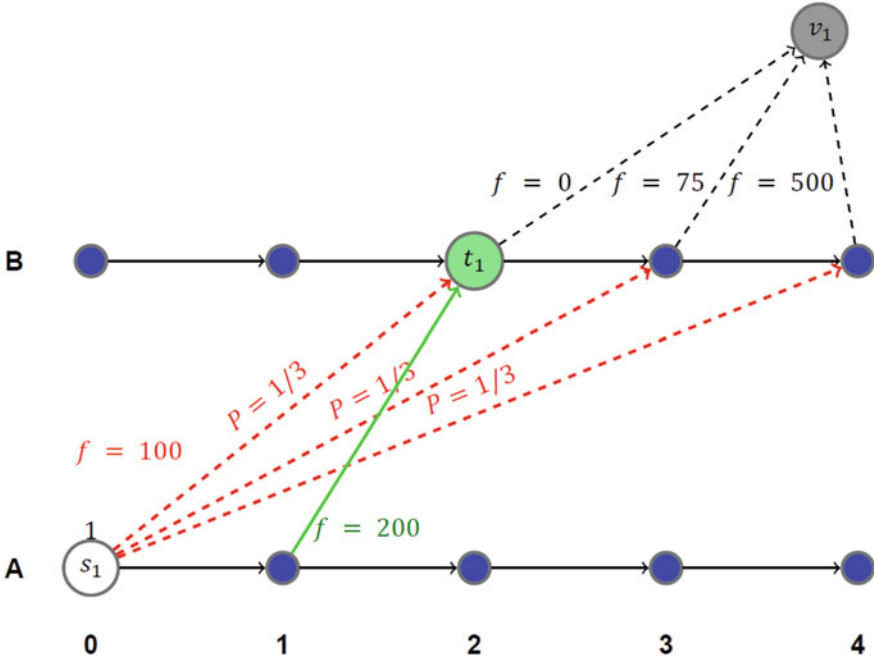


Fig. 4.1 One container has to be assigned to a transportation modality. The red, dashed option has probability $1/3$ of taking two time steps, thus arriving just on time, probability $1/3$ of taking three time steps, causing a lateness penalty of 75, and probability $1/3$ of taking four time steps, causing a lateness penalty of 500. The green option is guaranteed to be on time but has a base usage cost of 200 instead of 100. Both transports have capacity 10. At time $t = 0$, a planner is faced with the decision: red or green?

for example due to weather conditions, vehicle breakdowns, road congestion, or general unpredictability. The demand patterns are also often considered uncertain [11, 13]. Sometimes, handling time at terminals is considered uncertain due to disturbances [13]. Aside from these more common factors, however, some models also study uncertainty in things such as price and capacity of resources. Industry experts in operational planning affirm that “any element” of transport comes with uncertainties, and Caris et al. confirm that “real-life operational management is characterised by uncertainty” [4]. Even the total capacity of a barge, which one would almost always consider fixed, can be stochastic: for example, if a barge has to sail under a low bridge, then the maximum stacking height can become dependent on water levels. As such, a complete consensus on what uncertainties to model does not exist.

Where, then, does one draw the line? The move towards synchromodality is the move towards cooperation, which requires methods to remain applicable when different parties with different dynamics are integrated into the planning network. Therefore, in this chapter, methods are developed with a holistic view

towards uncertainty: all elements proposed in Chap. 2 are modelled as stochastic, if not controlled. In particular, this chapter will study *stochastic container-to-mode assignment*, by which the following $\widehat{R}|\widehat{D}, [D2R]|social(1)$ -problem is meant: to assign freight containers to transports, so that the containers reach their destinations before a deadline against minimum total costs, given that the transports have fixed given schedules and some features of the problem are stochastic.

Concepts and Definitions

In Chap. 3, the concept of solving min-cost multi-commodity flow on a space–time network was introduced. However, Fig. 4.1 shows an example where a decision has to be made on some form of “approximate” space–time network, where it is still uncertain which of its three possible outcomes it will eventually be. In this section, new concepts will be introduced from which to construct decision-making methods.

Transit Ideas and Transit Instances

The first gap that will be addressed is that a planning algorithm can no longer plan based on a list of fixed transits, but on a list of transits with yet uncertain characteristics, as also illustrated in Fig. 4.1.

In the philosophical work of Plato, the concept of an Idea is proposed: that all physical objects are instances or “shadows” generated by some higher object. For example, in his theory of Ideas, every hammer on the Earth is a different instance of one single Idea of a hammer [10]. This theory will not be discussed here, but it will be used as an analogy for the following definition.

A *transit Idea* \mathfrak{T} is defined by a sextuple $\mathfrak{T} = (\mathfrak{T}_{OD}, \mathfrak{T}_{DT}, \mathfrak{T}_{TT}, \mathfrak{T}_C, \mathfrak{T}_P, \mathfrak{T}_{\sim})$, where:

- \mathfrak{T}_{OD} is a random variable representing the origin–destination pair of the transit.
- \mathfrak{T}_{DT} is a random variable representing the departure time of the transport, so the time at which the transit starts.
- \mathfrak{T}_{TT} is a random variable representing the travel time of the transit.
- \mathfrak{T}_C is a random variable representing the capacity of the transport performing the transit.
- \mathfrak{T}_P is a random variable representing the unit price of the transit per container transported by it.
- \mathfrak{T}_{\sim} is a joint probability distribution on the random variables. If the random variables are independent, this probability distribution equals the product of marginal distributions.

In some problems, not all of these features are stochastic: for example, it is common to model transport capacity as fixed [2, 7, 8, 14]. This can be solved by making the

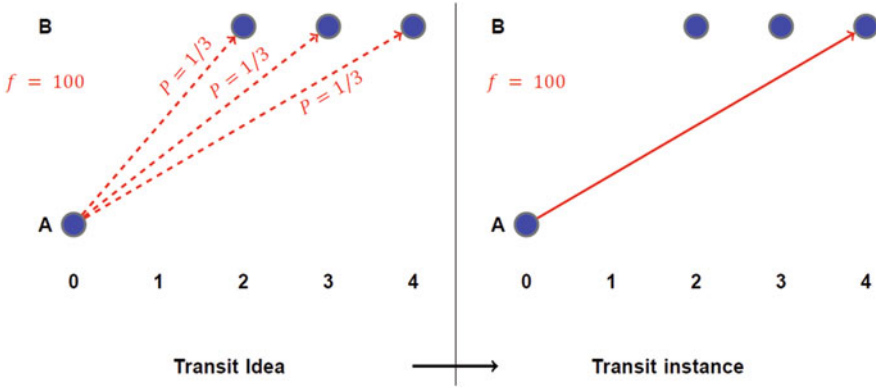


Fig. 4.2 On the left, a transit Idea of which only the travel time is uncertain; on the right, one of the three possible instances of the Idea

corresponding random variable equal to this fixed constant with probability 1. In the notation of transit Ideas, such a random variable will then be replaced by its fixed outcome. Having remarked this, the red dashed link in Fig. 4.1 corresponds to the following transit Idea \mathfrak{T}^{red} :

$$\mathfrak{T}^{red} = ((A, B), 0, X, 10, 100, U_X\{2, 4\}),$$

where $P(X = 2) = 1/3$, $P(X = 3) = 1/3$, and $P(X = 4) = 1/3$. So \mathfrak{T}^{red} will certainly instantiate with origin–destination pair (A, B) , departure time 0, capacity 10, and unit price 100, but the travel time X is drawn from a discrete uniform distribution.

When each of these five random variables are drawn, the result is henceforth called a *transit instance*. A transit instance corresponds to a fixed arc of a space–time network and is no longer subject to any randomness. In Chap. 3, exclusively transit instances were observed. The difference between transit Ideas and transit instances is also illustrated in Fig. 4.2.

Request Ideas and Request Instances

By the same philosophy of transit Ideas, a *request Idea* \mathfrak{R} is defined as a sextuple $\mathfrak{R} = (\mathfrak{R}_{OD}, \mathfrak{R}_{RT}, \mathfrak{R}_{DT}, \mathfrak{R}_A, \mathfrak{R}_P, \mathfrak{R}_\sim)$, where:

- \mathfrak{R}_{OD} is a random variable representing the origin–destination pair of the request.
- \mathfrak{R}_{RT} is a random variable representing the release time of the request, so the time from which the containers of the request can be picked up.
- \mathfrak{R}_{DT} is a random variable representing the due time of the request.
- \mathfrak{R}_A is a random variable representing the amount of containers in the request.

- \mathfrak{R}_P is a random variable representing the penalty function of lateness of the request, as also described in section “[Allowing Lateness with Virtual Sinks](#)”.
- \mathfrak{R}_\sim is a joint probability distribution on the random variables. If the random variables are independent, this probability distribution equals the product of marginal distributions.

Again, drawing these random variables creates an instance of the request Idea, which will be called a *request instance*.

Omnifutures

As transit Ideas generate all possible transit instances, and request Ideas generate all possible request instances, putting them together creates an object that generates all possible futures. Therefore, an *omnifuture* Ω is defined as a pair $\Omega = (\underline{\mathfrak{T}}, \underline{\mathfrak{R}}, \Omega_\sim^S)$, where:

- $\underline{\mathfrak{T}}$ is a list of transit Ideas.
- $\underline{\mathfrak{R}}$ is a list of request Ideas.
- Ω_\sim is a joint probability distribution on what instances are drawn from the Ideas. Ideas can be dependent: for example, if two transit Ideas correspond to the first and second transits of one vehicle, then if the former instantiates with large travel time, the latter may be more likely to instantiate with a late departure time. If all Ideas are independent, Ω_\sim equals the product of marginal distributions of the separate Ideas. Otherwise, it should be ensured that the final elements of the transit Ideas and request Ideas are correct marginal distributions of Ω_\sim .

Using Ω_\sim , one can draw values for all transit Ideas and request Ideas. The result, a *future*, can be interpreted as an instance of deterministic container-to-mode assignment and solved using the techniques from Chap. 3.

The example given in Fig. 4.1 displays an omnifuture consisting of two transit Ideas and a request Idea. For one of the transit Ideas, there is uncertainty in its travel time, and it could instantiate into one of three different values. The other transit Idea has no uncertainties: it generates only one fixed instance. The same goes for the request Idea. The three Ideas are independent, and the omnifuture generates three futures with equal probability. In a sense, an omnifuture can be seen as a “future Idea”.

The implicit assumption in this definition is that the probability distribution on possible futures is independent on the “current” state. This conflicts directly with the synchronomodal principal of adjusting decisions to updated information. This will be resolved in later sections by always basing an omnifuture on the “current” state: when moving into a new state, a new omnifuture is defined and assumed.

Finite Window Methods and Rolling Window Methods

Two important uncertainties have not been addressed in the previous sections:

1. In the definition of an omnifuture, the amount of transit Ideas and request Ideas was taken as fixed, while it may be difficult to predict how many requests will be placed.
2. None of the definitions take into account that new transit Ideas or request Ideas could enter the system during the “resolution of the future”.

Of course, these two issues are closely related. Both are resolved by designing *rolling window methods*: methods that keep adding new Ideas as they enter the system and make decisions according to the presence or anticipation of new request Ideas and transit Ideas. If a method assumes that no new Ideas will enter the system at any point, this will be referred to as a *finite window method*.

Locked Futures and Future Trees

As a final distinction, a model-maker must decide whether or not the future depends on the decisions made by the decision-maker: in other words, whether or not the omnifuture is independent of decisions. For example, some barge operators are known to wait with departure until their barge is at least 70% filled with cargo, in order to make their trip worthwhile: in this case, their uncertain departure time is dependent on the decision-maker’s container-to-mode assignment.

This chapter assumes *locked futures*, meaning that the instantiation of Ideas does not depend on choices in container-to-mode assignment. In other words: *there is only one true future, but the decision-maker does not yet know which one that is*. If one would not assume locked futures, one would have to make decisions regarding some form of *future tree*: a tree of possible futures that branches on decisions. Though the latter is more general and arguably more representative of real-world practice, it is not investigated in this chapter due to expected issues of computational and conceptual complexity.

Demifutures

The following two assumptions could be made to keep the problem of stochastic container-to-mode assignment manageable:

- The window is finite. At the start of the problem, all transit Ideas and request Ideas are given. No new ones will enter the system in the given time window. See also section “[Finite Window Methods and Rolling Window Methods](#)”.

- The future is locked. How the Ideas will instantiate is independent of the choices in container-to-mode assignment. See also section “[Locked Futures and Future Trees](#)”.

Under these assumptions, the following situation must occur. At the start of the time window, some omnifuture Ω_0 is given. Because the future is locked, there is some “true future” Φ among the futures generated by Ω_0 , but it is probably not yet known which one this is. If Φ were known, the optimal container-to-mode assignment could be easily found using the techniques from Chap. 3. Instead, one has to make decisions at time step 0 against an uncertain future. At time step 1, more may be known: for example, transits may turn out to depart in this time step, making their departure times and destinations known, or requests may have been released, making their release time and other features known. Now, decisions have to be made at time step 1 against some induced omnifuture Ω_1 .

Because the window is finite and all time-related features can be assumed to be finite as well, eventually, all transits and requests will have taken place, making all their features known. Φ will have revealed itself to be the true future, and hopefully, good decisions will have been made.

Under the two assumptions, a *demifuture* Ω_t^Φ can be defined as an omnifuture that was obtained in the following way: by taking some initial omnifuture Ω_0 , and knowing some true future Φ , revealing everything that can be known of Φ at time t , and limiting Ω_0 accordingly. If $t \rightarrow \infty$, Ω_t can only generate Φ as a possible future. See also Fig. 4.3.

In practice, of course, Φ is not known. The main purpose of demifutures is to simulate the synchromodal principal of more becoming known over time and to be able to design and test decision processes using this. Though equivalent definitions

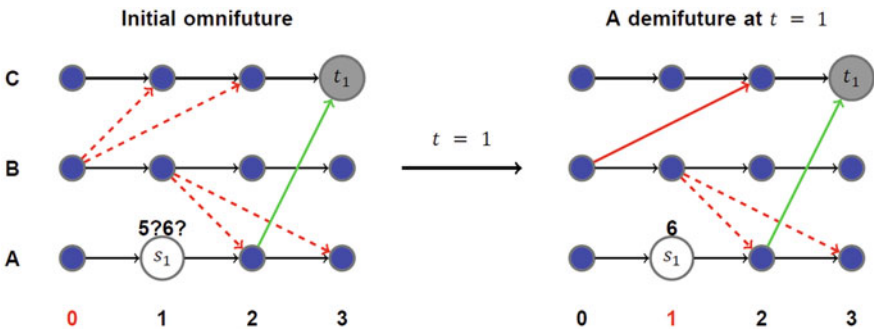


Fig. 4.3 On the left, an omnifuture with three transit Ideas, of which only two have an uncertainty (travel time), and a request Idea with only uncertain volume. This omnifuture generates 8 possible futures. At $t = 1$, the request has been released, and it turned out to have a volume of 6 containers. The transit that departed at $t = 0$ has not yet been finished, so it cannot have duration 1, so it must have duration 2, the only other option. In the demifuture at $t = 1$, only one Idea has uncertainty left. This demifuture generates 2 possible futures. Every demifuture after this will only generate Φ , the true future

and properties of demifutures may well be derived without these two assumptions, these are not necessary for this chapter.

Solving to Optimality

In this section, multistage stochastic programming and Markov decision processes will be discussed as methods to solve Problem 2 to optimality. However, they will also be shown to be computationally far too heavy to be of practical use.

Two-Stage Stochastic Programming

Multistage stochastic programming relies on a technique found in two-stage stochastic programming, which will be introduced here. *Stochastic programs* resemble linear programs, but with an expected value in the objective function based on randomness in the parameters that define the decision space and/or objective function.

If the stochastic component of the objective function can only take a finite amount of values against known probabilities, and these values are linear in decision variables, then the expected value can be substituted by these values times their probabilities to create one large linear program, which is called its *deterministic equivalent*. An example will be given later in this section.

In *two-stage stochastic optimisation*, some decisions have to be made before the random variables instantiate and some have to be made after. In the second stage, there are no more random variables, and the choice becomes much simpler; the difficulty lies in the first stage, where decisions have to be made that influence the decision space of the second stage based on uncertainties. One then has to balance the direct cost of decisions in the first stage and the implied expected cost of decisions in the second stage.

A classical example of this is the newsvendor problem [6]: one wants to sell as many newspapers as possible in stage 2 but has to order stock in stage 1 without knowing what the demand will be in stage 2. Unsold newspapers become worthless at the end of the day. Suppose, for the sake on an example, that newspapers can be ordered at cost 2 and sold at cost 4 and that if D is the demand,

$$P(D = 1) = 0.1, \quad P(D = 2) = 0.2, \quad P(D = 3) = 0.3, \quad P(D = 4) = 0.4.$$

If x is the amount of newspapers to order in stage 1, y is the amount to sell in stage 2, and y_i is the amount to sell knowing that $D = i$, problem 4.1 is a stochastic

program that describes this and problem 4.2 is its deterministic equivalent.

$$\begin{aligned} \max_{x,y} \quad & -2x + 4 \mathbb{E}[y] \\ \text{s.t.} \quad & y \leq x \\ & y \leq D \\ & x, y \in \mathbb{N} \end{aligned} \tag{4.1}$$

$$\begin{aligned} \max_{x,y_1,y_2,y_3,y_4} \quad & -2x + 4(0.1y_1 + 0.2y_2 + 0.3y_3 + 0.4y_4) \\ \text{s.t.} \quad & y_i \leq x \quad i = 1, 2, 3, 4 \\ & y_i \leq i \quad i = 1, 2, 3, 4 \\ & x, y_1, y_2, y_3, y_4 \in \mathbb{N}. \end{aligned} \tag{4.2}$$

The deterministic equivalent can be solved using an ILP solver, giving as optimal solution $x = 3, y_1 = 1, y_2 = 2, y_3 = 3, y_4 = 3$. This result can be interpreted as a “two stage policy”, stating how many newspapers to order and how many to sell given a certain outcome of the demand.

Multistage Stochastic Programming: An Illustrative Example

Multistage stochastic optimisation is a generalisation of two-stage stochastic optimisation that involves more than two stages. This is suitable for synchronomodal transport because decisions have to be made at each time step that influence later decisions, but there is still uncertainty surrounding these later decisions. Multistage stochastic optimisation problems can be solved using backward dynamic programming, where each recursion step involves a two-stage stochastic program. This technique will be referred to as *multistage stochastic programming*. An applied example is given here. Note that an example was formulated where common sense supports the solution, so as to aid the reader, while in many actual instances, the optimal policy may not be easy to see without this technique.

Consider the instance of Problem 2 described in Fig. 4.4:

Three containers have to be sent from C to A. They have a deadline at $t = 3$, which is another way of saying they have a due date at $t = 3$ with arbitrarily large lateness penalty. One transit Idea departs at $t = 0$ from C to B, but its duration X is uncertain: it has probability 0.6 of being 1 and probability 0.4 of being 2. Another transit Idea, departing from B to A at $t = 2$, has uncertain price and capacity: with probability 0.7, they are both 2, and with probability 0.3, they are both 4. All other features are fixed. Admittedly, it is not common in practice to see that price and capacity are equal and random, but it is an allowed model within the general scope

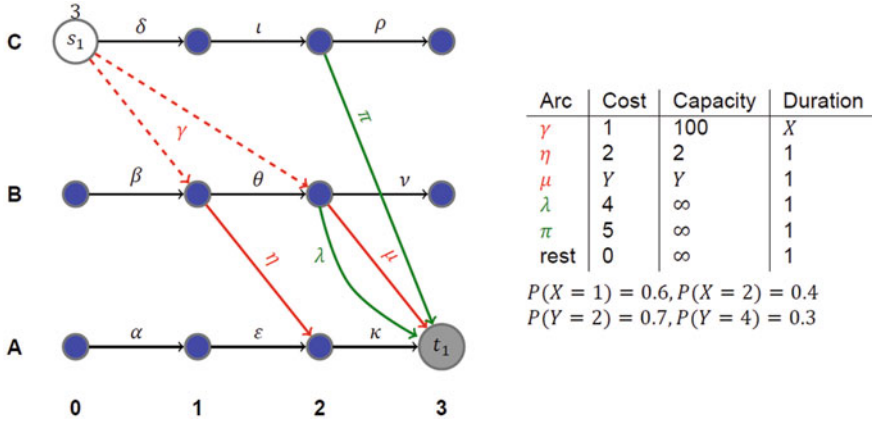


Fig. 4.4 A toy example of stochastic container-to-mode assignment, or Problem 2

of this chapter and serves for an example with manageable computation. Note that the transits corresponding to arcs λ and π represent the option of trucking at the last moment under a double matrix infinite resource model.

In this toy example, one can manually deduce that the optimal policy is to send all three containers over arc γ , whichever its end point may be, then preferring η over μ over λ . If the end point of γ is $s_{B,1}$, two containers can be sent over η , which is always at least as cheap as sending them over λ or μ . The third container then goes to $s_{B,2}$ by waiting. If the end point of γ is $s_{B,2}$, all three containers show up there. However, many containers are at $s_{B,2}$, as much as possible are sent over μ , as sending containers over μ is at least as cheap as sending them over λ . A container sent via this policy contributes at most 5 to the cost, which is always at least as cheap as sending it over π instead.

Now, it will be shown how the same result can be derived by solving a multistage stochastic problem by means of backward dynamic programming. This requires a definition of states. Note that in this example, there are four possible futures, denoted as follows:

$$\begin{aligned} \Phi_1 : (X, Y) &= (1, 2); & \Phi_2 : (X, Y) &= (1, 4); \\ \Phi_3 : (X, Y) &= (2, 2); & \Phi_4 : (X, Y) &= (2, 4). \end{aligned}$$

Further note that at any of the discrete-time moments, a container can be one of the six following “places”:

$$(unreleased, A, B, C, \text{on board of } \gamma, \text{ finished}).$$

If the source node in this example would have been at $s_{C,2}$ instead of $s_{C,0}$, the containers would have been at the location “unreleased” at time $t = 0$ and $t = 1$ and at location C at $t = 3$. In other words, containers are at location “unreleased”

before their release time; similarly, they are moved to the location “finished” as soon as they reach their due location, A. Further note that containers can only be at the location “on board of γ ” at $t = 1$, if $X = 2$.

Using all this, a system state can be fully described by a 6-dimensional vector v indicating where the containers currently are, an integer τ that represents the current time step, and a demifuture Ω_τ^ϕ that indicates both what has become certain so far and what uncertainties there still are. As such, let $S_\tau^\phi(v_1, v_2, v_3, v_4, v_5, v_6)$ denote the state that there are v_i containers at location i , $i = 1, 2, 3, 4, 5, 6$, that the current time is $t = \tau$, $\tau \in \{0, 1, 2, 3\}$, and that $\phi \in \{\Phi_1, \Phi_2, \Phi_3, \Phi_4\}$ is the true future, so the uncertainties are defined by the demifuture Ω_τ^ϕ . For efficiency, we will denote $\bar{v} = (v_1, v_2, v_3, v_4, v_5, v_6)$ and $\bar{v}' = (v'_1, v'_2, v'_3, v'_4, v'_5, v'_6)$.

Finally, denote $X_\tau^\phi(\bar{v})$ the multistage policy that, assuming that the system is in state $S_\tau^\phi(\bar{v})$, has minimal expected further cost $V_\tau^\phi(\bar{v})$ to get to the finishing state, where all containers are at location “finished”.

In this example, there are no more uncertainties at $t = 2$, regardless of which future is the true future. As such, $X_2^\phi(v)$ can be calculated easily for any container distribution v and future ϕ : this is a matter of solving deterministic container-to-mode assignment, using the current container locations as source nodes. In futures Φ_3 and Φ_4 , the containers on board of γ arrive at node $s_{B,2}$ and need to be absorbed into the system. One could thus solve problem 4.4 for futures Φ_3 and Φ_4 and problem 4.3 for futures Φ_1 and Φ_2 , where Y is always known.

$$\begin{aligned}
 V_2^\phi(\bar{v}) = \min \quad & \sum_{i=\kappa,\lambda,\mu,v,\pi,\rho}^n c_i x_i & (4.3) \\
 \text{s.t.} \quad & x_\kappa & = v_2 \\
 & x_\lambda + x_\mu + x_v & = v_3 \\
 & x_\pi + x_\rho & = v_4 \\
 & x_\kappa + x_\lambda + x_\mu + x_\pi & = 3 \\
 & x_\mu & \leq Y \\
 & x_\kappa, x_\lambda, x_\mu, x_v, x_\pi, x_\rho & \in \mathbb{N}
 \end{aligned}$$

$$\begin{aligned}
 V_2^\phi(\bar{v}) = \min \quad & \sum_{i=\kappa,\lambda,\mu,v,\pi,\rho}^n c_i x_i & (4.4) \\
 \text{s.t.} \quad & x_\kappa & = v_2 \\
 & x_\lambda + x_\mu + x_v & = v_3 + v_5 \\
 & x_\pi + x_\rho & = v_4 \\
 & x_\kappa + x_\lambda + x_\mu + x_\pi & = 3 \\
 & x_\mu & \leq Y \\
 & x_\kappa, x_\lambda, x_\mu, x_v, x_\pi, x_\rho & \in \mathbb{N}.
 \end{aligned}$$

Alternatively, it is easy to manually verify that all containers at C are transported over π , at most two containers at B are transported over μ , and a potential third container over λ , and nothing needs to be done with containers at A, so

$$V_2^{\Phi_1}(\bar{v}) = V_2^{\Phi_2}(\bar{v}_6) = 5v_4 + Y \cdot \min\{v_3, 2\} + 4 \cdot \max\{v_3 - 2, 0\}$$

$$V_2^{\Phi_3}(\bar{v}) = V_2^{\Phi_4}(\bar{v}) = 5v_4 + Y \cdot \min\{v_3 + v_5, 2\} + 4 \cdot \max\{v_3 + v_5 - 2, 0\}.$$

Note that, at $t = 2$, no containers can be at the locations “unreleased” or “finished”. This coincides with how the values v_1 and v_6 have no meaning in problems 4.3 and 4.4. The allowed states are only those where the three containers are divided as integers over locations A, B, and C, and “on board of γ ”, which can be done in $4 + 12 + 4$ ways. Therefore, using the simple computation above for all 20 allowed container distributions and all 4 futures, all $20 \cdot 4$ relevant values $V_2^\phi(0, v_2, v_3, v_4, v_5, 0)$ can be computed.

Next, the decision at $t = 1$ is examined. Herein lies a two-stage stochastic problem: the immediate costs made at $t = 1$ must be balanced against the expected costs $V_2^\phi(v)$ induced for $t = 2$. It is not yet known at $t = 1$, for example, if arc μ will have capacity and price 2 or 4. Note also that at $t = 1$, some containers might still be on board of γ if it turned out that $X = 2$. Therefore, in futures Φ_1 and Φ_2 , the decision problem is given in problem 4.5, and the slightly different decision problem for Φ_3, Φ_4 is given in problem 4.6.

$$V_1^\phi(\bar{v}) = \min \sum_{i=\varepsilon, \eta, \theta, \iota}^n c_i x_i + \sum_{j=1}^4 P(\Phi = \Phi_j | \Omega_1^\phi) V_2^{\Phi_j}(\bar{v}') \quad (4.5)$$

$$\begin{aligned} \text{s.t.} \quad x_\varepsilon &= v_2 \\ x_\eta + x_\theta &= v_3 + v_5 \\ x_\iota &= v_4 \\ x_\eta &\leq 2 \\ v'_1 &= 0 \\ v'_2 &= x_\varepsilon + x_\eta \\ v'_3 &= x_\theta \\ v'_4 &= x_\iota \\ v'_5 &= 0 \\ v'_6 &= 0 \\ x_i, v'_k &\in \mathbb{N} \quad i = \varepsilon, \eta, \theta, \iota, \quad k = 1, \dots, 6 \end{aligned}$$

$$\begin{aligned}
V_1^\phi(\bar{v}) = \min \sum_{i=\varepsilon,\eta,\theta,\iota}^n c_i x_i + \sum_{j=1}^4 P(\Phi = \Phi_j | \Omega_1^\phi) V_2^{\Phi_j}(\bar{v}') \quad (4.6) \\
\text{s.t.} \quad x_\varepsilon &= v_2 \\
x_\eta + x_\theta &= v_3 \\
x_\iota &= v_4 \\
x_\eta &\leq 2 \\
v'_1 &= 0 \\
v'_2 &= x_\varepsilon + x_\eta \\
v'_3 &= x_\theta \\
v'_4 &= x_\iota \\
v'_5 &= v_5 \\
v'_6 &= 0 \\
x_i, v'_k &\in \mathbb{N} \quad i = \varepsilon, \eta, \theta, \iota, \quad k = 1, \dots, 6.
\end{aligned}$$

There are several things to note about these two problems:

- When computing $V_1^{\Phi_1}(v)$, so when assuming Φ_1 will eventually turn out to be the true future and so at $t = 1$ it has become clear that $X = 1$, every future where $X = 2$ becomes impossible. This gives conditional probabilities $P(\Phi = \Phi_3 | \Omega_1^{\Phi_1}) = P(\Phi = \Phi_4 | \Omega_1^{\Phi_1}) = 0$. The outcome of Y is independent of X , so this gives the other conditional probabilities $P(\Phi = \Phi_1 | \Omega_1^{\Phi_1}) = P(Y = 2) = 0.7$, $P(\Phi = \Phi_2 | \Omega_1^{\Phi_1}) = P(Y = 4) = 0.3$. All other conditional probabilities can be computed analogously.
- $V_2^{\Phi_j}(v)$, in its current form, is not linear in its arguments, making the programs non-linear. Because integral arguments are expected and each argument must be at least 0 and the sum must equal 3, the amount of possible inputs v is finite. Therefore, the program can be made linear again by adding an indicator variable for each allowed input v using techniques for logic operations in linear programming [3]. Though this makes the problems linear again, the amount of added variables and constraints can be exponential in the amount of containers and locations.

Using these two observations, it is possible to find all values $V_1^\phi(v)$ through linear programming. Instead, it is manually observed here that getting containers from B to A is cheapest over arc η if possible, and this is the only arc that gives direct costs

in this stage, so, using $\check{v} = (0, v_2, v_3, v_4, v_5, 0)$,

$$V_1^{\Phi^1}(\check{v}) = 2 \cdot \min\{v_3 + v_5, 2\} + 0.7V_2^{\Phi^1}(0, v_2 + \min\{v_3 + v_5, 2\}, \max\{v_3 + v_5 - 2, 0\}, v_4, 0, 0) \\ + 0.3V_2^{\Phi^2}(0, v_2 + \min\{v_3 + v_5, 2\}, \max\{v_3 + v_5 - 2, 0\}, v_4, 0, 0)$$

$$V_1^{\Phi^2}(\check{v}) = 2 \cdot \min\{v_3 + v_5, 2\} + 0.7V_2^{\Phi^1}(0, v_2 + \min\{v_3 + v_5, 2\}, \max\{v_3 + v_5 - 2, 0\}, v_4, 0, 0) \\ + 0.3V_2^{\Phi^2}(0, v_2 + \min\{v_3 + v_5, 2\}, \max\{v_3 + v_5 - 2, 0\}, v_4, 0, 0)$$

$$V_1^{\Phi^3}(\check{v}) = 2 \cdot \min\{v_3, 2\} + 0.7V_2^{\Phi^3}(0, v_2 + \min\{v_3, 2\}, \max\{v_3 - 2, 0\}, v_4, v_5, 0) \\ + 0.3V_2^{\Phi^4}(0, v_2 + \min\{v_3, 2\}, \max\{v_3 - 2, 0\}, v_4, v_5, 0)$$

$$V_1^{\Phi^4}(\check{v}) = 2 \cdot \min\{v_3, 2\} + 0.7V_2^{\Phi^3}(0, v_2 + \min\{v_3, 2\}, \max\{v_3 - 2, 0\}, v_4, v_5, 0) \\ + 0.3V_2^{\Phi^4}(0, v_2 + \min\{v_3, 2\}, \max\{v_3 - 2, 0\}, v_4, v_5, 0).$$

This again yields 80 values $V_1^\phi(\check{v})$. Finally, the first decisions at $t = 0$ are determined in problem 4.7, which has only one feasible container distribution: namely, that all 3 containers have been released at location C.

$$V_0^\phi(0, 0, 0, 3, 0, 0) = \min \sum_{i=\alpha, \beta, \gamma, \delta}^n c_i x_i + \sum_{j=1}^4 P(\Phi = \Phi_j | \Omega_1^\phi) V_1^{\Phi_j}(\bar{v}') \quad (4.7)$$

$$s.t. \quad x_\alpha = v_2 = 0$$

$$x_\beta = v_3 = 0$$

$$x_\gamma + x_\delta = v_4 = 3$$

$$x_\gamma \leq 100$$

$$v'_1 = 0$$

$$v'_2 = x_\alpha$$

$$v'_3 = x_\beta$$

$$v'_4 = x_\delta$$

$$v'_5 = x_\gamma$$

$$v'_6 = 0$$

$$x_i, v'_k \in \mathbb{N} \quad i = \varepsilon, \eta, \theta, \iota, \quad k = 1, \dots, 6.$$

This program boils down to only one decision: how many of the three containers to send over γ , while sending the rest over δ . Therefore, only four values need to be checked:

- $0 \cdot 1 + \sum_{j=1}^4 P(\Phi = \Phi_j) V_1^{\Phi_j}(0, 0, 0, 3, 0, 0) = 15.$
- $1 \cdot 1 + \sum_{j=1}^4 P(\Phi = \Phi_j) V_1^{\Phi_j}(0, 0, 0, 2, 1, 0) = 13.24.$

- $2 \cdot 1 + \sum_{j=1}^4 P(\Phi = \Phi_j) V_1^{\Phi_j}(0, 0, 0, 1, 2, 0) = 11.48.$
- $3 \cdot 1 + \sum_{j=1}^4 P(\Phi = \Phi_j) V_1^{\Phi_j}(0, 0, 0, 0, 3, 0) = 10.64.$

The first value reflects the certainty that if all three containers are not sent over γ , they will be sent over π against cost 5. The second value reflects the certain cost of 10 for the two containers that do not travel over γ , plus the expected cost for the final container: 1 to traverse γ , then a further 2 against probability $P(X = 1) + P(X = 2, Y = 2) = 0.88$ or 4 against probability $P(X = 2, Y = 4) = 0.12$, so an expected cost of $1 + 0.88 \cdot 2 + 0.12 \cdot 4 = 3.24$. Similarly, the third value equals $5 + 2 \cdot 3.24$. Finally, the fourth value can be checked with the following computation: the first two containers each have expected cost 3.24, then the third container has cost 1 + 2 only in Φ_1 and 1 + 4 otherwise, so the third container has expected cost $0.42 \cdot 3 + 0.58 \cdot 5 = 4.16$ for a grand total expected cost of 10.64. Indeed, sending all containers over γ is the decision to take at $t = 0$ with the smallest expected cost.

Why Multistage Stochastic Programming Is Not Used

The example given in section “[Multistage Stochastic Programming: An Illustrative Example](#)” could be generalised to solve stochastic container-to-mode assignment to optimality. Dynamic programming largely halts the growth of complexity in the length of the time window: if the example had a hundred time steps instead of only four, each step would still only require the 80 values of the next time step and produce 80 values before terminating, due to the *property of state* [5].

Theorem Let $a \in \{0, 1, 2, \dots\}$ and $b \in \{1, 2, \dots\}$. The amount of ways to distribute a identical items over b recipients equals $\binom{a+b-1}{b-1} = \binom{a+b-1}{a}$. \square

Proof First, note that if $a = 0$, then the amount of distributions is 1: namely, all of the recipients get nothing. Indeed, the amount of distributions is then

$$\binom{a+b-1}{b-1} = \binom{b-1}{b-1} = 1.$$

Second, note that if $b = 1$, the amount of distributions is 1: namely, the one recipient gets all a items. Indeed, the amount of distributions is then

$$\binom{a+b-1}{a} = \binom{a}{a} = 1.$$

Now suppose $a \neq 0$, $b \neq 1$, so $a > 0$, $b > 1$. Denote C the amount of distributions of $a - 1$ items over b recipients. Denote D the amount of distributions of a items

over $b - 1$ recipients. It will be proven here that the amount of distributions of a items over b recipients equals $C + D$.

To this end, observe the distributions of a items over $b - 1$ recipients. How many distributions are there when another recipient is added? In each distribution, this new recipient either gets no items or gets at least one item. The amount of distributions in which it gets none of the items is the amount of distributions of a items over $b - 1$ recipients, so D . If the new recipient gets at least one item, the other $a - 1$ items must still be distributed over the b recipients, which can be done in C ways. So the amount of distributions of a items over b recipients equals $C + D$ (Fig. 4.5).

Observe Fig. 4.6. It is well known from the theory of Pascal’s triangle that this two-dimensional recursion, with the given base case, implies that the amount of distributions of a items over b recipients equals $\binom{a + b - 1}{b - 1} = \binom{a + b - 1}{a}$.

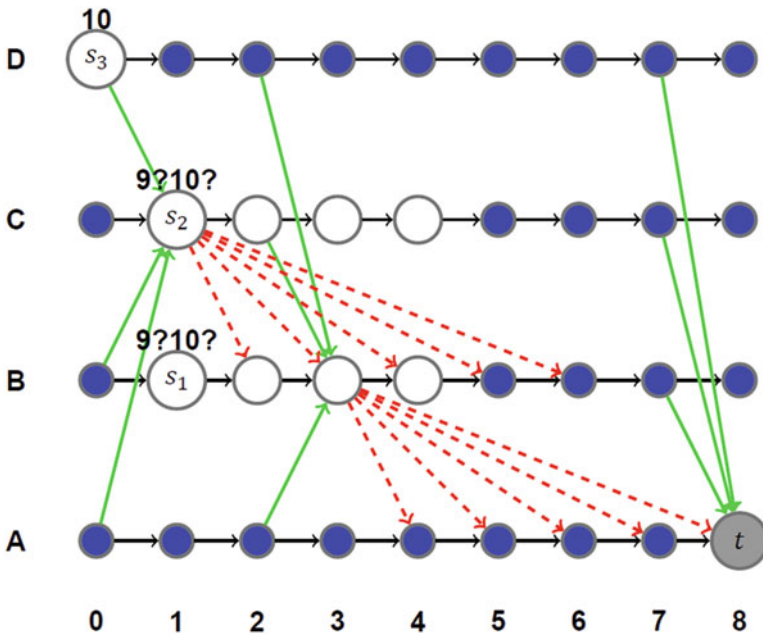


Fig. 4.5 An instance of Problem 2 that shows the explosive growth of the state space, the amount of possible system states per time step equals 11769142469427200. There are three request Ideas: one that is already revealed, the other two having 9 or 10 containers that are released at one of four possible times. The amounts are independent of the release time. All containers are due at the same place and time. There are two transit Ideas with uncertain travel time, one departing from $s_{B,3}$, the other from $s_{C,1}$; for the rest, it is possible to take a truck from any location to any other location in one time step to arrive just in time for the transit or the sink node. Though this instance is significantly smaller than “real life”, if the methodology from section “[Multistage Stochastic Programming: An Illustrative Example](#)” is applied, it would already involve having to keep track of values of over 10^{16} states, as explained in section “[Why Multistage Stochastic Programming Is Not Used](#)”

	$b = 1$	$b = 2$	$b = 3$	$b = 4$	$b = 5$	\dots
$a = 0$	1	1	1	1	1	\dots
$a = 1$	1	2	3	4	\dots	
$a = 2$	1	3	6	\dots		
$a = 3$	1	4	\dots			
$a = 4$	1	\dots				
\vdots	\vdots					

Fig. 4.6 The amount of distributions of a identical items over b recipients follows the behaviour seen in Pascal’s Triangle

However, the required amount of values per time step may equal the amount of possible futures times the amount of allowed container distributions. In the example, this was 4 times 20, or 4 times 28 when not manually discarding the locations “unreleased” and “finished”. In general, the amount of possible futures may equal the product of the amount of values that each stochastic element can take. Take, for example, the problem instance sketched in Fig. 4.5: there are only three request Ideas and two transit Ideas, one of the requests is already revealed at $t = 0$, the other request Ideas have release times X_1, X_2 , respectively, that may each take four different random values and an independent random volume Y_1, Y_2 of either 9 or 10 containers, and the transit Ideas have travel times Z_1, Z_2 , respectively, that may each take five different random values. Then that instance already has $4 \cdot 4 \cdot 2 \cdot 2 \cdot 5 \cdot 5 = 1600$ possible futures. If the instance concerns four real-life locations A, B, C, and D, so it concerns the 8 locations

(unreleased, A, B, C, D, on transit 1, on transit 2, finished),

and each request consists of 10 containers, then according to Theorem , there are

$$\binom{10 + 8 - 1}{10} = 19448$$

ways of distributing the 10 containers of the revealed request over the 8 locations, so if the others have 10 containers as well, $19448^3 = 7355714043392$ distributions of all containers over the locations. Multiplying this, the problem would have over

$1.176 \cdot 10^{16}$ or 11 quadrillion allowed states it could be in at each time step. In general, unless some smart reduction is found, multistage stochastic programming could require keeping track of

$$o \left(\prod_{\text{stochastic elements}} (\text{amount of possible values}) \prod_{\text{request } k} \binom{c_k + l - 1}{c_k} \right)$$

states, where c_k is the largest possible amount of containers in request k .

At each time step, for each state, a two-stage stochastic problem would have to be solved that can be made into an integer linear program at the cost of adding at least one variable and constraint for each state: thus, the growth of these integer linear programs is also exponential.

Long story short: unless some smart reduction is performed, the amount of states to keep track of grows uncontrollably, from 112 in Fig. 4.4 to 11769142469427200 in Fig. 4.5. For each of these states, an integer linear program with at least as many variables would have to be solved at each time step. This explosive growth of required space and time makes multistage stochastic programming unsuited for use beyond tiny instances.

Markov Decision Processes

The second way discussed here to solve Problem 2 is based on writing the decision process as a *Markov Decision Process*. This alternative to multistage stochastic programming appears to be more commonly discussed [7, 8]. In a Markov Decision Process (MDP), a number of actions can be taken in each state that yield a direct reward and move the system to another state with a certain probability. For examples, the reader is referred to White's survey [12].

In the context of this problem, one can again define a state based on a container distribution, a point in time, and an omnifuture specifying the uncertainties that are still left. The allowed actions are moving from one container distribution v_1 to another v_2 , which always succeeds, given that the modalities to do so are present. If containers are brought to their due location, they are immediately bounced to the "finished" location. Due to the assumption of locked futures, the probability that an action will move the system from state A with demifuture Ω_A to state B with demifuture Ω_B is exactly the conditional probability that state B has demifuture Ω_B given that its previous state has demifuture Ω_A . Rewards are exclusively non-positive: they correspond to the arc weights used in Chap. 3 for transportation costs and lateness penalties. See also Fig. 4.7.

Being able to write a decision process as a MDP depends on whether or not the decision process has the *Markov property* [9]: that probabilities and decisions depend only on the current state, not on previous ones. This property is closely related to the property of state mentioned in section "[Why Multistage Stochastic Programming Is Not Used](#)", and indeed, the methods appear related. These are at

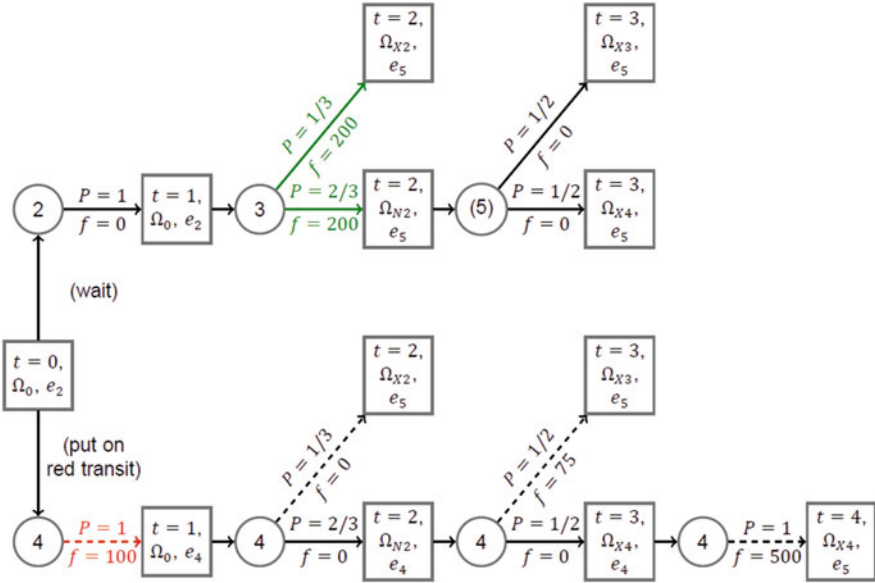


Fig. 4.7 The problem of Fig. 4.1 interpreted as a Markov Decision Process. There are five locations: (unreleased, A, B, on board of the uncertain transit, finished). There are five different demifutures: the initial omnifuture Ω_0 , the demifuture Ω_{X2} in which the transit certainly has travel time 2, the analogously defined Ω_{X3} and Ω_{X4} , and the demifuture Ω_{N2} , in which the transit certainly does not have travel time 2, but the probabilities that it is 3 or 4 are both 0.5. The states are given as rectangular nodes; a circular node with text i represents moving from the current container distribution to the container distribution e_i , that is to say, moving the container to location i . Note that there is only one decision with multiple feasible answers: whether, at $t = 0$, to commit the container to the uncertain transit or not. The Markov Decision Process ends when container distribution e_5 is attained, so when the containers reach location “finished”: in the case of state $t = 2; \Omega_{N2}; e_5$, it then turns into a Markov Process, in which Ω_{N2} is replaced by either Ω_{X3} or Ω_{X4}

least three ways in which using a MDP differs from using multistage stochastic programming:

- MDPs are often defined with a *discount factor* $\gamma \in [0, 1]$ that indicates how important immediate savings are when compared to expected future savings.
- Because of this discount factor, and because of the absence of backwards dynamic programming, MDPs lend themselves more naturally to rolling time window methods.
- MDPs have an action space with which to move around states, where multistage stochastic programming suggests moving from one state to another by means of two-stage stochastic programming. As a consequence of Theorem , the action space may have size $O(\prod_{commodity\ k} \binom{c_k + l - 1}{c_k})$, where c_k is the maximal amount of containers in request k and l is the number of locations.

It is noted that such a discount factor, within the context of synchromodal freight transport, is nice but not necessary. It was decided to limit this research to finite window methods, making the second difference also of lesser importance. The most important reason MDPs would be chosen over multistage stochastic programming would be if MDPs could be solved in significantly less time.

Several solution methods exist for MDPs [1]. However, in the context of freight transportation, it is a common conclusion that the state space and action space are both too large to solve MDP to optimality [7, 8]. Since this method uses almost the same enormous state space that is used in section “[Why Multistage Stochastic Programming Is Not Used](#)” and also an exponentially large action space, the same conclusion is drawn here without numeric evidence. Solving Problem 2 to optimality, using either multistage stochastic programming or MDPs, is only possible in tiny instances. When looking for a method that is fast enough for online use, it appears necessary to either smartly reduce the state spaces of these methods or use a heuristic or meta-heuristic.

Single Future Iteration Heuristics

The methods described in the previous section would probably not find solutions fast enough to be feasible in most practical applications. In particular, synchromodal planning depends on regular revaluations of the plannings, making it essential to formulate good plans in a relatively short amount of time. In this section, two basic ways of doing this are described.

Expected Future Iteration

If a barge has a travel time that is with equal probability 10, 11, or 12 time steps, then one could simplify the thought process by assuming the travel time will be its “average”, 11. Exactly this is the idea of computing an *expected future*: assume that all stochastic elements are independent, and then for every numeric stochastic element, compute its rounded expected value and work with that. See also Fig. 4.8.

For non-numeric stochastic elements, such as release location, one could randomly use any of the values with highest probability instead.

The idea of *Expected Future Iteration* (EFI) is the following:

1. Compute the expected future.
2. Interpret the expected future as an instance of deterministic container-to-mode assignment and solve it using the techniques from Chap. 3.
3. Use this solution to decide how to assign containers in the current time step. Enact these decisions, and then go to the next time step.

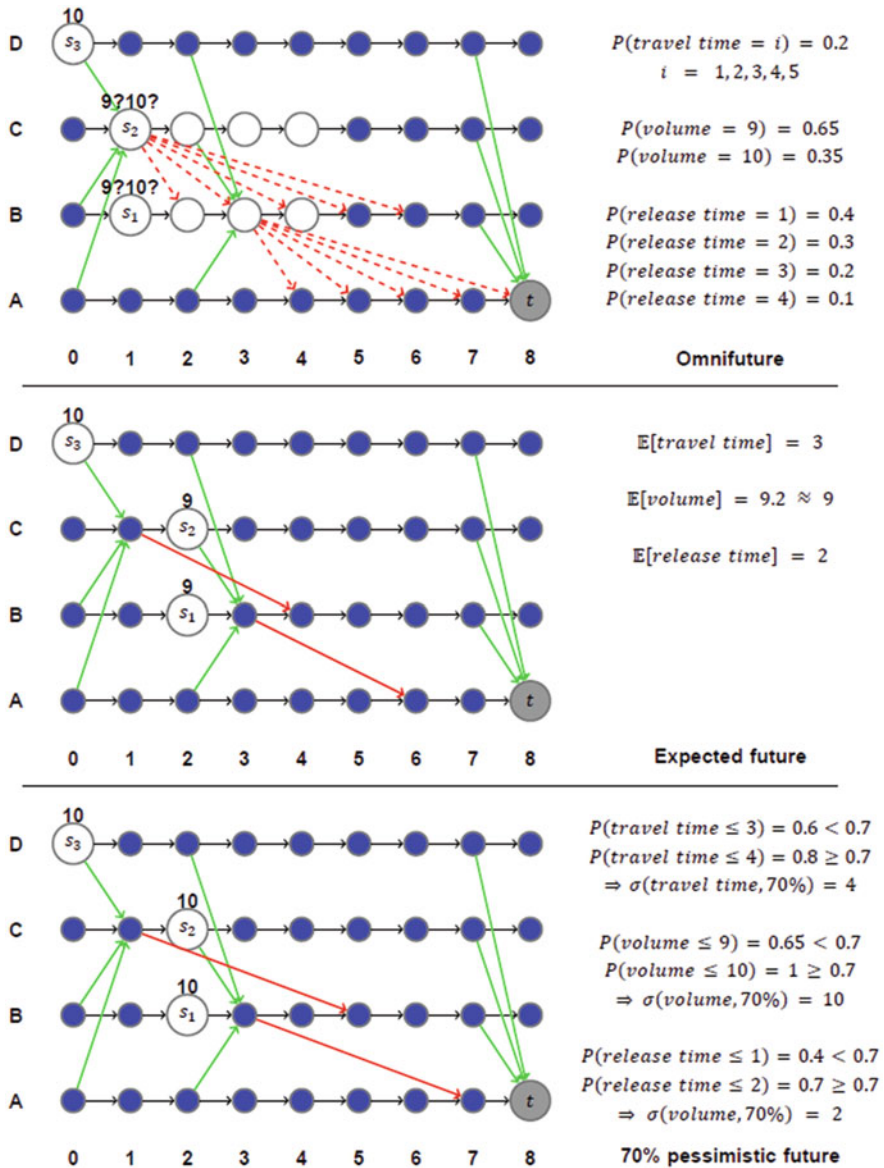


Fig. 4.8 At the top, the instance from Fig. 4.5. In the middle, its expected future, obtained by taking expected values for all numeric random variables. At the bottom, its 70% pessimistic future, obtained by taking all smallest values for which the probability is at least 70% that this value or a lower value is attained

4. Go back to 1., working with the current container locations and the demifuture of this new time step, that is to say, using all information that has become known up until this new time step.

This process is given in more detail in Algorithms 1 and 2. Note that Algorithm 2 was designed for testing purposes: it simulates the decision process that, assuming locked futures and finite windows, would take place if Algorithm 1 were used until all containers have reached their destinations. At the end of this process, the space–time network will be based on some true future: of this true future, the optimal container-to-mode assignment can be found using the techniques from Chap. 3, which then gives a lower bound on how well the decision process could have performed.

In practice, decisions would be made using only Algorithm 1, which relies less on the assumptions of locked futures and finite windows. In particular, the temporary assumption made at each time step that the time window is finite is not expected to be a large problem because when new requests do come in, they can simply be added to the next iteration. It may be interesting, however, to see if transportation should be “pulled” closer to the current time, so as to reserve capacity near the end of the time window for new requests: this and other questions concerning rolling time windows are proposed as future research.

Algorithm 1 Determining container-to-mode assignment for the current time step using Expected Future Iteration

Require: Omnifuture Ω , double matrix infinite resource matrices M^1, M^2

Ensure: Demand-to-resource assignment for current time step

- 1: **for** numeric stochastic variable **do**
 - 2: compute its expected value
 - 3: **for** non-numeric stochastic variable **do**
 - 4: obtain its mode, breaking ties randomly
 - 5: Create instance I of deterministic container-to-mode assignment by replacing all Ideas in Ω by instances with parameters equal to expected values and modes
 - 6: Obtain an optimal freight plan by solving I with the techniques from Chap. 3, given M^1, M^2
 - 7: From this solution, return all arcs that emanate from a node at time $t = 0$ and translate this back to a container-to-mode assignment at $t = 0$. Return this assignment.
-

Expected Future Iteration has several merits: it is conceptually simple, and it will be shown in section “[Numerical Results](#)” to give decent results in little time. It has one glaring downside, however: it fully trusts in the expected future and does not “optimise its plan B”, so if the actual values do not equal the expected values, the result can be arbitrarily bad. See also Fig. 4.9. Furthermore, the underlying assumption that all stochastic elements are independent may not always be a realistic assumption, especially when destinations are random. It would be better to somehow use the final elements of the Ideas, the joint probability distributions, but it is harder to define an expected value on these.

Algorithm 2 Simulating synchronodal decision-making over the entire time window using Expected Future Iteration. Designed for testing purposes

Require: Omnifuture Ω , true future Φ , double matrix infinite resource matrices M^1, M^2

Ensure: Synchronodal decisions over the entire time window

```

1: Initialise  $t = 0$ 
2: Initialise all containers at location ‘unreleased’
3: while there are still containers not yet at ‘finished’ do
4:   Obtain current demifuture  $\Omega_t^\Phi$ 
5:   for request in request Ideas do
6:     if request release time equals  $t$  according to  $\Phi$  then
7:       Move all containers of request from ‘unreleased’ to release location according to  $\Phi$ 
8:     if request released according to  $\Phi$  then
9:       for chunk of containers of request do
10:        if chunk at due location of request according to  $\Phi$  then
11:          Move chunk to location ‘finished’
12:        if chunk not at ‘finished’ then
13:          Interpret chunk as request instance with source node  $s_{v,t}$ , where  $v$  is the location of chunk and  $t$  the current time step
14:        Create instance  $I$  of stochastic container-to-mode assignment with the transit Ideas of  $\Omega_t^\Phi$  as transit Ideas and as request Ideas the request Ideas that are not yet released according to  $\Phi$  and the chunks interpreted as request instances
15:        Denote  $\Omega_I$  the omnifuture based on taking  $\Omega_t^\Phi$  and redefining the request Ideas as described
16:        Obtain a container-to-mode assignment for the current time step from Algorithm 1 ( $\Omega_I, M^1, M^2$ )
17:        Enact this assignment at the current time step
18:        Update  $t := t + 1$ 

```

Partially Pessimistic Future Iteration

The method described in this section could be viewed as a “safer” version of Expected Future Iteration. If X is a discrete random variable that may take values in the finite set $\chi \subset \mathbb{R}$, define

$$\sigma_{\leq}(X, \alpha) = \min\{x \in \chi \mid P(X \leq x) \geq \alpha\}$$

as the α -upper value of X and

$$\sigma_{\geq}(X, \alpha) = \max\{x \in \chi \mid P(X \geq x) \geq \alpha\}$$

as the α -lower value of X . For example:

$$P(X = 1) = P(X = 2) = P(X = 3) = P(X = 4) = P(X = 5) = 1/5$$

$$\Rightarrow \sigma_{\leq}(X, 70\%) = 4.$$

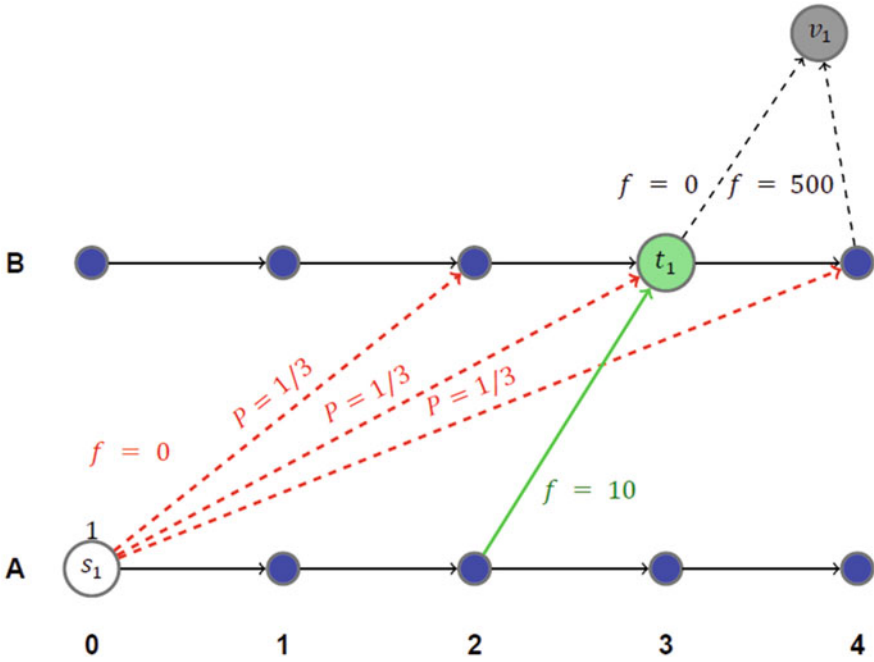


Fig. 4.9 An instance that shows how Expected Future Iteration can give arbitrarily bad results. If the transit with uncertain travel time is used, this will have cost 0 if the travel time is 2 or 3 but have the arbitrarily high cost of 500 if the travel time is 4. Expected Future Iteration will choose this option because it blindly assumes the travel time will be 3, the expected value. However, the expected cost of this option is $500/3$, whereas the expected cost of using the certain transit is only 10

Note that these values are defined for every $\alpha \in [0, 1]$: for example, $\sigma_{\leq}(X, 1) = \max\{x \in \chi\}$.

In general, if travel times turn out to be long, this is bad for minimising costs: it may cause containers to arrive at their destination late or it may cause expected transfers to be missed. In general, it makes the set of feasible modality paths smaller: every solution that uses a transit with long travel time has an equivalent solution for the case that the travel time was short, namely by taking the same transit and waiting out the difference, but the opposite is not true. Similarly, late request release dates are worse than early ones, but early due dates are worse than late ones.

Following this logic, Table 4.1 states for each potentially random parameter if it is bad for the value to be high or low.

Then, α -Pessimistic Future Iteration (α PFI) is exactly the same as Expected Future Iteration, except that instead of taking expected values for numeric random variables, the α -upper bound is taken instead if Table 4.1 states that it is bad for the value to be high, and the α -lower bound is taken if it is bad for the value to be low. See also Fig. 4.8.

Table 4.1 For each potentially random parameter, if it takes numeric values, whether it is generally bad for total transportation costs when the value is high or rather when it is low

Transit departure location: non-numeric	Request release location: non-numeric
Transit destination: non-numeric	Request due location: non-numeric
Transit capacity: bad when low	Request volume: bad when high
Transit departure time: bad when high	Request release date: bad when high
Transit travel time: bad when high	Request due date: bad when low
Resource price: bad when high	Request lateness penalty function: non-numeric
Terminal handling time: bad when high	Container-to-mode assignment: non-numeric

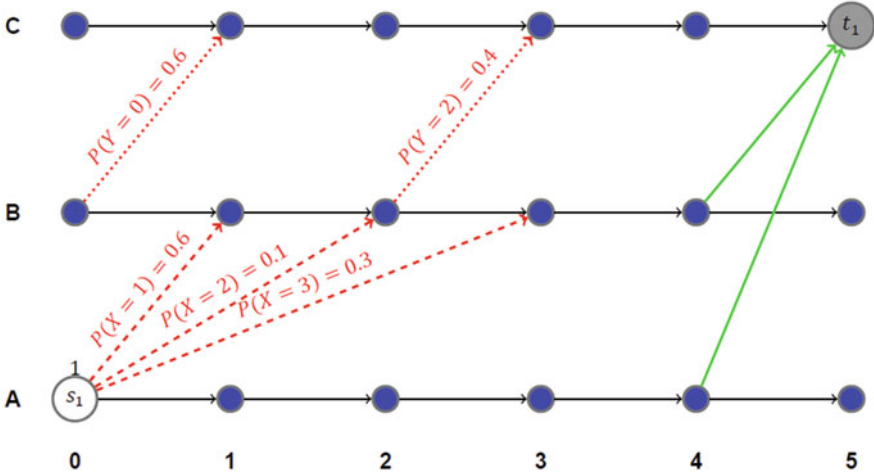


Fig. 4.10 An instance with two uncertain transits: one, dashed, from A to B that has uncertain travel time X ; another, dotted, from B to C that has uncertain departure time Y . X and Y are independent. $\sigma(70\%, X) = 2, \sigma(70\%, Y) = 2$, so in the 70% pessimistic future, $X = 2$ and $Y = 2$ so it is possible to take the first transit and follow it up with the second. However, the probability that this modality path indeed exists is only $P(X \leq 2, Y = 2) = 0.28$, not 0.7. Note also how in the expected future, $X = 2$ and $Y = 1$, so expected future does not equal the 50% pessimistic future

Note that no simple robustness conclusions can be drawn from using this method: for example, if a modality path exists in the 70% pessimistic future, this does not have to mean that the path has a probability of at least 70% of existing, as demonstrated in Fig. 4.10. The same figure illustrates how the expected future is not necessarily the same thing as the 50% pessimistic future. Quantifying or correcting this dissonance, by reformulating the method to be based on the probability that a used path exists and using this for robust optimisation, could be an insightful topic of future research.

Note also that α -Pessimistic Future Iteration retains the property of allowing for arbitrarily bad solutions, which can be seen by modifying Fig. 4.9. Still, intuition

dictates that the higher the pessimism parameter α is set, the more robust the solution will become, at the cost of discarding cheaper but riskier solutions.

Numerical Results

In this section, the methods of Expected Future Iteration and 70%-Pessimistic Future Iteration are tested on several random instances, under the same setup as in section “Numerical Results”. If the true future is known completely at the start, one could solve for that future to attain the perfect assignment. The two tested methods discover this true future gradually, and one can only hope that the decisions they make while doing so are indeed the correct decisions. Understanding this, it is possible to give a tight lower bound on what the objective value could have been and thus to express the efficiency of a decision process. This is one of the two results produced in this test.

The other result concerns computing time, rather than efficiency. In the tests, the iteration is performed until the end of the time window: however, in operational practice, one would only run the first iteration, seeing as one cannot iterate over a future that is not yet known. The relevant computing time, therefore, is the average computing time for the first iteration only. All results are shown in Table 4.2.

Table 4.2 Numerical results for stochastic container-to-mode assignment, or Problem 2. Expected Future Iteration and 70%-Pessimistic Future Iteration are abbreviated to EFI and 70PFI, respectively. Class 4 contains instances of a size that was specified as useful within an operational use case

		Class 1	Class 4
EFI cost over optimal	Mean	3.51%	3.54%
	Variance	28.52% ²	7.76% ²
	Worst	14.48%	8.56%
Time for first step	Mean	1.04 s	23.98s
	Variance	0.02 s ²	1.64 s ²
	Worst	1.44 s	26.35 s
70PFI cost over optimal	Mean	4.43%	3.34%
	Variance	38.41% ²	7.57% ²
	Worst	20.55%	8.67%
Time for first step	Mean	1.09 s	25.60 s
	Variance	0.02 s ²	2.45 s ²
	Worst	1.49 s	27.36 s

Discussion

Despite the relative naiveté of Expected Future Iteration and 70%-Pessimistic Future Iteration, the results in section “[Numerical Results](#)” show that both methods, in the testing environment, achieve quite good results in not too much time. More precisely: on problems of class 4, so “real life size” problems, they found assignments with costs that were 3.34–3.54% over the optimum on average and 8.56–8.67% in the observed worst cases. In practice, so making one decision per hour, the time to take this decision with either method was not observed to exceed thirty seconds.

By design, one would expect 70%-Pessimistic Future Iteration to achieve lower worst case costs than Expected Future Iteration, but higher average costs; surprisingly, one might say that these numerical results argue the opposite. It is unclear why this occurs, but two explanations are suggested:

- The two methods are structurally more different than they seem, which also relates to how Expected Future Iteration does not map exactly to 50%-Pessimistic Future Iteration, as observed earlier: as such, they cannot be compared well on the gradient of risk versus robustness.
- The random generation of instances may not have included enough situations in which the short-sightedness of single future iteration heuristics truly creates problems, not allowing 70%-Pessimistic Future Iteration to “shine” in this field.

Overall, there are several caveats to be addressed about these methods and results. For one, both methods have the theoretic property that they may yield arbitrarily bad results, although no spectacular excesses of cost were observed. Second, though the method of random instance creation may suit some chaotic problems well, it is difficult to promise upfront how efficiently these methods will perform on problems with more structurally defined system dynamics. For example, day and night cycles were not simulated in the environment, while these may cause great delays: if Expected Future Iteration blindly assumes that goods will come in just before a terminal’s closing time, the consequences of arriving slightly late are major, and methods that consider such consequences may be more suitable. This second downside may well be solved by using the simulation method of Kooiman [7], though this requires solving many instances of Problem 1 during the decision process, which may add up to a decision process that would currently not be fast enough for online use. Third, in practice, probability distributions for stochastic elements may likely be unknown: the developed methods depend greatly on reliable forecasting methods being available.

Added Value

Stochastic container-to-mode assignment has already been studied under different forms and names, even in contexts that follow synchronodal paradigms [7, 8, 14]. However, none of the encountered literature assumes the holistic stance of this research, in viewing almost anything as potentially stochastic. Such holism may be essential in moving towards a synchronodal setting that encompasses the different dynamics of different parties. As a consequence, this research has formalised some concepts necessary to build these generalistic foundations, in the form of Ideas and omnifutures. Furthermore, this chapter has verified that reasonable solutions can be found for the stochastic container-to-mode assignment problem fast enough for operational decision-support use.

Conclusion

This chapter sought to answer the following research question:

How can a low-cost net-centric container-to-transport assignment be found fast enough for online use if new data is still expected to come in?

Stochastic container-to-mode assignment, or “Problem 2”, was defined to be the problem of assigning freight containers optimally to transport modalities with pre-determined schedules, in the presence of time constraints and stochastic elements. It required the definition of not a singular request but a request Idea that generates different instances of the request against different probabilities. Similarly, transits were replaced by transit Ideas, and omnifutures were designed as objects that generate all possible futures.

It was illustrated that this problem can be solved to optimality using multistage stochastic programming, but also argued that multistage stochastic programming requires far too much computing time beyond tiny instances. Instead, two decision process heuristics were introduced: Expected Future Iteration, which at each time step assumes that all remaining unknowns will take on their marginalised expected values, and $\alpha\%$ -Pessimistic Future Iteration, which instead assumes the unknowns will take on a variant of their “ α -percentile value”.

Random instances of different sizes were generated to compare these two methods. They show that assignments for the current time step can be found for problems of a “real life size” in average 23.98 and 25.60 s, respectively, even when using a non-commercial MILP solver on a single computer. This makes the method suitable for online use. The numerical results suggest that the methods will find assignments with costs that are on average 3.54% and 3.34% above optimal, respectively.

To answer the sub-question: using either Expected Future Iteration or 70%-Pessimistic Future Iteration serves as a near-optimal decision policy as information

becomes known. Both methods run fast enough for online problems of “real life size”.

References

1. Arapostathis, A., Borkar, V. S., Fernández-Gaucherand, E., Ghosh, M. K., & Marcus, S. I. (1993). Discrete-time controlled Markov processes with average cost criterion: A survey. *SIAM Journal on Control and Optimization*, 31(2), 282–344.
2. Behdani, B., Fan, Y., Wiegmans, B., & Zuidwijk, R. (2014). Multimodal schedule design for synchromodal freight transport systems. *European Journal of Transport & Infrastructure Research*, 16(3), 424–444.
3. Brown, G. G., & Dell, R. F. (2007). Formulating integer linear programs: A rogues’ gallery. *INFORMS Transactions on Education*, 7(2), 153–159.
4. Caris, A., Macharis, C., & Janssens, G. K. (2008). Planning problems in intermodal freight transport: Accomplishments and prospects. *Transportation Planning and Technology*, 31(3), 277–302.
5. Chinneck, J. W. (2006). *Practical optimization: A gentle introduction. Systems and computer engineering*. Carleton University. <http://www.sce.carleton.ca/faculty/chinneck/po.html>
6. Dana Jr., J. D., & Petruzzi, N. C. (2001). Note: The newsvendor model with endogenous demand. *Management Science*, 47(11), 1488–1497.
7. Kooiman, K. (2015). *A classification framework for time stamp stochastic assignment problems and an application to inland container shipping*. TNO Internal Documentation.
8. Pérez Rivera, A., & Mes, M. (2016). Service and transfer selection for freights in a synchromodal network. *Lecture Notes in Computer Science*, 9855, 227–242.
9. Puterman, M. L. (2014). *Markov Decision Processes: Discrete stochastic dynamic programming*. Wiley.
10. Reeve, C. (1998). *Plato, Cratylus*. Indianapolis/Cambridge.
11. Sumalee, A., Uchida, K., & Lam, W. H. (2011). Stochastic multi-modal transport network under demand uncertainties and adverse weather condition. *Transportation Research Part C: Emerging Technologies*, 19(2), 338–350.
12. White, D. J. (1993). A survey of applications of Markov Decision Processes. *Journal of the Operational Research Society*, 44(11), 1073–1096.
13. Xu, Y., Chen, Q., & Quan, X. (2012). Robust berth scheduling with uncertain vessel delay and handling time. *Annals of Operations Research*, 192(1), 123–140.
14. Zhang, M., & Pel, A. (2016). Synchromodal hinterland freight transport: model study for the port of Rotterdam. *Journal of Transport Geography*, 52, 1–10.

Chapter 5

Deterministic Operational Freight Planning



D. Huizing

Abstract In this chapter, the final of three problems is presented. Now, there are no stochastic elements anymore; however, the decision-maker is given control over the vehicle timetables in addition to the control over container-to-mode assignments. This problem is argued to be a departure from classical optimisation problems, but shown to still be strongly NP-hard. An integer linear program is developed to solve the problem, but the results show that it scales too poorly to solve problems of “real life size” in an appropriate amount of time for decision support. The Greedy Gain heuristic and Compatibility Clustering heuristic are developed: they solve much more limited sub-problems with the ILP, but unfortunately, even these sub-problems require too much computational effort at the wished instance size.

Introduction

As last of the three examples, we focus on the question how a low-cost net-centric operational transport schedule can be found fast enough for online use if everything is known beforehand? In the two previous chapters, problems were studied where the timetables of barges and other resources were already fixed and only the container-to-mode assignment had to be optimised. The former chapter observed the deterministic case of this problem, and the latter the stochastic case.

In this chapter, however, the decision-maker also fully or partially controls the timetables of the transports. The decision-maker can decide where transports will go and when. The optimisation, now, lies in simultaneous decision-making of fleet routing in space-time and container-to-mode assignment. The final problem is the fast optimisation of *deterministic operational freight planning* (Problem 3), by which is the following $\bar{R}, [RD], [RDT]|\bar{D}, [D2R]|social(1)$ -problem is meant: to simultaneously determine transport routes, determine transport departure times, assign freight containers to transports, and determine when and where to load

D. Huizing (✉)
TNO, The Hague, The Netherlands

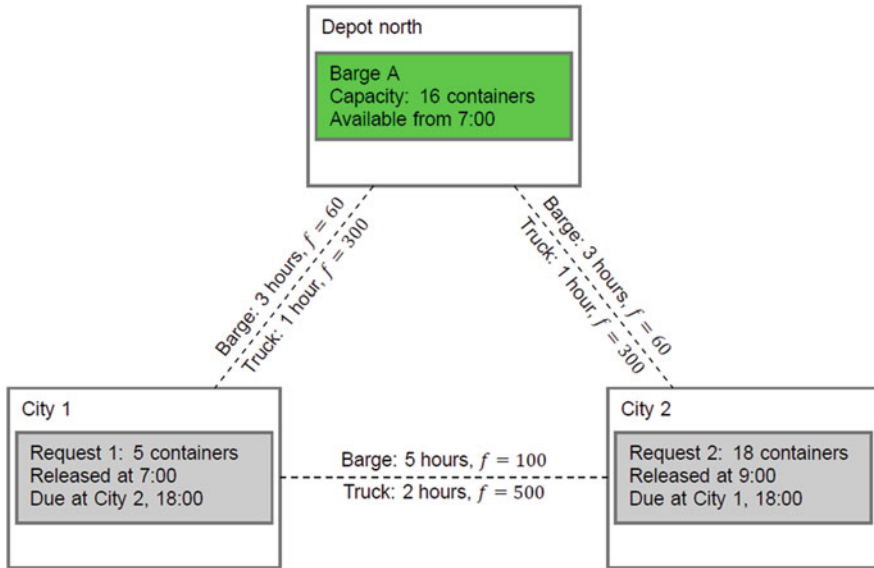


Fig. 5.1 Two requests are placed: Request 1 consists of 5 containers that appear at City 1 and are due at City 2, and Request 2 consists of 18 containers the other way round. One barge with a capacity of 16 containers becomes available at the depot at 07:00. Completely loading or unloading the barge at a terminal always costs one hour. Under the assumption of double matrix truck modelling, any amount of trucks can be deployed at any time against given costs and travel times. The problem here is how to assign containers to either the barge or trucks and how to move the barge around, knowing that trucks are always more expensive but there is not enough time to do everything by barge. Inspection shows that the optimal solution involves transporting 16 containers of Request 2 by only barge and all other containers by only truck

and unload said containers, so that the containers reach their destinations before a deadline against minimum total cost, given that all features of the problem are deterministic. Again, trucks or subcontracted transportation can either be explicitly modelled, left out, or modelled as double matrix infinite resources, as detailed in section “[Infinite Resource Models and the Corresponding Graph Reductions](#)”. A finite time window with finite discretisation is again observed. An example of this problem is given in Fig. 5.1. In the language of the previous chapters, Problem 3 could also be interpreted as follows: how can transit links be placed in a space–time network, respecting travel time and terminal handling time, such that the cost of the link placement and the corresponding minimum-cost multi-commodity flow together are minimal? See also Fig. 5.2.

This problem has direct applications in practice: for the optimisation of a social synchronomodal network of different parties, but also for the more current problem of real-time planning within the transport chain of one party. However, these applications hold only if all stochasticity is discarded. The stochastic version of this problem may better suit operational reality but is left as future research. The techniques developed here for the deterministic case may form the foundation for

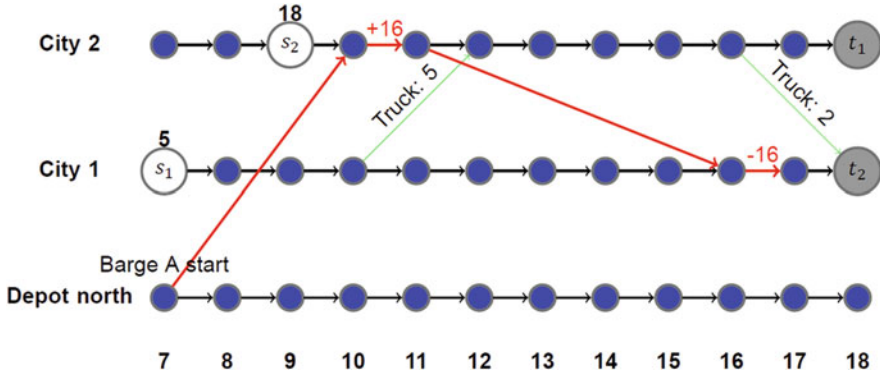


Fig. 5.2 One of the optimal solutions of the problem in Fig. 5.1 represented in a space–time network. Barge A, represented by the red arcs, starts at Depot north (7:00), departs (7:00), arrives at City 2 (10:00), loads 16 containers of Request 2, departs (11:00), arrives at City 1 (16:00), and unloads the 16 containers of Request 2. At 16:00, the remaining 2 containers of Request 2 are trucked. At 10:00, all 5 containers of Request 1 are trucked. Note that the moments of trucking are quite arbitrary, leading to many similar optimal solutions. The fact that the red barge could choose to arrive one time step later at City 2 or City 1 also creates similar solutions

such research of the stochastic case, which is the final reason this problem is studied here.

Notation of Variables and Parameters

The reader is reminded that controlled barges, trains, and the like will be referred to as *vehicles*, while separate truck departments and subcontracted transportation will be referred to as *infinite resources* with infinite capacity but typically high cost. The large and varied decision space of this problem merits a structured introduction of the variables used in this chapter, denoted with lower case letters, before any further discussion:

- $x_{w,k,t} \in \mathbb{N}$: the amount of containers from request k that vehicle w has on board at the start of time t .
- $y_{i,k,t} \in \mathbb{N}$: the amount of containers from request k present at location i at the start of time t .
- $z_{w,i,t} \in \{0, 1\}$: indicator variable that equals 1 if and only if vehicle w is present at location i at the start of time t .
- $a_{w,i,t} \in \{0, 1\}$: indicator variable that equals 1 if and only if vehicle w arrives at location i just before the start of time t .
- $b_{w,i,j,t} \in \{0, 1\}$: indicator variable that equals 1 if and only if vehicle w departs from location i to location j just after the start of time t .
- $r_{w,t} \in \mathbb{N}$: the amount of remaining travel time vehicle w has at the start of time t .

- $h_{w,i,t} \in \{0, 1\}$: indicator variable that equals 1 if and only if vehicle w stays at location i throughout time t to handle containers.
- $l_{w,i,k,t} \in \mathbb{N}$: the amount of containers from request k that are loaded from location i onto vehicle w during time t .
- $u_{w,i,k,t} \in \mathbb{N}$: the amount of containers from request k that are unloaded from vehicle w onto location i during time t .
- $q_{i,j,k,t} \in \mathbb{N}$: the amount of containers from request k that are sent by infinite resource from location i to location j , departing just after the start of time t .
- $v_{i,j,k,t} \in \mathbb{N}$: the amount of containers from request k that arrive at location j just before the start of time t , having been sent by infinite resource from location i .

Seeing how the departure variables $b_{w,i,j,t}$ are indexed on vehicle, origin, destination, and departure time, one may as well allow for the travel time incurred to also depend on vehicle, origin, destination, and departure time. Travel times can thus be defined as parameters $T_{w,i,j,t}$. If an instance has symmetric travel times that depend only on the origin–destination pair, one can simply set the values such that $T_{w,i,j,t} = T_{i,j} = T_{j,i} \quad \forall(w, i, j, t)$. Following this philosophy of generality, the parameters of this problem, denoted with upper case letters, are defined as follows, with $\mathbb{N}_+ = \{1, 2, 3, \dots\}$:

- $E_{w,i,j,t} \in \mathbb{N}$: cost incurred if vehicle w travels from location i to location j , departing just after the start of time t .
- $F_{w,i,k,t} \in \mathbb{N}$: unit cost of loading containers from request k from location i onto vehicle w during time t .
- $G_{w,i,k,t} \in \mathbb{N}$: unit cost of unloading containers from request k from vehicle w onto location i during time t .
- $T_{w,i,j,t} \in \mathbb{N}_+$: travel time if vehicle w travels from location i to location j , departing just after the start of time t .
- $P_{w,i,t} \in \mathbb{N}$: processing speed, or the total amount of containers that vehicle w can load or unload at location i during time t .
- $N_{i,t} \in \mathbb{N}$: the total amount of containers that can be loaded or unloaded at location i during time t .
- $C_{w,t} \in \mathbb{C} \mathbb{N}_+$: the total capacity of vehicle w at the start of time step t .
- $D_k \in \mathbb{N}_+$: volume of request k .
- $O_{i,j,k,t} \in \mathbb{N}$: unit cost of using infinite resources to move containers from request k from location i to location j departing just after the start of time t .
- $S_{i,j,k,t} \in \mathbb{N}_+$: the amount of time steps required before containers from request k arrive at location j , having been sent by infinite resource from location i just after the start of time t .

Problem 3, then, is finding a feasible solution that minimises the sum over the vehicle travel costs $E_{w,i,j,t}b_{w,i,j,t}$, the loading costs $F_{w,i,k,t}l_{w,i,k,t}$, the unloading costs $G_{w,i,k,t}u_{w,i,k,t}$, and the costs $O_{i,j,k,t}q_{i,j,k,t}$ incurred by using infinite resources. The definition of “feasible” will be more thoroughly discussed in section “ILP

Formulation”, but it largely follows such intuitions as “vehicles may only unload at a location if they are present at the location” and “vehicles may unload at most as many containers as they have on board”. The discussion up until that section should be clear enough to follow without the formal definition.

As in section “**Allowing Lateness with Virtual Sinks**”, one can define a soft due date for a request, next to a hard deadline, by setting appropriate values for the time-dependent costs $G_{w,i,k,t}$ of unloading at the due location.

Notably absent from these parameters are, for example, release time and deadline of a request. In this model, it is assumed that if a request k is not released yet, all of its containers are present but stuck at the release location. So for every time step t before the release time, $y_{\text{release-location},k,t} = D_k$ and $y_{\text{not-release-location},k,t} = 0$. Similarly, for every time step t from the deadline onwards, $y_{\text{due-location},k,t} = D_k$ and $y_{\text{not-due-location},k,t} = 0$. By enforcing these variables to have these values, the behaviour of release times and deadlines is achieved. Therefore, the final “parameter” used in this model is a set Δ of fixed value pairs: for example, $(x_{\text{Barge } A, 101, 0}, 2) \in \Delta$ denotes that $x_{\text{Barge } A, 101, 0} = 2$ must hold, so that the vehicle *Barge A* has 2 containers from request 101 on board at the start of time 0. If $\Delta = \emptyset$, no variables are enforced to have fixed values.

These fixed value pairs in Δ are important to model the following requirements, among others:

1. At the first time step, vehicles and containers can be at any place undergoing any activity. Δ must enforce the starting situation on any variable.
2. In particular, if using an infinite resource during the time window to transport containers from request k from location i to location j can get them there as early as some time τ according to parameters $S_{i,j,k,t}$, then $v_{i,j,k,t}$ has a fixed value until τ . This fixed value is only greater than 0 if transport was started some time before the start of the time window.
3. As discussed, the requests have release times and deadlines and Δ must enforce values D_k and 0 on variables $y_{i,k,t}$.
4. If a terminal i has closing times, then Δ must enforce $z_{w,i,t} = 0$ for every vehicle w if i is not open at time t .
5. If a terminal has specified the remaining time slots in which vehicles can still be handled, Δ must enforce $z_{w,i,t} = 0$ outside of these time slots.
6. If a terminal i cannot service barges perhaps because it has no water connection, then Δ must enforce $z_{w,i,t} = 0$ for every vehicle w if w is a barge and for every time t .

Note that in item 4, one could also choose to enforce that $a_{w,i,t} = 0$ rather than $z_{w,i,t} = 0$, seeing how a vehicle cannot be at a location without arriving there and vice versa. In every such case, it is encouraged to fix both values, to speed up the computation in the case of integer linear programming and to make the boundaries of the decision space more explicit.

Note finally that, in this formulation, the amount of containers a vehicle can load or unload in a time step is not request-dependent, and the amount of containers a terminal can load or unload depends on neither requests nor vehicles. This may not

exactly represent operational reality, but improvements would require some concept of the amount of “work” loading or unloading certain containers require, which is left as a topic of future research.

With a notation for this problem introduced, the discussion on the features and solution methods of this problem can continue more efficiently.

Problem Features

This problem is similar to the Capacitated Multi-Commodity Network Design (CMCND) problem investigated by Pedersen [6]. It is quite different, however, from the more well-known dial-a-ride problem (DARP) [1] and other Vehicle Routing Problems (VRP) [8], in two important senses:

- In intermodal transportation, it may be optimal for goods to be picked up by one vehicle and dropped off by another, changing vehicles along the way any amount of times. However, most DARP formulations assume that the entire journey from pickup to drop-off is performed by one vehicle.
- In Problem 3, it is not necessary for vehicles to start and end at some depot location. In fact, the real-time flexibility property of synchromodal planning demands that in the starting situation, vehicles and containers can be at any location undergoing any type of action. However, most VRP formulations assume that vehicles start and end at some depot.

Therefore, Problem 3 is a departure from many classical optimisation problems. It would seem prudent, thus, to supply a proof why it is still a strongly NP-hard problem. This can be done by a reduction from 3-partition, which is known to be a strongly NP-complete problem [3].

Theorem *Deterministic operational freight planning is a strongly NP-hard optimisation problem.* □

Proof First, observe the decision variant of deterministic operational freight planning: given some instance of deterministic operational freight planning, does it have a feasible solution with cost smaller than or equal to some threshold value Y ? This problem is obviously in NP:

- Solutions of this problem can be encoded in polynomial time and space with respect to the input size: the variables, described in section “[Notation of Variables and Parameters](#)”, are polynomially many in the amount of vehicles, locations, requests, and time steps.
- Whether or not a solution gives YES to the decision problem, can be checked in polynomial time and space with respect to the input size: this is merely a matter

of computing whether

$$\begin{aligned} & \sum_{w,i,j,t} E_{w,i,j,t} b_{w,i,j,t} + \sum_{w,i,k,t} F_{w,i,k,t} l_{w,i,k,t} + \sum_{w,i,k,t} G_{w,i,k,t} u_{w,i,k,t} \\ & + \sum_{i,j,k,t} O_{i,j,k,t} q_{i,j,k,t} \leq Y. \end{aligned}$$

Now, take any instance I_1 of 3-partition: thus, take any $m, A_1, A_2, \dots, A_{3m}, B \in \mathbb{Z}$ such that $\sum_{j=1}^{3m} A_j = mB$, and let I_1 be the decision problem “Can the list of numbers A_1, A_2, \dots, A_{3m} be partitioned into m triplets S_i such that $\sum_{j \in S_i} A_j = B$ for $i = 1, 2, \dots, m$?”

Next, construct the instance I_2 of deterministic operational freight planning illustrated in Fig. 5.3 as follows.

Let there be m homogeneous vehicles $\{1, 2, \dots, m\}$ with capacity B . Let there be $3m$ requests named $1, 2, \dots, 3m$ respectively, with volume $D_j = A_j$ for $j = 1, 2, \dots, 3m$. Let there be $3m + 1$ locations named $0, 1, 2, \dots, 3m$, respectively. Let the time steps be $0, 1, 2, \dots, 8$. Employ Δ as follows:

- Let all vehicles start empty.
- Let request k have release time 0, release location k , deadline 7, and due location 0 for $k = 1, 2, \dots, 3m$.
- Let each vehicle w start at location 0, with $r_{w,0} = 0$ and $a_{w,i,0} = 0$ for each location i .
- Let $v_{i,j,k,0} = 0$ for each location i , each location j , and each request k , which will imply that no infinite resources were called upon before time 0 that effect this time window.

Let all vehicle travel costs $E_{w,i,j,t}$, all loading costs $F_{w,i,k,t}$, and all unloading costs $G_{w,i,k,t}$ equal 0. Let all infinite resource costs $O_{i,j,k,t}$ equal 1. Let all vehicle travel times $T_{w,i,j,t}$ and all infinite resource delivery times $S_{i,j,k,t}$ equal 1. Let $P_{w,i,t} = C_w = B$ and $N_{i,t} = mB$ for each vehicle w , each location i , and each time step t , which implies that loading and unloading at a location always cost 1 time step, regardless of the amount of containers. With all parameters set, note that each vehicle can fully or partially service at most three different requests, by performing this sequence of actions that each cost one time step:

*(go to first pickup, load, go to second pickup, load,
go to third pickup, load, go to 0, unload all).*

Note also that for any solution, the objective value has costs equal to the amount of containers transported anywhere by infinite resource.

Finally, let I_3 be the decision problem “Does I_2 have a solution with value less than or equal to 0?” Clearly, decision problem I_3 can be constructed from decision

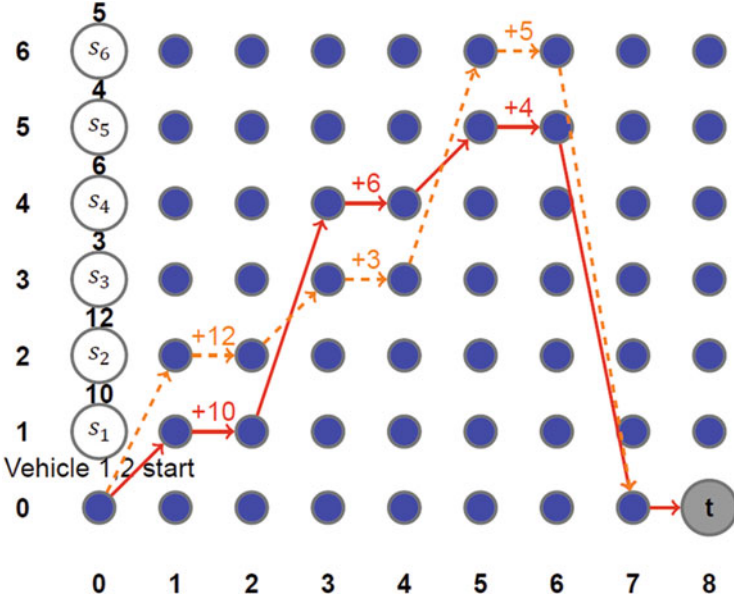


Fig. 5.3 A special instance of deterministic operational freight planning can be based on a random instance of 3-partition by giving each item of the partition problem its own request and its own location for release and letting all requests be due at time 8 at location 0. At location 0, m vehicles with capacity B start empty at time 0. Travelling, loading, and unloading always cost one time step and have cost 0. Using infinite resources always costs one time step and has cost 1. A resource, thus, has time to take care of at most three requests. This instance has a solution with cost less than or equal to 0, so a solution that uses no infinite resources, if and only if each vehicle fully loads three requests before unloading them all, which is possible if and only if a 3-partition over the m vehicles with capacity B exists. To illustrate, this figure is based on the 3-partition instance where $m = 2$, $B = 20$, $A_1 = 10$, $A_2 = 12$, $A_3 = 3$, $A_4 = 6$, $A_5 = 4$, and $A_6 = 5$, for which the partitioning $S_1 = \{1, 4, 5\}$, $S_2 = \{2, 3, 6\}$ gives YES. This partitioning corresponds to a set of routes in which all vehicles fill up exactly all of their capacity (20) before unloading; no infinite resources are required so the total cost is 0. The first route is indicated in red arcs, and the second in orange dashed arcs. For every three items added to the 3-partition problem, three new locations and one new vehicle would appear in this figure

problem I_1 in polynomial time and space in the input size of I_1 : this is simply a matter of creating m vehicles, $3m + 1$ locations, $3m$ requests, and 9 time steps, assigning a polynomial amount of parameters their given values and creating a polynomially sized list Δ .

If I_1 has answer YES, then let S_1, S_2, \dots, S_m be a 3-partition. Construct the following solution of I_2 : for $w = 1, 2, \dots, m$, if $S_w = \{\alpha, \beta, \gamma\}$, let vehicle w go to location α , load all containers of request α , do the same for β and γ , then arrive at location 0 at time 7, and unload all containers so they are in by their deadline 8. As argued earlier, this is feasible in time. By choice of parameters $P_{w,i,t}$ and $N_{i,t}$, this is feasible in the amount of processed containers. Because $\sum_{j \in S_w} B = B = C_w$,

the capacity of the vehicle is at no time exceeded. All in all, this solution is feasible and requires no infinite resources so its total cost is 0. So then I_3 has answer YES.

If I_3 has answer YES, then observe a solution of I_2 with total cost lower than or equal to 0. This solution, then, must use no infinite resources. Each request must be fully serviced by vehicles. Because of the deadlines, a vehicle can service at most three requests fully or partially. There are m vehicles and $3m$ requests, so if no infinite resources are used, each vehicle must service three requests and each request is serviced by exactly one vehicle. So each vehicle fully picks up three requests and then drops all three off at location 0. It must be possible, then, to partition the requests into triples S_w , where vehicle w takes full responsibility for the requests in S_w . Because the vehicles have capacity B , the sum of volumes D_k within in a triple cannot exceed B . The sum of all volumes is $\sum_{k=1}^{3m} D_k = \sum_{j=1}^{3m} A_j = mB$, so the sum of volumes D_k within a triple must be exactly B . Therefore, observing the three jobs serviced by vehicle w gives a set S_w such that S_1, S_2, \dots, S_m is a 3-partitioning of A_1, A_2, \dots, A_{3m} into m partitions of size B , so I_1 has answer YES.

To conclude: I_3 is in NP and can be constructed in polynomial time and space from the decision problem I_1 , I_1 and I_3 are equivalent, and I_1 is known to be strongly NP-complete. Therefore, I_3 is also strongly NP-complete. I_3 is a decision variant of I_2 , so I_2 is strongly NP-hard, and I_2 is an instance of deterministic operational freight planning, so deterministic operational freight planning is strongly NP-hard. \square

Therefore, unless $P = NP$, deterministic operational freight planning has no general solution method that runs in poly-time. Additionally:

Theorem *Unless $P = NP$, no Fully Polynomial-Time Approximation Scheme (FPTAS) exists for deterministic operational freight planning.* \square

Proof This follows directly from the fact that deterministic operational freight planning is a strongly NP-hard optimisation problem [4]. \square

So unless $P = NP$, each general solution method for deterministic operational freight planning runs in an exponential amount of time, and each non-exponential general approximation method is a Polynomial-Time Approximation Scheme (PTAS) at best.

A Note on Labour Conditions

Depending on the timescale of the problem instance, it may be important to also model that barge teams cannot work around the clock for weeks: sleep and refuelling may be necessary. It is likely possible to add rest periods with time windows to this model for Problem 3 by creating “requests” that represent a rest period, which can only be executed by a specific vehicle cheaply, so as to force them to go to the resting place within a specific time window. It is also likely that labour conditions can be modelled in with some extra variables and constraints instead and that this is

a less far-fetched solution. Expanding the model to encompass labour conditions is left as a topic of future research.

Solving to Optimality

The decision space of Problem 3 is obviously much larger and more convoluted than that of Problem 1. However, Problem 3 may still be solved to optimality for small problems in a reasonable amount of time, using the ILP developed in this section.

ILP Formulation

Denote W the set of vehicles. Denote I the set of locations. Denote K the set of requests. Without loss of generality, denote $T = \{0, 1, 2, \dots, end\} \subset \mathbb{N}$ the set of time steps. Using the notation from section “[Notation of Variables and Parameters](#)”, Problem 3 can be formulated as the following integer linear program:

$$\begin{aligned} \min \quad & \sum_{w,i,j,t} E_{w,i,j,t} b_{w,i,j,t} + \sum_{w,i,k,t} F_{w,i,k,t} l_{w,i,k,t} \\ & + \sum_{w,i,k,t} G_{w,i,k,t} u_{w,i,k,t} + \sum_{i,j,k,t} O_{i,j,k,t} q_{i,j,k,t} \\ \text{s.t.} \quad & \text{var} = \text{val} \quad \forall(\text{var}, \text{val}) \end{aligned} \quad (5.1)$$

$$x_{w,k,\hat{t}} = x_{w,k,\hat{t}-1} + \sum_{i \in I} l_{w,i,k,\hat{t}-1} - \sum_{i \in I} u_{w,i,k,\hat{t}-1} \quad (\forall w, k, \hat{t}) \quad (5.2)$$

$$\begin{aligned} y_{i,k,\hat{t}} = y_{i,k,\hat{t}-1} - \sum_{w \in W} l_{w,i,k,\hat{t}-1} + \sum_{w \in W} u_{w,i,k,\hat{t}-1} \\ - \sum_{j \in I} q_{i,j,k,\hat{t}-1} + \sum_{j \in I} v_{j,i,k,\hat{t}} \quad (\forall i, k, \hat{t}) \end{aligned} \quad (5.3)$$

$$z_{w,i,\hat{t}} = z_{w,i,\hat{t}-1} + a_{w,i,\hat{t}} - \sum_{j \in I} b_{w,i,j,\hat{t}-1} \quad (\forall w, i, \hat{t}) \quad (5.4)$$

$$\sum_{i \in I} u_{w,i,k,t} \leq x_{w,k,t} \quad (\forall w, k, t) \quad (5.5)$$

$$\sum_{j \in I} q_{i,j,k,t} + \sum_{w \in W} l_{w,i,k,t} \leq y_{i,k,t} \quad (\forall i, k, t) \quad (5.6)$$

$$v_{i,j,k,t} = \sum_{\tau \in T^*} q_{i,j,k,\tau} \quad (T^* = \{\tau \in T : \tau + S_{i,j,k,\tau} = t\}) \quad (\forall i, j, k, t) \quad (5.7)$$

$$\sum_{k \in K} x_{w,k,t} \leq C_{w,t} \quad (\forall w, t) \quad (5.8)$$

$$\sum_{k \in K} l_{w,i,k,t} + \sum_{k \in K} u_{w,i,k,t} \leq P_{w,i,t} h_{w,i,t} \quad (\forall w, i, t) \quad (5.9)$$

$$\sum_{w \in W} \sum_{k \in K} l_{w,i,k,t} + \sum_{w \in W} \sum_{k \in K} u_{w,i,k,t} \leq N_{i,t} \quad (\forall i, t) \quad (5.10)$$

$$h_{w,i,t} \leq z_{w,i,t} \quad (\forall w, i, t) \quad (5.11)$$

$$a_{w,i,\hat{t}} \leq \sum_{j \in I} \sum_{\tau < \hat{t}} b_{w,j,i,\tau} - \sum_{\tau < \hat{t}} a_{w,i,\tau} \quad (\forall w, i, \hat{t}) \quad (5.12)$$

$$r_{w,\hat{t}} \geq r_{w,\hat{t}-1} - 1 \quad (\forall w, \hat{t}) \quad (5.13)$$

$$r_{w,t} \geq \sum_{i \in I} \sum_{j \in I} T_{w,i,j,t} b_{w,i,j,t} \quad (\forall w, t) \quad (5.14)$$

$$r_{w,t} \leq \text{end} - \text{end} \cdot \sum_{i \in I} a_{w,i,t} \quad (\forall w, t) \quad (5.15)$$

$$\sum_{j \in I} b_{w,i,j,t} \leq z_{w,i,t} \quad (\forall w, i, t) \quad (5.16)$$

$$\sum_{j \in I} b_{w,i,j,t} + h_{w,i,t} \leq 1 \quad (\forall w, i, t) \quad (5.17)$$

$$\sum_{i \in I} \sum_{j \in I} b_{w,i,j,t} \leq 1 \quad (\forall w, t) \quad (5.18)$$

$$b_{w,i,i,t} = 0 \quad (\forall w, i, t) \quad (5.19)$$

$$q_{i,i,k,t} = 0 \quad (\forall i, k, t) \quad (5.20)$$

$$v_{i,i,k,t} = 0 \quad (\forall i, k, t) \quad (5.21)$$

$$a_{w,i,t}, b_{w,i,j,t}, h_{w,i,t}, z_{w,i,t} \in \{0, 1\} \quad \forall w, i, j, k, t \quad (5.22)$$

$$l_{w,i,k,t}, q_{i,j,k,t}, r_{w,t}, u_{w,i,k,t}, v_{i,j,k,t}, x_{w,k,t}, y_{i,k,t} \in \mathbb{N} \quad \forall w, i, j, k, t \quad (5.23)$$

$$i \in I, k \in K, t \in T, w \in W, (var, val) \in \Delta, \hat{t} \in T \setminus \{0\}. \quad (5.24)$$

The cost function minimises the total costs of moving the vehicles around, loading containers, unloading containers, and employing infinite resources. The reader is reminded that unit penalties for delivering after a soft due date can be embedded into the time-dependent cost parameters. Equalities (5.1) state that some variables have fixed values, as detailed in section “[Notation of Variables and Parameters](#)”. Equalities (5.2) state that the amount of containers from request k a vehicle has on board at any time equals the amount it held in the previous time step, plus the amount it has loaded in the previous time step, minus the amount it has unloaded in the previous time step. Equalities (5.3) state that the amount of containers from request k present at a location at any time equals the amount present in the previous time step, minus the amount vehicles took away in the previous time step, plus the amount vehicles have unloaded here in the previous time step, minus the amount sent away by infinite resource in the previous time step, plus the amount that appears here from having been sent by infinite resource. Equalities (5.4) state that whether or not a vehicle is present at a location at any time depends on whether it was present in the previous time step, whether it arrived just before the start of this time step, and whether it has departed to another location during the previous time step. Inequalities (5.5) state that a vehicle cannot unload more containers from request k than it has on board. Inequalities (5.6) state that no more containers from request k can be sent away by infinite resource or loaded onto vehicles than there are present. Equalities (5.7) ensure that if containers are sent away by infinite resource, they arrive at the right point in space-time. Inequalities (5.8) state the total amount of containers on board of a vehicle may never exceed the vehicle capacity. Inequalities (5.9) state that the total amount of containers that a vehicle can load or unload, or process, at a location is limited by a processing speed. Furthermore, processing can only be done if the vehicle has explicitly decided to spend this time step on handling goods at this location. Inequalities (5.10) state that a location can handle only so many containers in one time step. Inequalities (5.11) state that a vehicle can only stay at a location to handle goods if it is present. Inequalities (5.12) state that

a vehicle can only arrive at a location if it has departed towards that location more often than it has arrived there; in other words, a vehicle can only arrive somewhere if it has travelled to that location, not counting previous trips. Inequalities (5.13) enact a “remaining travel time counter”: each time step, it may be decremented by 1, but it always remains greater than or equal to 0. When departing from one location to another, inequalities (5.14) set that counter equal to the travel time. Inequalities (5.15) make it so that a vehicle cannot arrive somewhere at any time step, while it has more than zero remaining travel time on its counter; if some $a_{w,i,t}$ equals 1, then $r_{w,t}$ must be less than or equal to 0, while if all $a_{w,i,t}$ equal 0, $r_{w,t}$ must be less than or equal to the length of the time window, which is a reasonable upper bound on any $r_{w,t}$. Inequalities (5.16) state that a vehicle can only depart from a location if it is present at that location. Inequalities (5.17) state that it is not allowed for a vehicle to both leave from a location and stay at the location to handle goods in the same time step. Inequalities (5.18) state that a vehicle can depart to only one location at a time; if this were not present, it could start “manifesting” at multiple locations. Equalities (5.19), (5.20) and (5.21) disallow that anything “goes” from location i to location i by setting the appropriate variables equal to 0; the same could be achieved by leaving the variables out or enforcing fixed values upon them with Δ . Finally, (5.22) and (5.23) state that some variables are binary decision variables and some variables are non-negative integer decision variables.

Notably absent from this formulation are inequalities that a vehicle can only be at one place at a time and an analogue for containers. However, if the starting situation is properly defined with Δ , the constraints disallow for vehicles and containers to “manifest” at several places at the same time, so the proper behaviours are implicitly present. Whether or not to make these explicit is discussed in section “[Speed-up from Additional Constraints](#)”.

Furthermore, it must be noted that the amount of variables and constraints grows polynomially, but still unfavourably fast, in the amount of vehicles, locations, requests, and time steps. For example, the amount of variables $l_{w,i,k,t}$ is equal to the amount of vehicles times the amount of locations times the amount of requests times the amount of time steps; additionally, each of these variables may take any integer between 0 and D_k as its value. Future research may have to investigate ways to make this model better scalable, possibly by employing reasoning similar to those used in simplifying instances of of Problem 1, as defined in Chap. 3.

Speed-up from Additional Constraints

This section describes a number of constraints that can be added to the ILP from section “[ILP Formulation](#)”. These do not change the integral solution set because they are explicit versions of constraints that are already implicitly true. However, it will be shown in section “[Numerical Results](#)” that including them significantly reduces computation time. Most likely, this is because they constrain the solution

polytope in such a way that the LP relaxations can be solved more easily, but explanations were not further investigated.

The added constraints are as follows:

$$s.t. \sum_{i \in I} z_{w,i,t} \leq 1 \quad (\forall w \in W)(\forall t \in T) \quad (5.25)$$

$$\sum_{i \in I} a_{w,i,t} \leq 1 \quad (\forall w \in W)(\forall t \in T) \quad (5.26)$$

$$\sum_{i \in I} h_{w,i,t} \leq 1 \quad (\forall w \in W)(\forall t \in T). \quad (5.27)$$

Inequalities (16.24) state that a vehicle cannot be at more than one location at a time. Inequalities (16.25) state that a vehicle cannot arrive at more than one location at a time. Inequalities (16.26) state that a vehicle cannot handle goods at more than one location at a time.

Although the results in show that these added constraints significantly reduce computation time, they also show that the achieved computation times are still too long to make this method useful for online use in problems of “real life size”.

Greedy Gain Heuristic

As discussed earlier, heuristics will have to be used to find reasonable solutions for Problem 3 fast enough for online use. A first step is often to develop a simple, greedy algorithm: while not sophisticated, they are often fast and serve as a good starting point for further development.

The idea of the *Greedy Gain heuristic* (GG) is the following. Suppose for now that the infinite resources are trucks and none of the vehicles starts with containers on board. The vehicles are not under way to some location; they are simply waiting somewhere. Requests may already be released and scattered over locations, but they are not on board of vehicles or trucks: they are simply waiting. Each of these batches is considered a separate request. Then, an initial solution can be found by simply trucking everything at its cheapest possible time: this solution is easy to find, but probably very expensive, because infinite resources are typically more expensive than vehicles and the vehicles are not being used at all. In the Greedy Gain heuristic, each request is then assigned to one vehicle, one at a time. To determine which request to assign to which vehicle next, it is checked how much can be gained immediately from each assignment, and then the assignment with the highest gain is chosen, as long as it is positive. When such an assignment is done, the request is “erased” from the system: if 9 containers of this request are on board of a vehicle w , this is no longer described by variables, but rather by w having 9 less capacity at that time in future problems. This way, throughout most of the algorithm, only instances of Problem 3 are solved with just one vehicle and just one request: this counteracts the fast scaling of the ILP computation time. An example of the GG process is shown in Figs. 5.4, 5.5, and 5.6.

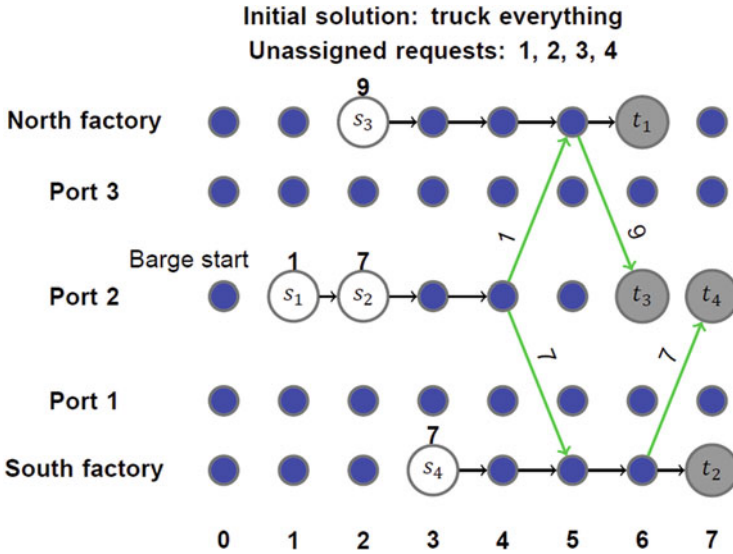


Fig. 5.4 An instance of Problem 3 with one barge and four requests. The barge can visit ports, but not factories. South factory is very close to Port 1, and North factory very close to Port 3; for the rest, distances and prices are all the same. Infinite resources, in the form of trucks, are more expensive than barge usage. The barge starts at Port 2 with nothing on board. The barge has capacity 20: $C_{barge,t} = 20$ for $t = 0, 1, \dots, 7$. In the Greedy Gain heuristic, the initial solution is to just truck everything; in each iteration, one of the requests will be added to a vehicle’s schedule

If, however, in the starting situation a vehicle is not waiting but under way, let it become available to the algorithm only as soon as it arrives at its destination by enforcing values of arrival variables $a_{w,i,t}$ and presence variables $z_{w,i,t}$ with Δ . Given an instance of Problem 3, these values should already be specified in its set Δ . If containers are under way on an infinite resource to some location j , arriving at t , ignore them if j is their due location, and interpret them as a batch that is released at j , time t otherwise. If containers start on board of a vehicle, interpret them as already assigned to that vehicle; in the initial solution, that vehicle handles only those containers, and the resulting schedule is again expressed in parameter changes rather than variables.

With these details addressed, the Greedy Gain heuristic is formally presented in Algorithm 5.1.

The most potent upside of the GG heuristic is that, throughout most of the algorithm, only sub-problems are solved with just one vehicle and one request; this counteracts the fast computational growth in the amount of vehicles and requests. However, it does nothing to reduce the amount of locations and time steps in a sub-problem; in section “[Numerical Results](#)”, it will become clear that these still greatly contribute to the computation time. Reducing the amount of locations and time steps in a sub-problem, or solving a sub-problem without the use of the ILP, may be worthwhile venues for future research.

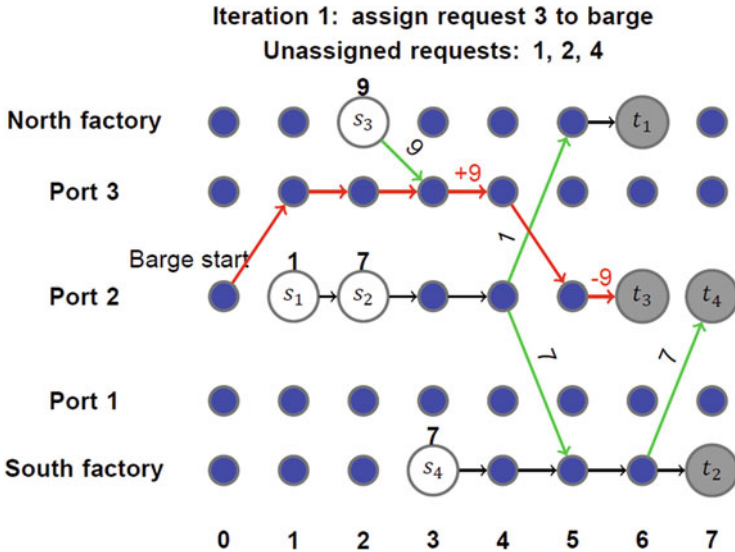


Fig. 5.5 In the first iteration, request 3 is assigned to the barge because this assignment has the highest immediate gain: namely, 9 containers are now largely transported by barge instead of truck. In future iterations, request 3 no longer computationally “exists”; instead, the barge has a mysterious appointment at Port 3, time 3, in which its capacity changes from $C_{barge,3} = 20$ to $C_{barge,4} = 11$, and a mysterious appointment at Port 2, time 5, in which its capacity changes from $C_{barge,5} = 11$ to $C_{barge,6} = 20$. This way, the sub-problem of adding request 1 or another request to the current schedule of the barge involves only variables of the barge and the new request, not of the old request; this makes it computationally more manageable

An obvious downside of the GG heuristic, being a greedy algorithm, is that it may make “bad choices” in early stages that influence the decision space of later iterations. This happens in Fig. 5.5, where greedily picking the request of size 9 makes it so that in the next iteration, the algorithm can only pick a “close-by” request of size 1; a better solution would have been attained if the two “close-by” requests of size 7 were picked. It would be smarter; thus, if an algorithm could somehow recognise the “compatible” requests of size 7 as one cluster and the other two as another cluster, then assign the vehicle to the cluster with the best value. This idea is the basis of the next heuristic discussed.

Compatibility Clustering Heuristic

The idea of the *Compatibility Clustering heuristic* (CC heuristic) is as follows: if there are n requests and m vehicles, divide the requests into m clusters of requests that are “compatible”, that is to say, likely to be handled together by one vehicle efficiently. If two requests have releases at the same location at virtually the same

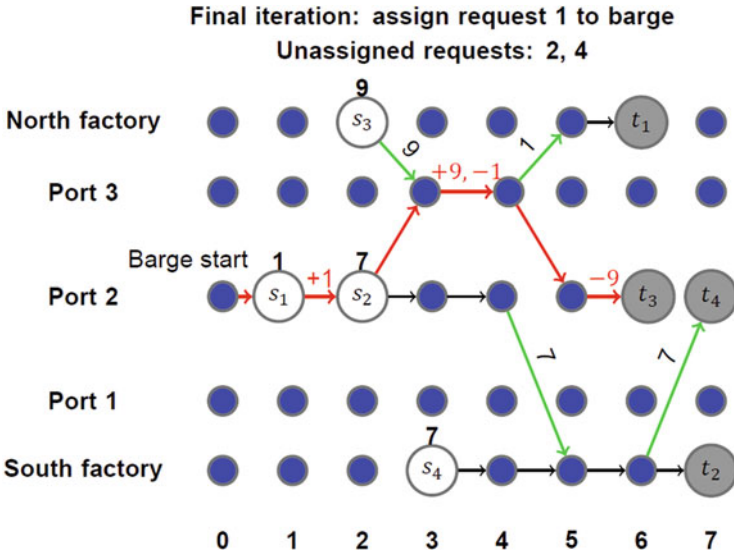


Fig. 5.6 In the second iteration, the algorithm would like to assign one of the requests of 7 containers to the barge, but it cannot: the barge already has appointments at time 3 and time 5, so it cannot wait for the release of request 2 or 4. The containers of request 2 could theoretically be trucked to Port 3 so they can be loaded during the existing appointment and unloaded during the second appointment, but this would be more expensive than just trucking them straight to South factory, so this assignment would have negative gain. The only assignment left with positive gain is to add request 1 to the schedule of the barge; it consists of only one container, but the barge is already taking a favourable route for it anyway. After this assignment, the next iteration has only negative gains, so the algorithm stops. The final solution is sub-optimal: the barge handles requests 1 and 3 for a total of 10 containers, while it would have been smarter to handle requests 2 and 4 for a total of 14 containers

time, and the same goes for their due points in space-time, they are extremely compatible: it is very likely efficient that one vehicle handles both these requests. If the due node in space-time of the first request almost coincides with the release node of the second request, so dropping off the first request smoothly leads into picking up the second request, these requests are also quite compatible, but not as compatible as in the previous situation: in the first situation, the requests share two places in space-time where the vehicle is needed, where in the latter, they share only one. See also Fig. 5.7.

In order to actually use the existing clustering algorithms to cluster requests on their “compatibility”, the compatibility of requests must be properly expressed as a metric. Before this is done, however, some remarks and an outline of the CC heuristic are given, to show that some other metrics are necessary as well.

Compatibility must depend on both space and time: if two requests have the same origin–destination pair, but one has its release today and the other next week, their shared origin–destination pair loses all value. Understanding this, suppose the time window is $2m$ days. If the requests are blindly divided into m clusters, it may well

Algorithm 5.1 The Greedy Gain heuristic: completely assign requests to vehicles, one at a time, picking the one with highest immediate gain. This gain is computed by observing sub-problems of one vehicle and one request, where the other requests handled by this vehicle are embedded as parameters: capacities are decreased and handling appointments enforced

Require: Instance of Problem 3 (set of vehicles W , set of locations I , set of requests K , list of time steps; set Δ that forces fixed values to describe starting distribution and system properties; parameters as described in section “[Notation of Variables and Parameters](#)”)

Ensure: Solution of Problem 3, possibly non-optimal

- 1: **for** k in set of requests **do**
 - 2: Interpret each separate batch of containers belonging to k as a ‘separate request’ κ
 - 3: **for** w in set of vehicles **do**
 - 4: Initialise $AR(w)$, the set of ‘separate requests’ assigned to w , as only those already on board in the starting situation
 - 5: **if** $AR(w) = \emptyset$ **then**
 - 6: Initialise $H(w) = \emptyset$, the set of fixed values of h that enforce the current schedule of w
 - 7: Initialise $f(w)$, the cost of the current schedule of w , as 0
 - 8: **else**
 - 9: Solve, using the ILP, the sub-problem where only w handles only the requests in $AR(w)$, denote the solution X
 - 10: Initialise $H(w) = \{(h_{w,i,t}, 1) | h_{w,i,t} = 1 \text{ in } X\}$, initialise $f(w)$ the cost of X
 - 11: Update $C_{w,t} := C_{w,t} - \sum_{\kappa \in AR(w)} x_{w,\kappa,t}$
 - 12: Update $P_{w,i,t} := P_{w,i,t} - \sum_{\kappa \in AR(w)} (l_{w,i,\kappa,t} + u_{w,i,\kappa,t})$
 - 13: Update $N_{i,t} := N_{i,t} - \sum_{\kappa \in AR(w)} (l_{w,i,\kappa,t} + u_{w,i,\kappa,t})$
 - 14: Init the unassigned requests: $UR = (\text{all separate requests}) / \{AR(w) | w \text{ in set of vehicles}\}$
 - 15: **for** unassigned request κ in UR **do**
 - 16: Compute infinite resource cost $IRC(\kappa)$ the cheapest cost of sending all of κ from its release location to its due location by infinite resource
 - 17: **for** vehicle w in set of vehicles **do**
 - 18: Create, solve and store sub-problem where only w handles only κ , respecting the current $H(w)$, $C_{w,t}$, $P_{w,i,t}$ and $N_{i,t}$
 - 19: Denote $f(w + \kappa)$ the cost of this sub-problem
 - 20: Compute $gain(w, \kappa) = (f(w) + IRC(\kappa)) - f(w + \kappa)$
 - 21: **while** $UR \neq \emptyset$ and $\max_{\kappa \in UR} gain(w, \kappa) > 0$ **do**
 - 22: Determine $(w, \kappa) = \arg \max gain(w, \kappa)$ with corresponding solution X
 - 23: Update $AR(w) := AR(w) \cup \{\kappa\}$, $UR := UR \setminus \{\kappa\}$
 - 24: Update $H(w) := H(w) \cup \{(h_{w,i,t}, 1) | h_{w,i,t} = 1 \text{ in } X\}$, $f(w) := f(w + \kappa)$
 - 25: Update $C_{w,t} := C_{w,t} - \sum_{\kappa \in AR(w)} x_{w,\kappa,t}$
 - 26: Update $P_{w,i,t} := P_{w,i,t} - \sum_{\kappa \in AR(w)} (l_{w,i,\kappa,t} + u_{w,i,\kappa,t})$
 - 27: Update $N_{i,t} := N_{i,t} - \sum_{\kappa \in AR(w)} (l_{w,i,\kappa,t} + u_{w,i,\kappa,t})$
 - 28: **for** unassigned request κ in UR **do**
 - 29: Create, solve and overwrite new sub-problem where only w handles only κ , respecting the current $H(w)$, $C_{w,t}$, $P_{w,i,t}$ and $N_{i,t}$
 - 30: Denote $f(w + \kappa)$ the cost of this sub-problem
 - 31: Compute $gain(w, \kappa) = (f(w) + IRC(\kappa)) - f(w + \kappa)$
 - 32: Compute and return solution where all κ in UR are trucked as in the initial solution and the solved (w, κ) -sub-problems enforce values on h, l, q, u, v, x, y through Δ .
-

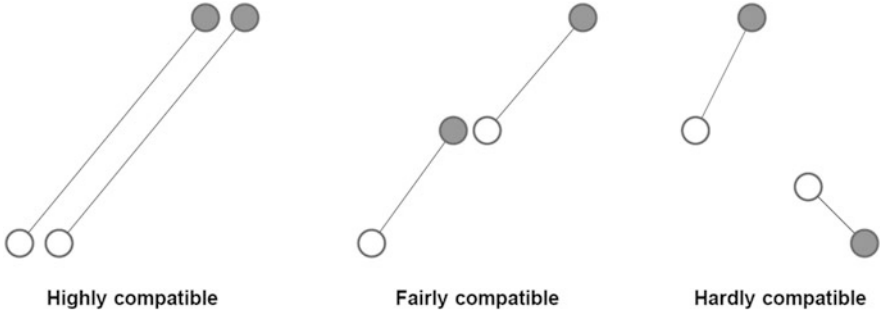


Fig. 5.7 Pairs of requests with a different “compatibility”, that is to say, likelihood that it is efficient to assign them both to the same vehicle. The white nodes signify release points in space-time, and the grey nodes signify where they are due. The left pair consists of two requests that have to be picked up around the same place in space-time and dropped of around the same place as well: they share two places in space-time where a vehicle would have to go to. The middle pair shares only one point in space-time and the right pair shares none

happen that the first cluster consists of all requests of the first two days, the second cluster comprises the second two days, etc. Assigning vehicles to clusters, then, means telling one vehicle to try and take care of everything that happens in two days and not contributing anything in all other days: this is not the desired planning behaviour. So the CC heuristic consists of first splitting up the instance into a given amount of periods, based on time, and then within each period finding m clusters, based on space-time. Vehicles should then be assigned to a sequence of clusters, one cluster for each period. To determine these sequences smartly, one would have to know how “connectible” a cluster of one period is with a cluster of the next period, disregarding time: if there are three remote major areas of activity and three vehicles, one would try to sequence the clusters so the vehicles can stay within one area as much as possible. Thus, the CC heuristic needs a variety of different metrics, which are developed in section “[Used Metrics](#)”.

Used Metrics

This section introduces the metrics necessary for clustering. The reader is reminded that any distance function $d : X \times X \rightarrow [0, \infty)$ is a metric and the pair (X, d) a metric space if and only if they satisfy the following properties:

1. $d(x, y) \geq 0$ for all $x, y \in X$.
2. $d(x, x) = 0$ for all $x \in X$.
3. $d(x, y) = 0 \Rightarrow x = y$ for all $x, y \in X$.
4. $d(x, y) = d(y, x)$ for all $x, y \in X$.
5. $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$.

Famous metrics include Euclidean distance and the taxicab metric. Furthermore, Fujita formulated the following crucial result: if (X, d) is a metric space, and A and B are finite subsets of X , then the “average distance” between points in A and points in B is a function that again satisfies the properties of a metric [2].

First, if $s_1 = (v_1, t_1)$ and $s_2 = (v_2, t_2)$ are two space–time nodes in an instance of Problem 3, the *vehicle-based space–time distance between space–time nodes*, $\sigma_w((v_1, t_1), (v_2, t_2))$ is defined as

$$\sigma_w(s_1, s_2) = \max\{T_{w,v_1,v_2,t_1}, |t_2 - t_1|\}.$$

It can be interpreted as follows: if vehicle w wants to perform an infinitely short action at one place in space–time and then another action at another location in space as soon as possible, how much time steps will be there between these actions? If these actions are picking up containers but their release times are far apart, σ_w equals that difference in time, $|t_2 - t_1|$. If instead the actions occur very close together in time, the difference is the time it takes to get from the first location to the second, T_{w,v_1,v_2,t_1} .

Theorem 5.1 *Under the assumptions that for all locations i and j , $T_{w,i,j,t}$ is time-independent, is symmetric in i, j , equals 0 if $i = j$ and is positive otherwise, and satisfies the triangle inequality, $\sigma_w(s_1, s_2)$ is a metric on the space–time nodes of the instance. \square*

Proof

1. By assumption, $T_{w,i,j,t} \geq 0$ for all locations i, j . Also, $|x - y| \geq 0$ for all $x, y \in \mathbb{R}$. So $\sigma_w(s_1, s_2) \geq 0$ for all space–time nodes s_1, s_2 .
2. $\sigma_w(s_1, s_1) = \max\{T_{w,v_1,v_1,t_1}, |t_1 - t_1|\} = \max\{0, 0\} = 0$.
3. Suppose $\sigma_w(s_1, s_2) = 0$ for some space–time nodes $s_1 = (v_1, t_1)$, $s_2 = (v_2, t_2)$.
So $T_{w,v_1,v_1,t_1}, |t_2 - t_1| \leq 0$. $|t_2 - t_1| \leq 0 \Rightarrow t_1 = t_2$, because the Euclidean distance is a metric. By assumption, $T_{w,v_1,v_2,t} \leq 0$ implies that $v_1 = v_2$. So $s_1 = (v_1, t_1) = (v_2, t_2) = s_2$.
4. Because $T_{w,i,j,t}$ is assumed to be symmetric in i, j and time-independent and $|t_2 - t_1| = |t_1 - t_2|$, it follows that $\sigma_w(s_1, s_2) = \max\{T_{w,v_1,v_2,t_1}, |t_2 - t_1|\} = \max\{T_{w,v_2,v_1,t_2}, |t_1 - t_2|\} = \sigma_w(s_2, s_1)$.
5. Let s_1, s_2 , and s_3 be space–time nodes of the instance. Suppose $\sigma_w(s_1, s_3) = T_{w,v_1,v_3,t_1}$. By the assumptions of triangle inequality and time independence, $T_{w,v_1,v_3,t_1} \leq T_{w,v_1,v_2,t_1} + T_{w,v_2,v_3,t_2}$, so

$$\sigma_w(s_1, s_3) = T_{w,v_1,v_3,t_1} \leq T_{w,v_1,v_2,t_1} + T_{w,v_2,v_3,t_2} \leq \sigma_w(s_1, s_2) + \sigma_w(s_2, s_3).$$

If instead $\sigma_w(s_1, s_3) = |t_3 - t_1|$,

$$\sigma_w(s_1, s_3) = |t_3 - t_1| = |(t_3 - t_2) + (t_2 - t_1)| \leq |t_3 - t_2| + |t_2 - t_1|$$

$$\leq \sigma_w(s_1, s_2) + \sigma_w(s_2, s_3).$$

\square

A request is largely described by two space–time nodes: its release node in space–time and its due node in space–time. Theoretically, it is not at all necessary that a request be picked up near its release time and delivered near its deadline, nor at the release or due locations, but it may often be the case in efficient synchromodal practice. If a request J_1 is seen as a pair of two nodes $\{s_1, s_2\}$, and another request as $J_2 = \{s_3, s_4\}$, then if every pair of requests with the same release node and due node is considered as one large request, the result of Fujita can be used to define the *vehicle-based space–time distance between requests*, $d_w(J_1, J_2)$:

$$d_w(J_1, J_2) = \frac{1}{2|\{s_1, s_2, s_3, s_4\}|} \left(\sum_{s_i \in \{s_1, s_2\}} \sum_{s_j \in S_2 \setminus S_1} \sigma_w(s_i, s_j) + \sum_{s_j \in \{s_3, s_4\}} \sum_{s_i \in S_1 \setminus S_2} \sigma_w(s_i, s_j) \right).$$

Theorem *If σ_w is a metric on space–time nodes, d_w is a metric on requests.* \square

Proof This follows immediately from Fujita. The concerned reader may note that if $J_i = \{\text{release node } i, \text{due node } i\}$ and $J_1 = J_2$, then it might be that *release node 1 = due node 2* and *due node 1 = release node 2*; however, assuming that due nodes are always strictly further in time than release nodes, $J_1 = J_2$ implies that the requests have equal release nodes and equal due nodes. \square

It was explained earlier in section “[Compatibility Clustering Heuristic](#)” that requests will first be divided over a specified amount of time periods. This can also be done with clustering, using a metric that depends only on time, namely the *temporal distance between requests*:

$$\tau(J_1, J_2) = \frac{1}{2|\{s_1, s_2, s_3, s_4\}|} \left(\sum_{s_i \in \{s_1, s_2\}} \sum_{s_j \in S_2 \setminus S_1} |t_j - t_i| + \sum_{s_j \in \{s_3, s_4\}} \sum_{s_i \in S_1 \setminus S_2} |t_j - t_i| \right).$$

From Fujita and the fact that $|t_j - t_i|$ is an Euclidean distance, it immediately follows that τ is a metric on the requests.

Finally, the need was argued to express connectability of request clusters from different time periods, based mainly on spatial distance. This is done by first describing the *vehicle-based spatial distance between space–time nodes* s_1, s_2 as $\pi_w(s_1, s_2) = T_{w, v_1, v_2, t_1}$, then the *vehicle-based spatial distance between requests*, denoted $\pi'_w(J_1, J_2)$, as the average spatial distance between the sets $\{s_1, s_2\}$ and $\{s_3, s_4\}$, then describing the *vehicle-based spatial distance between clusters* $C_1 = \{J_1, \dots, J_2\}$ and $C_2 = \{J_3, \dots, J_4\}$, denoted $\pi''_w(C_1, C_2)$, as the average spatial distance between the sets $\{J_1, \dots, J_2\}$ and $\{J_3, \dots, J_4\}$. So

$$\pi''_w(C_1, C_2) = \frac{1}{|C_1 \cup C_2| |C_1|} \sum_{J_i \in C_1} \sum_{J_j \in C_2 \setminus C_1} \pi'_w(J_i, J_j) + \frac{1}{|C_1 \cup C_2| |C_2|} \sum_{J_j \in C_2} \sum_{J_i \in C_1 \setminus C_2} \pi'_w(J_i, J_j).$$

If the same assumptions on $T_{w,i,j,t}$ apply as in Theorem 5.1, π'' is a metric on request clusters, though this is not actually necessary for the functioning of the algorithm described in section “[Description of Algorithm](#)”.

With these metrics, it becomes possible to cluster requests before assigning them. Three downsides were discovered, however, in having to use metrics. For one, the functions are currently only guaranteed metrics if the travel times $T_{w,i,j,t}$ “behave metrically” and are time-independent: though this is often quite a natural assumption to make, it is still a loss of generality. Second, if two requests are released and due at exactly the same times but slightly different locations, they may in practice be exactly *not* compatible because the vehicle can only be present at one location for loading and miss the loading window for the other; if, however, compatibility were modelled to decrease when requests come too close together, but be perfect when the requests exactly overlap, this would lead to an inevitable loss of the triangle inequality. Fortunately, this “non-decreasingness” of compatibility is often not a problem, assuming that infinite resources can be used to get both batches in one place cheaply, because they are close together. Finally, basing distances on average distances may even out the distinctiveness of requests too much, making all requests “about as compatible”; though intuitively one may want to use a minimum distance rather than an average distance, this leads to a loss of property 3 of metric functions as soon as requests may share a node. If any of these behaviours are deemed too undesirable, it may be interesting to study whether the algorithm in section “[Description of Algorithm](#)”, or an appropriate adjustment of it, still works when the undesirable behaviours are modelled out at the possible cost of non-metricity.

Description of Algorithm

Finally, the CC heuristic is described as Algorithm 5.2. Figure 5.8 illustrates the desired result of the algorithm and may serve as a visual aid when reading it.

Note that for its clustering, it uses the UPGMA method [5]; because among clustering methods with non-Euclidean distances, it is known to produce clusters relatively equal in size [7], which is beneficial in dividing workload fairly over vehicles and thus preventing having to resort to infinite resources unnecessarily. Note furthermore that, though the metrics developed in section “[Used Metrics](#)” are defined vehicle-based, it is not yet known at some points in the algorithm which vehicle will service certain requests or clusters; at these points, the distance is averaged out over vehicles. In some real-life instances, one may expect travel times to be similar for different vehicles, and if not, this is partially compensated in the final stages of the algorithm, where specific vehicles are taken into account. Further discussion on properties of this algorithm is postponed to section “[Discussion](#)”.

Algorithm 5.2 The Compatibility Clustering heuristic: in each time period, cluster the requests into $|W|$ clusters according to their $d(J_1, J_2)$ distance, then make $|W|$ strings of clusters over the time periods by using the Hungarian algorithm between each time step and the next, then solve each sub-problem where only vehicle w handles only the requests in a given cluster sequence, and then use the results to assign vehicles to cluster sequences using the Hungarian algorithm

Require: Instance of Problem 3 (set of vehicles W , set of locations I , set of requests K , list of time steps, amount of time periods Ξ ; set Δ that forces fixed values to describe starting distribution and system properties; parameters as described in section “[Notation of Variables and Parameters](#)”)

Ensure: Solution of Problem 3, possibly non-optimal

- 1: **for** request k in K **do**
 - 2: Interpret each separate batch of containers belonging to k as a ‘separate request’ κ
 - 3: Cluster all requests into Ξ time periods, using UPGMA with the $\tau(J_i, J_j)$ temporal distance between requests and place the clusters into a chronological list
 - 4: **for** cluster in list of time period clusters **do**
 - 5: **if** amount of requests in this time period $\leq |W|$ **then**
 - 6: Make every request its own cluster
 - 7: **else**
 - 8: Cluster the requests within this time period into $|W|$ clusters, using UPGMA with the average (over vehicles) $d_w(J_i, J_j)$ space-time distance between requests
 - 9: Initialise $|W|$ empty cluster sequences
 - 10: Add each cluster in the first time period to its own cluster sequence
 - 11: **for** time period in $\{2, \dots, \Xi\}$ **do**
 - 12: Compute each spatial cluster distance $\pi''(C_1, C_2)$ between the clusters at the end of the current cluster sequences and the clusters in this time period
 - 13: **if** amount of non-empty cluster sequences \geq amount of clusters in this time period **then**
 - 14: Assign each cluster in this time period to a cluster at the end of a non-empty cluster sequence, using the Hungarian algorithm
 - 15: Set each cluster in this time period at the end of the sequence to which it is assigned
 - 16: **else**
 - 17: Assign the last cluster in each non-empty cluster sequence to a cluster in this time period, using the Hungarian algorithm
 - 18: Set each unassigned cluster in this time period at the end of a currently empty cluster sequence
 - 19: Set each assigned cluster in this time period at the end of the sequence to which it is assigned
 - 20: **for** vehicle w in W **do**
 - 21: **for** cluster sequence in set of cluster sequences **do**
 - 22: Compute the cost of assigning w to this cluster sequence by solving the sub-problem where only w handles only the requests in this cluster sequence, using the ILP and storing the solution
 - 23: Use the Hungarian algorithm to assign vehicles to cluster sequences
 - 24: Return the solution where each vehicle w and each request k does exactly what it does in the solved sub-problem if w is assigned to the cluster sequence that contains k .
-

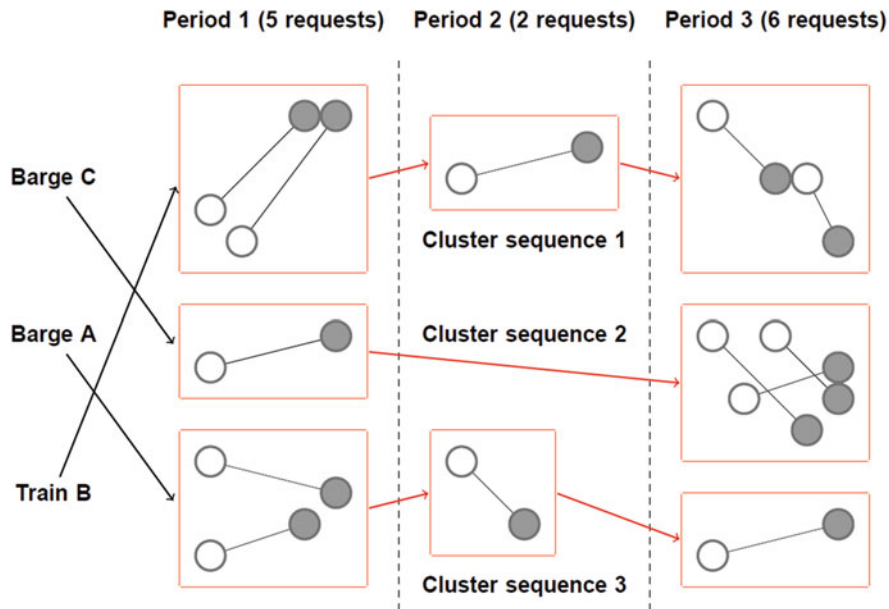


Fig. 5.8 The desired result of the Compatibility Clustering heuristic: first, the requests are divided into a given number of time periods. Then, within each time period, the requests are divided into m clusters based on compatibility, with m amount of vehicles; if the amount of requests in a time period is less than or equal to m , each request forms its own cluster, as here in Period 2. Each period then has at most m clusters: these are stringed together into cluster sequences, by looking at the spatial cluster distance between each cluster in one period and each cluster in the next and then finding a minimum weight perfect matching using the Hungarian algorithm. Finally, this yields m cluster sequences, and the m vehicles are assigned to the m cluster sequences, again with a minimum weight perfect matching: the assignment costs are determined by solving a sub-problem where the given vehicle executes the requests in the given cluster sequence

Numerical Results

In this section, the various methods discussed in this chapter are tested for objective value efficiency and computation time. “ILP” represents solving the instance to optimality with the ILP as given in section “[ILP Formulation](#)”, while “ILP+” represents solving it with the additional constraints from section “[Speed-up from Additional Constraints](#)”. Though the hardware setup is still the same as in section “[Numerical Results](#)”, different class definitions are used: Class A consists of random problem 3 instances with 2 vehicles, 5 locations, 4 requests, and 14 time steps; Class B has 3 vehicles, 6 locations, 5 requests, and 14 time steps; Class C has 4 vehicles, 8 locations, 8 requests, and 14 time steps. Unfortunately, no results are given for a class that represents “real life size”: this would involve 121 time steps rather than 14, and even the heuristics were unable to find a single solution in under

Table 5.1 Numerical results for random instances of Problem 3. Class A concerns instances with 2 vehicles, 5 locations, 4 requests, and 14 time steps; the other classes can be interpreted the same way from their descriptions at the top of the table

		Class A (2, 5, 4, 14)	Class B (3, 6, 5, 14)	Class C (4, 8, 8, 14)
ILP cost over optimal	Mean	0%	–	–
	Variance	0% ²	–	–
	Worst	0%	–	–
ILP computation time	Mean	67.9 s	–	–
	Variance	9549.6 s ²	–	–
	Worst	288.9 s	–	–
ILP+ cost over optimal	Mean	0%	0%	–
	Variance	0% ²	0% ²	–
	Worst	0%	0%	–
ILP+ computation time	Mean	57.7 s	26.2 m	–
	Variance	4197.8 s ²	2149.1 m ²	–
	Worst	162.8 s	2.56 h	–
GG cost over optimal	Mean	210.5%	152.5%	Unknown
	Variance	72973% ²	40821% ²	Unknown
	Worst	887.5%	508.3%	Unknown
GG computation time	Mean	15.4 s	31.5 s	113.5 s
	Variance	9.0 s ²	14.8 s ²	191.9 s ²
	Worst	22.0 s	40.0 s	140.2 s
CC cost over optimal	Mean	243.8%	249.4%	Unknown
	Variance	150233% ²	74553% ²	Unknown
	Worst	1200.0%	720.0%	Unknown
CC computation time	Mean	15.1 s	28.8 s	116.9 s
	Variance	14.0 s ²	43.3 s ²	349.0 s ²
	Worst	22.4 s	40.7 s	145.1 s

five hours, which is sufficient to prove that these methods cannot yet be applied in online practice.

As problem classes become computationally more intense, certain solution methods are left out. In the first two classes, the exact optima are known, but not in Class C; here, it can only be stated that the Compatibility Clustering heuristic achieves results that have a cost of, on average, 129.0% over the cost attained with the Greedy Gain heuristic. Among these instances, none were found where CC found a cheaper solution than GG did.

The results can be viewed in Table 5.1 and are discussed in section “[Discussion](#)”.

Discussion

From the results in section “[Numerical Results](#)”, several things can be concluded:

1. Using the additional constraints from section “[Speed-up from Additional Constraints](#)” leads to a significant decrease in ILP computation time.
2. Going from Class A to Class B, which is “only slightly larger”, increases the average ILP+ computation time by a factor 26.
3. Class B has an average ILP+ computation time of 26.2 min, with worst observed case 2.56 h, making the ILP unsuitable for decision support in cases of size B or up under the current hardware setup.
4. On average, the GG heuristic achieves costs that are a factor 2.5 to 3.1 of the optimal cost, though this gap appears to become smaller as problem instances grow.
5. The computation time of both heuristics is much more stable and predictable than that of solving the ILP, and the growth of computation time is more manageable.
6. The computation times of the two heuristics are surprisingly similar, but the GG heuristic attains better results in cost.
7. Class C consists of instances that have an average computation time of two minutes for both heuristics. Depending on goals and preferences, this is “pushing the limit” of how much computation time is allowed in decision support.
8. The growth of computation time observed in all methods makes it clear that instances of the last class in previous chapters, consisting of 6 vehicles, 32 locations, 40 requests, and 121 time steps, cannot yet be solved efficiently with these methods.

Conclusion 1 is probably due to the LP-relaxation solution polytopes being smaller, leading to better bounds in the ILP process. Conclusion 4, the non-optimality of the GG heuristic, is explained in section “[Greedy Gain Heuristic](#)”. Conclusion 6, or rather than the GG heuristic performs better than the CC heuristic, is unfortunate, as the CC heuristic was designed to deal with the non-optimality of the GG heuristic. It should be investigated if other clustering methods and other distance metrics yield better results, perhaps even methods that discard metricity as suggested in section “[Used Metrics](#)”, and studying optimal solutions may lead to other ideas of how to cluster, or other ideas for heuristics in general. Additionally, both heuristics assign a full request to at most one vehicle, which limits options of intermodality and eliminates options of consolidation in order to find a solution more quickly.

All other conclusions are about how the computation time grows too quickly for any of these methods to be useful in operational decision support for problems of “real life size”. This is easy to explain: the variables $b_{w,i,j,t}$, $q_{i,j,k,t}$, and $v_{i,j,k,t}$ and their corresponding inequalities grow enormously in amount, though these are only the worst offenders. Not only does this mean that the ILP solver has to handle thousands of variables and constraints, but also that the decision space is very large: the more time steps there are, the more random sequences of a vehicle visiting locations there are to investigate. More broadly, the solution set contains

“nonsensical” solutions such as a vehicle picking up one container, going to some random location, dropping the container, picking it up again, and repeating this until trucking it at the end, but the non-optimality of such strategies should be picked up very quickly in the branch-and-bound process; however, whether or not it is non-optimal to visit certain locations under certain conditions may require a deeper search tree. The heuristics solve sub-problems with only one vehicle w and a limited amount of requests, but even then, the amount of variables $b_{w,i,j,t}$ is still the amount of locations squared times the amount of time steps and the same or worse goes for $q_{i,j,k,t}$ and $v_{i,j,k,t}$.

Though speed-ups can probably be attained using commercial ILP solvers, parallelisation, cloud computing, and streamlining of the implementations, this may cost a considerable amount of money before the current methods are useful in operational practice. Instead, there are still several mathematical innovations and refinements that are likely to reduce computation time:

- In the field of heuristics, develop heuristics that solve ILP sub-problems more rarely and carefully or not at all. This may not be a trivial task: the great benefit of solving a sub-problem using the ILP is not so much the minimal cost, but that the ILP solver looks at the ILP and works out all details to make the solution feasible. When not using an ILP solver, much attention may have to be spent on ensuring solution feasibility.
- When working with the GG heuristic or other iterative heuristics, try an ILP solver that allows passing initial solutions: when checking the gain of adding request k to the schedule of vehicle w , it may be very useful to pass the current schedule of w and the current trucking decision for k as an initial solution.
- When solving sub-problems of one vehicle and one request, investigate ways to limit the amount of relevant locations and time steps as well, as these still make the sub-problems too large for “real life size” instances.
- When solving the ILP, look into column generation. In a given solution, the amount of variables that have a value greater than 0 is comparatively small because one only needs a handful of positive variables to describe the route of a vehicle or the used routes of a request. Applying column generation, however, is again not trivial because the complex system dynamics may make it so that a variable cannot be added one at a time.
- Find lesser-indexed versions of variables. For example, it appears to be possible to replace variables $b_{w,i,j,t}$ (barge w departs from location i to location j at time t) with variables $b_{w,j,t}$ by splitting each equality

$$z_{w,i,t} = z_{w,i,t-1} + a_{w,i,t} - \sum_{j \in I} b_{w,i,j,t-1}$$

into four inequalities

$$a_{w,i,t} \leq z_{w,i,t} \leq a_{w,i,t} + 2z_{w,i,t-1},$$

$$a_{w,i,t} + 2z_{w,i,t-1} - \sum_{j \in I} b_{w,j,t} - 1 \leq z_{w,i,t} \leq a_{w,i,t} - \sum_{j \in I} b_{w,j,t} + 1$$

and splitting each inequality $r_{w,t} \geq \sum_{i \in I} \sum_{j \in I} T_{w,i,j,t} b_{w,i,j,t}$ into $|I|^2$ inequalities

$$r_{w,t} \geq T_{w,i,j,t} (z_{w,i,t} + b_{w,j,t} - 1)$$

and treating the contribution to the cost function the same, but with $E_{w,i,j,t}$.

- Similar to how Algorithm 1 eliminates redundant information in Problem 1, investigate how redundant information can be eliminated in Problem 3.
- When solving the ILP, see if it is beneficial to add more inequalities like the ones in section “[Speed-up from Additional Constraints](#)”.

Each of these is recommended for future research.

Added Value

In this chapter, a problem was formulated that is a departure from many classical VRP problems, despite it having direct applications in practice. Its strong NP-hardness was proven, an ILP was formulated to solve it, and two heuristics were proposed. Though the high computation times of these methods make them only applicable to small instances without resorting to more costly hardware configurations, clear venues were set out for how to find solutions more quickly through mathematical innovation.

Conclusion

This chapter sought to answer the following research question:

How can a low-cost net-centric operational transport schedule be found fast enough for online use if everything is known beforehand?

Deterministic operational freight planning, or “Problem 3”, was defined to be the problem of making the timetables for vehicles for travelling to locations and handling goods at terminals and simultaneously assigning containers to vehicles and infinite resources, all in such a way that the total cost of getting the containers to their destinations is minimal.

Though one would expect Problem 3 to resemble VRP problems, it is actually quite a departure from them, in that goods may switch vehicles any amount of times and vehicles do not have to start or end at a depot. Problem 3 was proven to still be strongly NP-hard, thus to not have an FPTAS.

An ILP formulation of this problem was developed, complete with the options to specify what the vehicles are doing at time 0 and to disallow certain vehicles to arrive at certain locations at certain times, so as to model opening times, terminal time slot availabilities and whether a certain terminal can even handle a certain modality. It was made slightly faster by adding additional inequalities.

Because the amount of variables in this ILP is polynomial but high in the amount of vehicles, locations, requests, and time steps, a Greedy Gain heuristic was developed that solves sub-problems of only one vehicle and one request to see how much gain there is to be had in adding this request to the schedule of this vehicle and then iteratively adding requests to vehicle schedules based on the highest immediate gain. To counteract the obvious non-optimality this strategy may attain, a Compatibility Clustering heuristic was developed, which first clusters requests on how efficient it is to handle them together, then checks for each vehicle how much it would cost to handle this cluster by solving the corresponding sub-problem, and then uses the Hungarian algorithm to assign vehicles to clusters.

Though the computation time of the CC heuristic is surprisingly similar to that of the GG heuristic and both are much faster than solving the complete ILP, the CC heuristic finds worse solutions than the GG heuristic, and none of the methods can find a solution fast enough for operational decision support in instances of “real life size” on modest hardware because even the sub-problems are too large for those instances. Both heuristics find solutions with significantly non-optimal costs, possibly because they sacrifice some intermodality and all consolidation for speed, but the gap appears to become smaller for the GG heuristic as instances grow. A number of recommendations were given to find a solution more quickly: most importantly, a heuristic may have to be developed that does not solve sub-problems with the ILP at all.

To answer the sub-question: small instances may be solved to optimality fast enough, and slightly larger instances may be solved with the Greedy Gain heuristic. For instances of “real life size”, however, more research will have to be done to find solutions fast enough for operational decision support, and a number of venues for this were formulated.

References

1. Cordeau, J. F., & Laporte, G. (2003). The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, 1(2), 89–101.
2. Fujita, O. (2013). Metrics based on average distance between sets. *Japan Journal of Industrial and Applied Mathematics*, 30(1), 1–19.
3. Garey, M. R., & Johnson, D. S. (1978). “Strong” NP-completeness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3), 499–508.

4. Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness* (pp. 90–91). WH Free.
5. Michener, C. D., & Sokal, R. R. (1957). A quantitative approach to a problem in classification. *Evolution*, 11(2), 130–162.
6. Pedersen, M. B., Madsen, O. B., & Nielsen, O. A. (2005). Optimization models and solution methods for intermodal transportation. Ph.D. Thesis, Technical University of Denmark, Department of Transport, Traffic Modelling.
7. Python: Clustering methods in Python. Retrieved July 29, 2017 from <http://scikit-learn.org/stable/modules/clustering.html>
8. Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265.

Chapter 6

Alternative Performance Indicators for Optimising Container Assignment in a Sychromodal Transportation Network



M. R. Ortega del Vecchyo

Abstract Several different attributes are deemed important in the container-to-mode assignment on a sychromodal transportation network. This chapter proposes a way to quantify several of this different attributes: Robustness, Flexibility and Customer Satisfaction. These attributes are used as alternative objectives when optimising the container assignment in a Sychromodal Transportation Network, modelling it as a Minimum Cost Multi-Commodity Flow on a Space-Time Network.

Introduction

We start here with the MCMC flow problem as defined in section “[Modelling the Problem as a MCMC Flow Problem on a Space–Time Network](#)”. The question ‘what to optimise’ in such networks has been scarcely addressed in the literature, despite it being a recognised problem in [8] and [11]. In [8], it is proposed that cost, service, frequency, service time, delivery reliability, flexibility and safety are all performance indicators. In [10], customer responsiveness and quality as objectives are also objectives. Next to this, in supply chain logistics in general, there is a growing attention for environmental risks and sustainability [1, 2, 5, 6, 13, 14]. Note that we do not require the alternative objectives or performance indicators to be the objective value. They can also be used as constraint to guarantee a certain (minimum or maximum) value.

In most papers, however, cost of the operation and service time are still the only used objectives, and other attributes are neglected [12]. As it is stated in [4], many transportation planning problems are solved via a deterministic optimisation-based tool where the lowest cost solution is chosen. However, the used forecasts can be very inaccurate and realisations may lead to new plans that has to be changed drastically or might be unfeasible. In the literature, these problems are sometimes addressed using the terms reliability, flexibility, robustness and resilience, where

M. R. Ortega del Vecchyo (✉)
TNO, The Hague, The Netherlands

different terms can be used for similar things. In [7], definitions are proposed as an attempt to encompass consistently the meaning intended in other papers for each concept:

- Robustness is the ability to endure foreseen and unforeseen changes in the environment without adapting.
- Flexibility is the ability to react to foreseen and unforeseen changes in the environment in a pre-planned manner.
- Agility is the ability to react to unforeseen changes in the environment in an unforeseen and unplanned manner.
- Resilience is the ability to survive foreseen and unforeseen changes in the environment that have a severe and enduring impact.

In this work, in the context of transportation planning, we will use the following definitions based on meanings explained both implicitly and explicitly on several sources such as [3, 4, 7, 9, 12]:

- Robustness is the capacity of a plan to overcome uncertain events or disturbances in the future and still be carried over as planned.
- Flexibility is the capacity of a plan to adapt to uncertain events or disturbances, when these force the plan not to be able to be carried on anymore.

We propose, define and compare different performance indicators that are in play on a synchromodal transportation chain. As indicated, we start using the formulation of section “[Modelling the Problem as a MCMC Flow Problem on a Space–Time Network](#)”. Additionally, we consider the following assumptions for our problem:

1. At every timestamp, there are an unlimited number of trucks going from any location to any other location. These trucks are more expensive and quicker than any other means of transportation.
2. Truck price is fixed and is the same for every OD pair.
3. Every number of containers has the possibility of remaining idle in a given location with no additional cost.
4. Only one arc from (A,t) to (B,s) is allowed.

The remainder of this chapter is organised as follows. In the next section, the theoretical basis of this work is presented. Next, in section “[Attributes](#)”, the attributes that are considered are built from concept to implementation. Finally, in section “[Conclusions](#)”, we will present the conclusions and give directions for further research.

Attributes

In the previous section, we gave the new definitions of Robustness and Flexibility. These are important to cope with uncertain events. In practice, an ‘uncertain event’ on a transportation network can come in many different forms: disturbances in

handling times upon arrival on a terminal, arrival of new orders, assignment of time slots for arrival of certain modes and so on. Note that the relevance of these uncertainties usually varies depending on the different time-scales. Planners often deal with these uncertainties via strategic behaviour, that is, by acknowledging these uncertainties and taking them into consideration when making their decisions. In this study, we restrict ourselves to consider the uncertain events of travel times and handling times on terminals. Therefore, the definitions of the concepts can be read as follows:

- Robustness is the capacity of a plan to overcome delays in travel times and handling times on terminals and still be carried on as planned.
- Flexibility is the capacity of a plan to adapt to delays in travel times and handling times on terminals when these force the plan not to be able to be carried on anymore.

Next to these two, we take into account the lateness of a whole plan via the attribute customer satisfaction. These three attributes will be defined in the following sections, and for each a numerical example will be used to show the impact on the MCMCF problem.

Robustness

To illustrate the meaning of robustness in our model, we first show how we would like to quantify robustness for a simple case. In an STN with a number of orders, we want to give a numerical value to each solution of the problem, that is, a value per transportation plan. In the case of a single order, we may assign a value to a path. Consider a path P such as the one in Fig. 6.1 with an OD pair $((A, 0), (C, 5))$, that is, with a source on location A at time 0 and sink on location C at time 5. The robustness value is meant to represent how likely this plan can endure despite delays in travel times and handling times, that is, we need to see how likely the transportation mode from location B at time 3 to location C at time 4 (arc from $(B, 3)$ to $(C, 4)$) will be able to take place for path P despite delay in travel time from A to B and handling times at B . If the resource doing the trip $((A, 0), (B, 1))$ is also the one on the trip $((B, 3), (C, 4))$, then there will be no handling at location B , and thus the flow of containers through this path will certainly make this connection. Otherwise, the handling at location B will depend on these factors:

- The number of containers going through arc $((A, 0), (B, 1))$, since all of these will be handled at location B , which is, in this case, the flow going through path P .
- The number of timestamps available from the estimated time of arrival to arc B , which is 1, and the time of departure of the trip to C , which is 3.

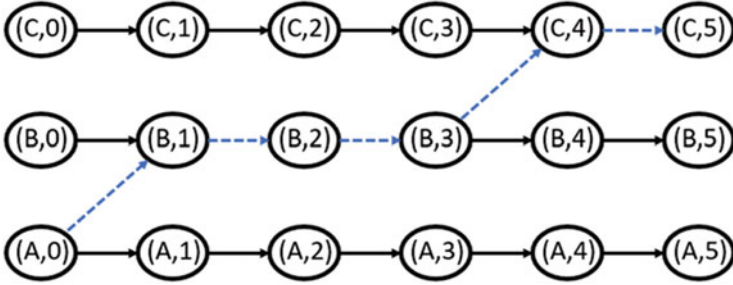


Fig. 6.1 Robustness of a path (in dotted blue)

This kind of link is what we refer to as an event, and we define the robustness of a plan with respect to the robustness of these events.

Definition 6.1 For a given path P on a space-time graph, we say that $e = ((A, t_0), (B, t_1), (B, t_2))$ is an **event** of the path P if the path $((A, t_0), (B, t_1), (B, t_1 + 1), \dots, (B, t_2), (C, t_3))$ for some $C \neq B$ and $B \neq A$ is a subpath of P , and the resource of the trip $((A, t_0), (B, t_1))$ is a different resource from the one of trip $((B, t_2), (C, t_3))$. Also, $e = ((A, t_0), (B, t_1), (B, t_2))$ is an event of P if the path $((A, t_0), (B, t_1), (B, t_1 + 1), \dots, (B, t_2))$ is a sub subpath of P and (B, t_2) is the last node on P . If the event is of the latter form we refer to it as the **last event** of P . We use the short notation $e \in P$ to denote that the event e is an event of the path P . For a path-based multi-commodity flow problem Pr on a space-time graph, we say that e is an event of the problem Pr if it is an event of a path P of an OD pair in Pr . We use the short notation $e \in Pr$ to denote that the event e is an event of the problem Pr . If x_P is the flow variable of a path P and F is a solution to Pr , the flow on an event is defined as $F_e = \sum_{P \in P(e)} x_P$ where $P(e) = \{P \in \cup_k P(k) | ((A, t_0), (B, t_1)) \in P\}$. \square

In the path of Fig. 6.1, if the resource of edge $((A, 0), (B, 1))$ is a different resource from the one in edge $((B, 3)(C, 4))$, the path has the event: $e_1 = ((A, 0), (B, 1), (B, 3))$. In any case, the last event $e_2 = ((B, 3), (C, 4), (C, 5))$ in on the path. Our main assumption when determining robustness of an event is that the information in the three elements that constitute the event and the flow of the event are necessary and sufficient to determine the robustness of the event. More specifically, we determine the robustness of an event via a measure of robustness, which depends on the amount of flow f and the number of timestamps t .

Definition We say that a function r' is a **measure of robustness** if $r' : \mathbb{R}^+ \times \mathbb{Z}^+ \rightarrow [0, 1]$ and the following holds: \square

- $r'(0, t) = 1$ for all t , $\lim_{f \rightarrow \infty} r'(f, t) = 0$ for any fixed t , $r'(f, t)$ is a decreasing function of f for any fixed t .
- $r'(f, 0) = \epsilon$ for $\epsilon > 0$ a number close to zero, $\lim_{t \rightarrow \infty} r'(f, t) = 1$ for any fixed f , $r'(f, t)$ is an increasing function of t for any fixed f .

We define the robustness $r(e, f)$ of an event $e = ((A^e, t_0^e), (B^e, t_1^e), (B^e, t_2^e))$ with flow f as $r(e, f) = r'(f, t_2^e - t_1^e)$. If the first argument f is omitted, then $r(e) = r'(F_e, t_2^e - t_1^e)$.

Thus, the two variables of the function $r'(f, t)$ are thought of as denoting the amount of flow of the first arc of the event and the timestamps available. The properties of the measure of robustness attempt to be the minimum requirement we would expect for a way to measure the robustness of such an event: if the amount of flow is small with respect to the number of timestamps, then quite likely (probability close to 1), the event will be a success in terms of arrival before departure time of the next transportation mode, whereas if the amount of flow is large with respect to the number of timestamps, then it might be difficult to make this connection. We should note that whenever a specific flow is considered large or small depends of course on the units considered for each timestamp, but in any case, the equalities and the limits must hold. The reason for the small value $r'(f, 0) = \epsilon$ is that just-in-time connections might not be very robust, but they can still be made.

Definition Let F be a solution flow for a path-based multi-commodity flow problem Pr on a space-time graph. We define the **robustness of the solution** $R(F)$ as the product of the robustness of the events of the plan, that is,

$$R(F) = \prod_{e \in Pr} r(e) = \prod_{e \in Pr} r'(F_e, t_2^e - t_1^e).$$

Thus, in order to quantify the robustness, a robustness measure r' must be specified. We propose the function

$$r'(f, t) = \begin{cases} e^{-\lambda \frac{f}{t}} & \text{if } t > 0 \\ e^{-\lambda \frac{f}{0.5}} & \text{if } t = 0 \end{cases}$$

with $\lambda > 0$ a parameter to be specified depending on the units that represent each timestamp. For simplicity, if an event e is such that $t_2^e - t_1^e = 0$, we write $\frac{F_e}{t_2^e - t_1^e}$ when we actually mean $\frac{F_e}{0.5}$. Then, robustness is defined as

$$R(F) = \prod_{e \in Pr} r'(F_e, t_2^e - t_1^e) = \prod_{e \in Pr} e^{-\lambda \frac{F_e}{t_2^e - t_1^e}}.$$

Maximising the robustness function is the same as maximising the logarithm of the robustness function, such that

$$\begin{aligned} \log R(F) &= \log \prod_{e \in Pr} e^{-\lambda \frac{F_e}{t_2^e - t_1^e}} = \sum_{e \in Pr} \log e^{-\lambda \frac{F_e}{t_2^e - t_1^e}} \\ &= \sum_{e \in Pr} -\lambda \frac{F_e}{t_2^e - t_1^e} = -\lambda \sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e}. \end{aligned}$$

Since $\lambda > 0$, maximising the robustness function is equivalent to minimising

$$\sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e}. \quad (6.1)$$

This expression is linear with respect to the flow path-based variables x_P , and the sum depends only on the events of the problem, which are independent of the solution proposed. Therefore, this expression can be constructed as a linear objective on a linear program (LP). We refer to the expression 6.1 as the **robustness expression**.

Notice that $R(F)$ tends to decrease if the number of events in the problem $|\{e \in Pr\}|$ increases. For this reason, in order to treat instances of different sizes in a similar way, it is practical to introduce the **geometric mean robustness of the solution** $MR(F)$ defined as

$$MR(F) = \left(\prod_{e \in Pr} r(e) \right)^{\frac{1}{|\{e \in Pr\}|}}.$$

Then, we obtain, using the same robustness measure as before,

$$\log MR(F) = \frac{\log \prod_{e \in Pr} e^{-\lambda \frac{F_e}{t_2^e - t_1^e}}}{|\{e \in Pr\}|} = \frac{-\lambda}{|\{e \in Pr\}|} \sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e}.$$

The objective $\sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e}$ is the linear expression that represents robustness when the measure of robustness is chosen to be $r'(f, t) = e^{-\lambda \frac{f}{t}}$. Robustness and mean robustness are maximised when this expression is minimised, regardless of the $\lambda > 0$. The equalities

$$\sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e} = -\frac{\log R(F)}{\lambda}$$

and

$$\sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e} = -\frac{\log MR(F) |\{e \in Pr\}|}{\lambda}$$

allow us to calculate the robustness and the mean robustness of a solution and give a value for the robustness expression when we are aiming for a robustness or mean robustness value. Also, a good estimate of λ should be chosen when the unit of the timestamps is determined, from the interpretation given to $r'(f, t) = e^{-\lambda \frac{f}{t}}$.

In order to gauge the influence of robustness, we present some results about the impact of the robustness expression as a constraint on the MCMCF problem on a

space-time graph. Suppose we want to solve an instance with time horizon $T = 80$, 10 terminals, 200 orders, and that each timestamp represents one hour. We assume that quite certainly (with probability .90) 10 containers can be handled in one hour, that is to say

$$\begin{aligned} r'(f, t) = e^{-\lambda(10)} > 0.90 &\implies -10\lambda > \log(0.90) \\ &\implies \lambda < 0.0105. \end{aligned}$$

We fix $\lambda = 0.01$. By solving the MCMCF problem without involving robustness, we obtain an optimal solution F_1 with a mean robustness of

$$MR(F_1) = \exp\left(\frac{-\lambda}{|\{e \in Pr\}|} \sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e}\right) = 0.7761$$

and a cost $C(F_1) = 204,399$. If we wish to improve the mean robustness slightly to 0.78, we input the constraint $\sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e} = -\frac{\log MR(F)|\{e \in Pr\}|}{\lambda} \leq -\frac{\log(0.78)|\{e \in Pr\}|}{\lambda} = 7602.9175$ and rerun the solver. This constraint will guarantee, the solution will have a mean robustness of at least 0.78

This change, as little as it may seem, already brings some significant differences on the new solution F_2 . F_1 and F_2 differ in the transportation planning of 21 out of the 200 orders, despite the fact that $C(F_2) = 204,400.96 \approx C(F_1) + 1$ and both have the same number of trucks used, that is, the plan is altered without compromising cost.

Comparison between increasingly robust solutions reveals the following tendencies about the more ‘‘average robust’’ solutions:

- They tend to prefer paths that have less connections between different resources.
- They tend to prefer earlier arrival.
- They tend to prefer paths connected by the same resource.

The first characteristic may be helpful to prevent from possible handling costs incurred in terminals. The second characteristic can be beneficial so that future resources are allocated for future uncertain happenings, but it can affect if there are costs for long idle times at destination terminals.

Flexibility

The second attribute proposed is flexibility, which was defined as the capacity of a plan to adapt to delays in travel times and handling times on terminals, when these delays force the plan not to be able to be carried out anymore. To calculate the flexibility of a single path (simple case with a single commodity, and one path carries all the flow) such as the one described in Fig. 6.2, we first identify those links

in the path that could be problematic in terms of flexibility as we have defined it. As in the case of robustness, we refer to these problematic links as events. In this path, with OD pair $((A, 0), (C, 5))$, if there was a delay on the transportation arc from $(A, 1)$ to $(B, 2)$ or on the handling time at B such that the connection with the arc $(B, 3)$ to $(C, 4)$ is lost (in this case, the delay made the trip arrive at time 4), there is still the possibility to take the arc from $(B, 4)$ to $(C, 5)$. The flexibility of this path is defined in terms of the cost of this alternative route with respect to the cost of the original route. In the case where there is more than one event on the path, the flexibility of the path is done with respect to the cost of the alternative routes corresponding to each of these events. In order to state unambiguously the flexibility of a flow, a series of definitions are necessary. The definition of event is still as in Definition 6.1, except that in this context, last events are not considered events.

Definitions

- For a path P on an STN and an event $e = ((A, t_1), (B, t_2), (B, t_3))$ on the path, we define the subpath P_e with respect to e as the subpath of P that contains all the nodes from (B, t_3) onward. In the case of the example in Fig. 6.2, for the event $e = ((A, 1), (B, 2), (B, 3))$, the subpath defined is $P_e = ((B, 3), (C, 4), (C, 5))$.
- For a solution F of a multi-commodity flow problem on an STN G , we denote by $G \setminus F$ the STN G whose arcs' capacity has been lowered according to the flow of F , that is, the capacity of an arc in $G \setminus F$ is the capacity of the arc on G minus the flow passing through that arc on F .
- For a pair of nodes (A, t_1) and (B, t_2) on a space-time graph G and a positive real number r , we denote by $\text{mincost}((A, t_1), (B, t_2), r)_G$ the cost of the optimal solution of the minimum cost flow problem with source node (A, t_1) , sink node (B, t_2) and flow r in G .
- For a path P with flow x_P of a solution F of a multi-commodity flow problem on an STN G and an event $e = ((A, t_1), (B, t_2), (B, t_3))$ on the path, we define the anti-flexibility $\varphi_{G \setminus F}(e, x_P)$ of the event as the least cost that would be incurred

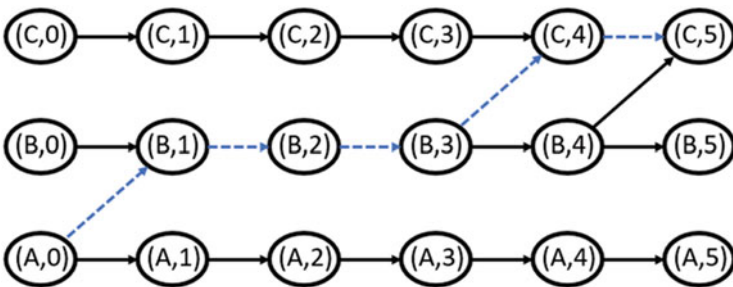


Fig. 6.2 Flexibility of a path (in dotted blue)

if the trip scheduled from A at time t_1 to B at time t_2 would arrive one timestamp after time t_3 to B . That is,

$$\begin{aligned} & \varphi_{G \setminus F}(e, x_P) \\ &= \text{mincost}((B, t_3 + 1), (S_P, t_P), x_P)_{G \setminus F} - C(P_e)x_P. \end{aligned}$$

Here, $C(P_e)$ is the cost of the subpath P_e and (S_P, t_P) is the last node on P . Notice the dependency of the min-cost algorithm on the solution flow F as well as on G , that is, the capacity of the arcs on G is lowered corresponding to the flow F . We call the above anti-flexibility because $\varphi_{G \setminus F}(e, x_P)$ decreases as the flexibility of the event increases, according to our definition of flexibility.

- For a solution flow F of a path-based multi-commodity flow problem on a space-time graph G and a robustness function r , we define its anti-flexibility $\phi_G(F)$ as

$$\phi_G(F) = \sum_{P \in F, x_P > 0} \sum_{e \in P} \varphi_{G \setminus F}(e, x_P)(1 - r(e)).$$

In our case, the robustness function r used is the one implied by the exponential robustness measure $r'(f, t) = e^{-\lambda \frac{f}{t}}$, as shown in the previous section. The last expression is a sum of all the incurred costs that could happen on the plan from delays, this is, the expression we seek to minimise, and however, it is far from linear in terms of the flow variables of the paths. Notice also that in order to calculate the anti-flexibility of an event of a path P , the value of the flow variable x_P must be known in advance, which is of course not the case. In addition, a constraint whose coefficients involve solving several min-cost problems can be very computationally heavy. Thus, a linear expression that overcomes these challenges is sought for in order to include it in an LP formulation. For this purpose, the following linearisation is constructed.

Definition For a path P on an STN G and an event $e = ((A, t_1), (B, t_2), (B, t_3))$ on the path, we define the *linear anti-flexibility* of the event $\iota_G(e)$ as

$$\iota_G(e) = C(P_e^d) - C(P_e),$$

where P_e^d is the shortest (least costly) path on G from $(B, t_3 + 1)$ to (S_P, t_P) . For a path P on an STN G and a robustness function r , we define the *linear anti-flexibility* of the path $\iota_G(P)$ as

$$\iota_G(P) = \sum_{e \in P} \iota_G(e)(1 - r(e, c)),$$

where c is an arbitrary fixed number, preferably close to the average flow of a path. We fix c to be the lowest possible order size. \square

Table 6.1 Trade-off between anti-flexibility and cost

	Linear anti-flexibility	Cost	Anti-flexibility
F_1	5075	153,655	6079
F_2	3000	153,843	3502
F_3	2000	154,285	2665
F_4	1000	155,724	1567
F_5	650	156,471	1341

Now, the linear anti-flexibility of a path is a coefficient relatively easy to calculate. With it, we define the *linear anti-flexibility expression*

$$\sum_P \iota_G(P)x_P. \quad (6.2)$$

To explore the effect of the linear anti-flexibility expression obtained, we consider an MCMCF problem on an instance with the same characteristics as the one in the robustness example. The problem without any linear anti-flexibility constraint yields a solution F_1 with an anti-flexibility value $\phi_G(F_1) = 6078$, a linear anti-flexibility of $\sum_P \iota_G(P)x_P = 5075$ and a cost of $C(F_1) = 153,655$. By adding the linear flexibility as a constraint and changing the constraint value, we obtain costs and anti-flexibility values as shown in Table 6.1. Note that the anti-flexibility is massively reduced by adding barely any cost with a linear flexibility value of 3000. Also, with a value of 2000, anti-flexibility is reduced by more than half.

In terms of how different the solution is, F_2 has a different plan for 34 orders when compared to F_1 , whereas F_4 has 39 orders with a different plan with respect to plan F_1 . This suggests that the linear anti-flexibility constraint affects the plan considerably.

Comparison between the solutions reveals the following tendencies about the more “flexible” solutions:

- They tend to prefer trips with a cheap backup alternative in the future (notice that flexibility of a path can be negative, meaning that the backup route is cheaper).
- They tend to prefer single link trips or trips from the same voyage.

As it was defined, anti-flexibility represents the expected extra costs that will be incurred on the plan, assuming there is a full refund for the arcs that were planned to be used but were not reached on time. Of course, this refund does not necessarily take place, making the anti-flexibility more of a lower bound on the expected extra costs. If the expected costs are to be minimised, then it may be appropriate to minimise the lower bound on this expected cost, that is, the sum of costs and anti-flexibility. Unfortunately, anti-flexibility cannot be put on the LP, so one must rely on the use of the linear anti-flexibility to reach a low value for the sum of costs and anti-flexibility.

For the case of one commodity that can be served with a single path, given a collection of possible solution paths, observe that if the anti-flexibility of a path is minimised, then the path obtained might have negative values, meaning that the

backup paths are cheaper than the solution path. If these paths are much cheaper, then the anti-flexibility is much lower. Of course, this has to be avoided, and it shows that the anti-flexibility is an expression that comes with trade-offs. Anti-flexibility measures how cost-effective the backup plans are with respect to the chosen plan, and although it is good to have these alternatives cost-effective, it is probably not good to have them much cheaper than the chosen plan.

Customer Satisfaction

From the point of view of a customer, perhaps the most important thing of an order is its timely delivery. However, as our meetings with different stakeholders in practice revealed, this attribute is dependent on the client. That is to say, some clients might have no problem if their order arrives later than the agreed arrival time, whereas for others it might be crucial to have it on time. In our model, we consider this customer dependent lateness as an independent attribute. We fix a maximum amount of lateness that can be allowed for each order, and in order to measure customer satisfaction of a solution plan, we observe how late the arrival of each order is, taking into account the priority of each client. Since the order cannot be considered delivered until every container has arrived at the destination, we make the following definitions:

Definition For a solution flow F of a multi-commodity flow problem Pr on a space-time graph and an order $o \in Pr$ of the problem, we define the delivery time of the order $d(o)$ as the maximum of the arrival times of the containers on that order. \square

Definition For each order o with an OD pair and $t \in \{0, 1, 2, \dots, r\}$ a number of timestamps, we refer to the satisfaction $s(o, t) \in [0, 1]$ as the number that reflects how satisfied the customer of order o will be if the order arrives t timestamps after the due time. For a fixed o , we assume $s(o, t)$ to be decreasing on t . The maximum number of lateness $r < T$ is fixed for computational ease. \square

Notice that this can be extended to a case where there is also penalty for early arrival, or even further, for arrival at any specific timestamp per order. However, this is not done in this chapter.

Definition For a solution flow F of a multi-commodity flow problem Pr on a space-time graph and a family of numbers $w(o) \in [0, 1]$ such that $\sum_{o \in Pr} w(o) = 1$, we define the customer satisfaction as

$$\left(\sum_{o \in Pr} s(o, t_o) w(o) \right)^2, \quad (6.3)$$

where t_o is the delay in a number of timestamps of order o . \square

Customer satisfaction can be implemented via several indicator variables, one per order, per number of timestamps delayed. However, given the addition of several binary variables, considering customer satisfaction comes with a computational burden.

As with the previous attributes, we use an instance with 200 orders to see the effect of the customer satisfaction expression as a constraint, with randomly generated weights. The maximum number of delayed timestamps r is set to 10. Without any constraint related to customer satisfaction, we obtain a solution F_1 with a customer satisfaction value of 0.8190. By increasingly constraining the value of customer satisfaction, we obtain the costs summarised in the following table.

	Customer satisfaction	Cost
F_1	0.8190	60,082
F_2	0.8525	60,094
F_3	0.9	60,548
F_4	0.95	62,474
F_5	0.98	64,488

The table shows that substantial cost reduction can be obtained at the expense of customer satisfaction, i.e., timely delivery. Overall, by adding the attribute of customer satisfaction to the model, and stretching the possibilities of delayed containers, we obtain a new range of solutions and a way to compare their effectiveness.

Conclusions

We constructed linear formulations of three different attributes: robustness, flexibility and customer satisfaction. These formulations can be used as objectives in optimising a synchronodal transportation problem. As optimisation environment we used the Minimum Cost Multi-Commodity Flow formulation on a space-time network. As expected the alternative objectives have a trade-off with other objectives, such as cost. To handle this, further research is done on multi-objective optimisation, where various objectives are considered together and a pareto-front of alternative optimal solutions are presented. Another direction that is being investigated is the use of robust optimisation methods within the synchronodal transportation approach.

References

1. Ahluwalia, P. K., & Nema, A. K. (2006). Multi-objective reverse logistics model for integrated computer waste management. *Waste Management & Research*, 24(6), 514–527.
2. Baykasoğlu, A., & Subulan, K. (2016). A multi-objective sustainable load planning model for intermodal transportation networks with a real-life application. *Transportation Research Part E: Logistics and Transportation Review*, 95, 207–247.
3. Beuthe, M., & Bouffieux, C. (2008). Analysing qualitative attributes of freight transport from stated orders of preference experiment. *Journal of Transport Economics and Policy*, 42(1), 105–128.
4. Caplice, C., & Jauffred, F. (2014). Balancing robustness and flexibility in transportation networks <http://ctl.mit.edu/sites/ctl.mit.edu/files/caplice-SCB-MarApr2014.pdf>
5. Caramia, M., & Dell’Olmo, P. (2008). *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms*. Springer.
6. Govindan, K., Paam, P., & Abtahi, A. R. (2016). A fuzzy multi-objective optimization model for sustainable reverse logistics network design. *Ecological Indicators*, 67, 753–768.
7. Husdal, J. (2010). A conceptual framework for risk and vulnerability in virtual enterprise networks. Managing risk in virtual enterprise networks: implementing supply chain principles.
8. Ishfaq, R., & Sox, C. R. (2010). Intermodal logistics: The interplay of financial, operational and service issues. *Transportation Research Part E: Logistics and Transportation Review*, 46(6), 926–949.
9. Miller-Hooks, E., Zhang, X., & Faturechi, R. (2012). Measuring and maximizing resilience of freight transportation networks. *Computers & Operations Research*, 39(7), 1633–1643.
10. Ramezani, M., Bashiri, M., & Tavakkoli-Moghaddam, R. (2013). A new multi-objective stochastic model for a forward/reverse logistic network design with responsiveness and quality level. *Applied Mathematical Modelling*, 37(1), 328–344.
11. Riessen, B. V., Negenborn, R. R., & Dekker, R. (2015). Synchronodal container transportation: An overview of current topics and research opportunities. In *International Conference on Computational Logistics* (pp. 386–397). Springer.
12. SteadieSeifi, M., Dellaert, N., Nuijten, W., Woensel, T. V., & Raoufi, R. (2014). Multi-modal freight transportation planning: A literature review. *European Journal of Operational Research*, 233(1), 1–15.
13. Tuzkaya, G., Kilic, H. S., & Aglan, C. (2016). A multi-objective supplier selection and order allocation model for green supply chains. *Journal of Military and Information Science*, 4(3), 87–96.
14. Xifeng, T., Ji, Z., & Peng, X. (2013). A multi-objective optimization model for sustainable logistics facility location. *Transportation Research Part D: Transport and Environment*, 22, 45–48.

Chapter 7

Decision Making in a Dynamic Transportation Network: A Multi-Objective Approach



M. R. Ortega del Vecchyo

Abstract Multiple different attributes are important in the container-to-mode assignment in a transportation network. This work proposes an interactive multi-objective optimisation approach for planners of those transportation networks. This approach offers a range of solutions according to her/his preferences and offers the opportunity to seek for new ones if the planner is not satisfied with the solutions found so far.

Introduction

To use the alternative performance indicators as presented in the previous chapter, we propose an interactive approach based on multi-objective optimisation, which is meant to be used as a decision support tool for a transportation planner in a synchromodal context. The mathematical framework we use as basis in the approach is again the *Multi-Commodity Flow* problem as defined in section “[Modelling the Problem as a MCMC Flow Problem on a Space–Time Network](#)”. On this model, we build an interactive multi-objective analysis method, inspired by Miettinen et al. [1].

The remainder of this section is organised as follows. In section “[Multi-Objective Analysis](#)”, we present the MCMCF problem and propose and define the new objectives. Then, we present the interactive approach and illustrate this by applying it on an example. Finally, in section “[Conclusions and Future Work](#)”, we will present the conclusions and give directions for further research.

M. R. Ortega del Vecchyo (✉)
TNO, The Hague, The Netherlands

Multi-Objective Analysis

In this section, we use the MCMCF problem formulation of section “[Modelling the Problem as a MCMC Flow Problem on a Space–Time Network](#)”. Instead of only minimising costs, now multiple objectives are proposed as defined in the previous chapter. These objectives were constructed using the definitions:

- Robustness is the capacity of a plan to overcome delays in travel times and handling times on terminals and still be carried on as planned.
- Flexibility is the capacity of a plan to adapt to delays in travel times and handling times on terminals when these force the plan not to be able to be carried on anymore.
- Customer satisfaction indicates how satisfied the customer will be if his order arrives a certain time after the due date.

We will give a short derivation of the mathematical definitions of those objective here.

For a given path P on a space-time graph, we say that $e = ((A, t_0), (B, t_1), (B, t_2))$ is an **event** of the path P if the path $((A, t_0), (B, t_1), (B, t_1 + 1), \dots, (B, t_2), (C, t_3))$ for some $C \neq B$ and $B \neq A$ is a sub-path of P , and the resource of the trip $((A, t_0), (B, t_1))$ is a different resource from the one of trip $((B, t_2), (C, t_3))$. Also, $e = ((A, t_0), (B, t_1), (B, t_2))$ is an event of P if the path $((A, t_0), (B, t_1), (B, t_1 + 1), \dots, (B, t_2))$ is a sub-path of P and (B, t_2) is the last node on P . If the event is of the latter form, we refer to it as the **last event** of P . We use the short notation $e \in P$ to denote that the event e is an event of the path P . For a path-based multi-commodity flow problem Pr on a space-time graph, we say that e is an event of the problem Pr if it is an event of a path P of an OD pair in Pr . We use the short notation $e \in Pr$ to denote that the event e is an event of the problem Pr . If x_P is the flow variable of a path P and F is a solution to Pr , the flow on an event is defined as $F_e = \sum_{P \in P(e)} x_P$, where $P(e) = \{P \in \cup_k P(k) \mid ((A, t_0), (B, t_1)) \in P\}$.

Let F be a solution flow for a path-based multi-commodity flow problem Pr on a space-time graph and the robustness measure $r'(f, t) = e^{-\lambda \frac{f}{t}}$, with $\lambda > 0$ a parameter to be specified depending on the units that represent each timestamp. We introduce the **geometric mean robustness of the solution** $MR(F)$ as $MR(F) = \left(\prod_{e \in Pr} r(e) \right)^{\frac{1}{|\{e \in Pr\}|}}$.

Definition The geometric mean robustness is minimised by minimising the log of the geometric mean robustness of the solution, calculated by

$$\log MR(F) = \frac{\log \prod_{e \in Pr} e^{-\lambda \frac{F_e}{t_2^e - t_1^e}}}{|\{e \in Pr\}|} = \frac{-\lambda}{|\{e \in Pr\}|} \sum_{e \in Pr} \frac{F_e}{t_2^e - t_1^e}.$$

For a path P on an STN and an event $e = ((A, t_1), (B, t_2), (B, t_3))$ on the path, we define the sub-path P_e with respect to e as the sub-path of P that contains all the nodes from (B, t_3) onward. Also, for a solution F of a multi-commodity flow problem on an STN G , we denote by $G \setminus F$ the STN G whose arcs' capacity has been lowered according to the flow of F , that is, the capacity of an arc in $G \setminus F$ is the capacity of the arc on G minus the flow passing through that arc on F . Next, for a pair of nodes (A, t_1) and (B, t_2) on a space-time graph G and a positive real number r , we denote by $\text{mincost}((A, t_1), (B, t_2), r)_G$ the cost of the optimal solution of the minimum cost flow problem with source node (A, t_1) , sink node (B, t_2) and flow r in G . For a path P with flow x_P of a solution F of a multi-commodity flow problem on an STN G and an event $e = ((A, t_1), (B, t_2), (B, t_3))$ on the path, we define the anti-flexibility $\varphi_{G \setminus F}(e, x_P)$ of the event as the least cost that would be incurred if the trip scheduled from A at time t_1 to B at time t_2 would arrive one timestamp after time t_3 to B . That is, $\varphi_{G \setminus F}(e, x_P) = \text{mincost}((B, t_3 + 1), (S_P, t_P), x_P)_{G \setminus F} - C(P_e)x_P$. Here, $C(P_e)$ is the cost of the sub-path P_e and (S_P, t_P) is the last node on P . Notice the dependency of the min-cost algorithm on the solution flow F as well as on G , that is, the capacity of the arcs on G is lowered corresponding to the flow F . We call the above anti-flexibility because $\varphi_{G \setminus F}(e, x_P)$ decreases as the flexibility of the event increases, according to our definition of flexibility.

Definition For a solution flow F of a path-based multi-commodity flow problem on a space-time graph G and a robustness function r , we define its anti-flexibility $\phi_G(F)$ as $\phi_G(F) = \sum_{P \in F, x_P > 0} \sum_{e \in P} \varphi_{G \setminus F}(e, x_P)(1 - r(e))$ \square

Definition For a solution flow F of a multi-commodity flow problem Pr on a space-time graph and a family of numbers $w(o) \in [0, 1]$ such that $\sum_{o \in Pr} w(o) = 1$, we define the customer satisfaction as $(\sum_{o \in Pr} s(o, t_o)w(o))^2$, where t_o is the delay in a number of timestamps of order o . \square

Definition As last objective, we define Cost as $\sum_k \sum_{P \in P(k)} C(P)x_P$. \square

Multi-Objective Approach

First we use a lexicographic method to obtain a Pareto solution. For the lexicographic method, we need to rank the objective functions in order of importance. The lexicographic method can be very stiff in some problems, since it does not allow for any decrease in value from the top ranked objectives to increase less important objectives. For this reason, we also consider a slight variation of the lexicographic method: when optimising cost, look at the value of number of trucks and constraint the problem with the respect to this number of trucks instead of cost. Notice that, strictly, with this procedure, we cannot guarantee that the solution obtained is a Pareto optimal solution, and therefore, if a Pareto optimal solution is needed, then one should use the usual lexicographic method. We propose several different orderings for obtaining the (Pareto) solutions, see Table 7.1.

Table 7.1 Three lexicographic orders

First	Second	Third
Cost (Truck)	Cost (Truck)	Cost (Truck)
Linear anti-flexibility	Mean Robustness	Customer satisfaction
Customer satisfaction	Customer satisfaction	Linear anti-flexibility
Mean Robustness	Linear anti-flexibility	Mean Robustness

The first lexicographic order minimises costs and possible unforeseen costs, the second minimises costs and the need to change the plan and the third minimises costs and maximising customer satisfaction. Each of these orderings emphasises on one of the attributes constructed. The solutions provided by the lexicographic methods proposed will serve as a starting point for our interactive method.

Perhaps the most interesting methods of multi-objective optimisation for our case are the interactive methods. On these methods, the user is expected to have input on the algorithm to explore the solutions that are of interest. In [1], the main steps of an interactive method in multi-objective analysis are explained in the most general sense. Briefly, these steps are:

1. Provide the DM (decision maker) with the range that the different objectives can take, when possible.
2. Provide a starting Pareto optimal solution(s) to the problem.
3. Ask the decision maker for preference information.
4. Generate new Pareto optimal solution(s), show them and other possible relevant information to the DM.
5. Stop or go back to 3.

The purpose of the first two steps is to get the DM to be acquainted with the possibilities and limitations of the problem at hand. The last three steps will also provide further insight to the DM but are mainly geared towards finding the best Pareto optimal solution with respect to the DM preferences. This kind of methods has some nice benefits. The expertise of the DM is used as input on the method, which should give more satisfactory results from the point of view of the DM. The expert stirs the solution with respect to her or his preferences, and the method provides a solution towards these desired goals. Thus, in this method, the DM plays a very important role. Next, the decision maker does not need to know in advance the limitations of the problem with respect to the objectives. Rather, she or he learns from the problem at each iteration. Other benefits are that a variety of solutions will be provided, which is a desired feature for our case, and that there is no need to have preference for objectives in advance.

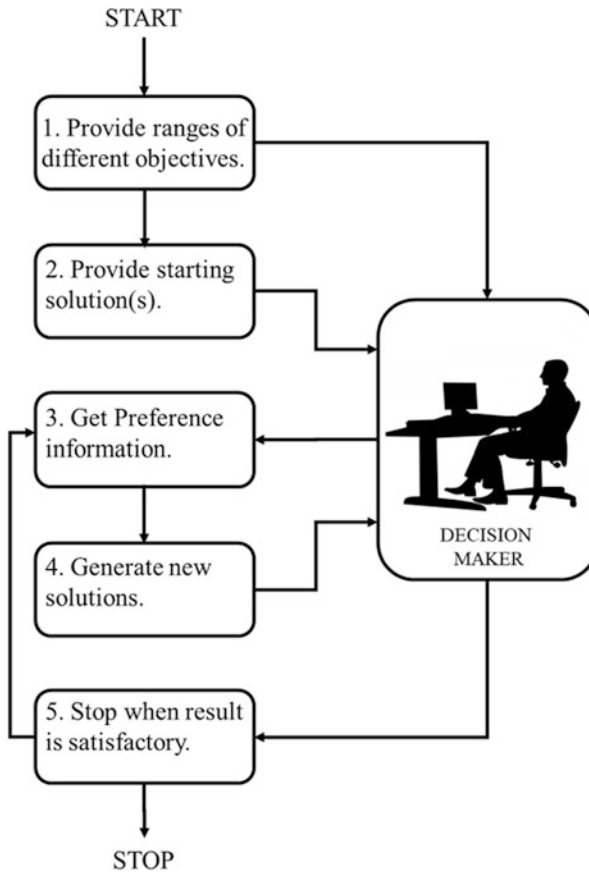


Fig. 7.1 Proposed approach in flowchart

Proposed Approach

The steps in the synchromodal planning setting we propose are (also depicted in Fig. 7.1):

1. Provide the decision maker with the range that the different objectives can take, when possible.
2. Provide a starting solution(s) to the problem.

We do not require the starting solution to be a Pareto optimal solution. However, a Pareto optimal solution can provide a valuable insight. The information of these solutions is gathered and kept for further assessment. Additionally, it is useful to build one optimal solution for a scalarisation of each objective (other than cost), that is, the optimal solution of the scalarisation of optimising one objective if the

cost is allowed a 1% increase with respect to the optimal cost (or, if more margin is given, a greater percentage).

3. Ask the decision maker for preference information.

In this step, the influence of the decision maker is crucial. The information available from the solutions of the problem found so far must be assessed and used to make decisions. This information may include but is not limited to:

- The value of the objectives of the solutions obtained so far, for example, the value of the base solution F_1 and the influence of achieving this cost in terms of the values of other attributes (obtained from the lexicographic methods).
- A better assessment of the range of values from the objectives done in step 1, that is, the limitations of the values of attributes.
- The approximate time for obtaining a solution, and given the time left for using the method, the number of solution extra we can expect to obtain.

From this information the following questions need to be answered:

- What objective to optimise next?
- What range to restrict the rest of the objectives to?
- Is there a minimum capacity needed for owned transport? If so, what percentage?
- Is there a specific arc whose capacity should be updated?
- Is there a path whose value should be constrained?

The last question includes whether some trip should not be used, some path must be fixed, some departure time of an arc must be fixed, etc. This characteristic allows the solver to process new information and therefore make it more synchronodal. When implemented, instead of building linear programmes from scratch, the code modifies the existing linear programme to optimise the required objective and satisfy the constraints selected, thus saving computational time.

4. Generate new Pareto optimal solution(s), show those solutions together with other possible relevant information to the DM.

A new LP is solved based on the questions obtained from the last step. The information of this solution is gathered and kept for further assessment

5. Stop, or back to 3.

Depending on whether a satisfactory solution has been provided, and on the time available, we either stop the method or reassess.

Example

We illustrate the functioning of the interactive method proposed by showing a use case. This method is meant to be used by a decision maker (which in this case is a planner) whose choices will stir the method in a certain direction, and thus the

method will give different solutions depending on the DM’s preferences. Therefore, in this example, we consider the presence of a hypothetical planner and conjecture the choices that this fictional DM may make.

Instance: we generate the problem by constructing the space-time graph and the orders to be dispatched on it with the following parameters: 15 terminals (A to O), Time Horizon of 200 time units, 400 orders (OD pairs uniformly randomly generated) 16 journeys of transport resources (other than trucks) and an allowed delay of 10 time units.

Next, we assume the capacity of owned transport within the system 154 and the capacity of a subcontracted transport uniformly distributes between 50 and 55. The number of containers per order can vary between 1 and 30. We assume a Truck price of 40 and the price per container in other transport uniformly distributed between 2 and 4.

To generate the values required for Customer satisfaction, we generate the random values $s(o, t) \in [0, 1]$ ensuring that for each o , $s(o, t)$ is decreasing with respect to t . The weights $w(o)$ are uniformly random generated by assigning $w'(o) = \text{unif}[0, 1]$ to each order and then setting the weight $w(o) = w'(o) / \sum_o w'(o)$.

Interactive method on instance: after the problem has been set, we follow the steps of the interactive method in the synchromodal context. This will be done at each point in time where realisations and new information becomes available and urges the planner to replan.

1. Provide the decision maker with the range that the different objectives can take, when possible. In this case, we have the following possible values: Cost: \mathbb{R}^+ , Anti-flexibility: \mathbb{R} , Robustness: $[0, 1]$ and Customer satisfaction: $[0, 1]$.
2. Provide starting solution(s) to the problem.

We first obtain the base solution F_1 by solving the LP with respect to costs, with no constraint on the other objectives. This results in a solution F_1 with the characteristics shown in Table 7.2.

Suppose the DM chooses to follow the first lexicographic method, then we add the constraint on trucked containers to be less than or equal to 3420 and optimise linear anti-flexibility. From this, we obtain the solution $F_{l,2}$, and, following

Table 7.2 Attribute values of the solutions of the lexicographic method

	F_1	$F_{l,2}$	$F_{l,3}$	$F_{l,4}$
Cost	146,387	147,812	147,695	147,653
Mean robustness	0.8778	0.8831	0.8834	0.8836
Anti-flexibility	2365.55	442.46	439.82	442.65
Customer satisfaction	0.8917	0.8907	0.8926	0.8926
Trucked containers	3420	3420	3420	3420
Linear anti-flexibility	116.79	20.73	20.73	20.73
Computational time (s)	245	65	85	70

the lexicographic method, solutions $F_{l,3}$ and $F_{l,4}$ with attribute values as shown in Table 7.2. These solutions show a very significant decrease in terms of anti-flexibility, a slight change in mean robustness, and barely any change in terms of customer satisfaction. Notice that neither cost nor anti-flexibility follows a strictly monotonic behaviour with respect to the solution number, despite the fact that this behaviour is expected from a lexicographic method. For the case of cost, this is a consequence of the fact that we are using a slight variation of the lexicographic method where we do not allow trucked containers to increase, instead of cost. For anti-flexibility, it is not expected to have any particular monotonic behaviour since it is not constrained directly on the LP. Further analysis on the full transportation plan file corresponding to each solution reveals that despite their similarity in terms of attributes, solutions $F_{l,1}$ and $F_{l,4}$ differ on the transport plan of 95 out of 400 orders.

The value of the attributes between solutions is relatively similar because the lexicographic method is quite restrictive, but the solutions provide us the insight of how much are the other attributes subject to change when the cost is (almost) rigid. Also, in this case, as it is often the case on lexicographic methods, the Pareto optimal solution $F_{l,4}$ is a very good proposal in terms of the attributes when compared to the other solutions obtained (that is, because all the attributes have been optimised at some stage). This solution will serve as a good reference for the capabilities of the solutions in terms of the attributes.

We now calculate the set of solutions corresponding to optimising each objective (that is, scalarisation) allowing 1% increase of cost over the optimal cost. We write F_f , F_r and F_{cs} for the solution corresponding to flexibility, robustness and customer satisfaction, respectively. The results are summarised in Table 7.3

From the scalarisation solutions obtained, we can see how much other attributes can be improved when we allow the cost to increase with as little as 1%: customer satisfaction can be improved with 0.6 and mean robustness can be increased with 0.3. In terms of anti-flexibility, there can be a reduction of almost 2000 units. It should be noted that the computational time to derive the solution F_{cs} is comparatively larger than the other ones.

Table 7.3 Attribute values of solutions

	F_1	F_r	F_f	F_{cs}	F_2	F_3	F_4
Cost	146,387	147,851	147,851	147,844	146,395	146,395	148,352
Mean robustness	0.8778	0.9056	0.8850	0.8859	0.8761	0.8812	0.8761
Anti-flexibility	2365.55	626.40	396.84	1498.30	755.53	799.44	801.94
Customer satisfaction	0.8917	0.9069	0.8937	0.9619	0.9060	0.9060	0.9035
Trucked containers	3420	3435	3443	3454	3420	3420	3474
Linear anti-flexibility	116.79	36.49	16.53	77.24	39.99	39.99	39.93
Computational time (s)	245	60	169	729	229.96	2128.02	215.33

3. Ask the decision maker for preference information.

At this stage, the decision maker has to assimilate the information she/he has of the problem so far, provided by the previous steps. We conjecture here our fictional DM's train of thought: the values of the attributes of the solutions provide a better idea of the range of the attributes; customer satisfaction is quite cost-effective to improve. Also, from $F_{l,2}$ and F_f , we see that anti-flexibility can be reduced substantially for little cost. Additionally, the DM knows that for this particular problem any plan with a customer satisfaction value over 0.9 is acceptable. Therefore, the DM chooses the next solution to be the solution F_2 of the scalarisation of optimising cost with a constraint on linear anti-flexibility of 40 and a customer satisfaction of 0.9.

4. Generate new Pareto optimal solution(s), show those solutions together with other possible relevant information to the DM.

Solution F_2 (see Table 7.3) has just an increase of 8 in terms of cost, and it provides a very substantial decrease on anti-flexibility, as well as an increase in customer satisfaction.

5. Stop or back to 3.

The solution seems satisfactory, but the DM decides to try to improve the robustness of the solution without compromising the other attributes, resulting in a new step (3):

(3) The DM decides to optimise with respect to robustness, with cost, linear anti-flexibility and customer satisfaction to be as good as the values in F_2 (similar to a lexicographic method).

(4) We obtain a solution F_3 (see Table 7.3).

Notice that the computational time to obtain F_3 is quite long, and therefore, depending on the time available, the DM may have stopped the simulation and picked a solution from the solutions obtained so far (probably F_2). This of course depends on the importance the DM gives to improving slightly the robustness of the solution in this circumstances.

(5) Suppose the DM chooses not to finish the simulation to obtain F_3 and reports F_2 as her/his solution of choice. The DM reviews the solution obtained and is informed that F_2 uses a specific trip that has been cancelled, which is represented by the arc $((K', 35), (L', 44))$ on the space-time network used for the problem. She/he is also informed that another trip from another transport used in F_2 will not be departing at the time the plan F_2 uses it, which is represented by arc $((C', 48), (I', 54))$. Additionally, a particular order has been specified to be served exclusively via truck, namely, order 2. The DM is therefore forced to go back to 3 again:

(3) With these new constraints, the DM has to make a choice depending on the time available: either build a solution F_4 using constraints like the ones used to obtain the best solution so far, namely, F_2 , or restart from step 1 considering

the problem with this newly added constraints as a new problem. Assuming a decision must be taken in a short time, the DM decides for the former.

- (4) The new constraints are added to the LP. We then optimise cost constraining linear anti-flexibility to 40 and a customer satisfaction of 0.9 and obtain the solution F_4 .
- (5) The DM is satisfied with the attribute values and proposes F_4 as a solution.

Conclusions and Future Work

We developed an interactive multi-objective optimisation method, which is meant to be used as a decision support tool for planners. This tool provides the user the possibility to explore solutions as she/he seems fit and provides a range of different planning solutions for the planner to choose from, which are both properties sought for in a decision support tool.

The method above proposes a hypothetical scenario where a planner uses this tool for the purpose of making a plan. However, the tool itself is an optimising method and it could be used for other purposes; for example, given a particular problem, it can quantify the impact that certain attributes have on cost, such as the delayed delivery, or the minimum capacity on barges. This could illustrate how certain behaviours on the network are affecting the cost–performance of the network or how some advantages are not being exploited.

In order to understand the tool, the user needs familiarisation with the concepts used, such as space-time network, optimisation and a fair notion of the mathematics involved. Since the goal of this tool is to illustrate the benefits that can come from planners using multi-objective optimisation, it is very important to keep things simple. Therefore, the command inputs proposed in this method attempt to be used and understood (as much as possible) by a non-technical user. On the other hand, when compared to other interactive solution approaches in the literature, this method requires less input from the DM and also less technical expertise. As the future work, once the value of such a tool has been acknowledged by the planners and the planners are committed to the interactive use of the tool, the complexity and usage of the tool can be increased. If more advanced interactive methods are developed, there should always be sensitivity into the context, use and level of involvement of the DM.

Reference

1. Miettinen, K., Ruiz, F., & Wierzbicki, A. P. (2008) Introduction to multiobjective optimization: Interactive approaches. In *Multiobjective optimization* (pp. 27–57). Springer.

Chapter 8

Reduction of Variables for Solving Logistic Flow Problems



K. Kalicharan

Abstract In logistic problems, an Integral Multi-Commodity Network Design Problem on a time-space network is often used to model the problem of routing transportation means and assigning freight units to those means. In Physical Internet and Synchronodal networks, an interactive planning approach is preferable, meaning that calculation times of a single planning step should be short. In this chapter, we provide finding ways to reduce the number of variables in the problem formulation, which are effective at reducing the computation time for ILP-based solution methods.

Introduction

As indicated before, in synchronodal networks, a more interactive planning scheme is preferable, following the dynamic and uncertain nature of the underlying networks. Here routing decisions can (or have to) be made, each time logistic units arrive at an intermediate node, incorporating the decisions of other agents and, possibly, the uncertainty of decisions or events in the future. One way of offering the interactivity is by making a new planning every time new information is available, requesting a way to solve the problem quickly. Starting with the MCMCF problem as formulated in section “[Modelling the Problem as a MCMC Flow Problem on a Space–Time Network](#)”, we have to find ways to make this problem easier to solve, meaning relaxing constraints to obtain a simpler problem. This yields lower bounds that together with heuristically found upper bounds can be combined in, for instance, a branch-and-bound algorithm, see, among others, the paper by Crainic et al. [6] and Holmberg and Yuan [11]. Holmberg and Hellstrand [10] propose an exact solution method for the uncapacitated problem based on a Lagrangian heuristic. A dual ascent procedure is treated by Balakrishnan et al. [4], which finds lower bounds within 1–4% of optimality. Heuristics and meta-heuristics (such as *Tabu*

K. Kalicharan (✉)
TNO, The Hague, The Netherlands

Search, Simulated Annealing and Genetic Algorithms) are also widely used. See, for instance, the paper by Crainic et al. [7], who look at a path-based formulation of the same problem and solve it with tabu search. Other papers looking into these meta-heuristics are among others [3, 5, 17].

We start this chapter with modelling this problem as an Integral Multi-Commodity Network Design (MCND) problem. We will present novel reduction approaches to reduce the computation time when solving the problem. The MCND problem will be introduced in section “[Multi-Commodity Network Design Problem](#)”. In section “[Variable Reductions](#)”, the proposed reduction approaches are introduced, using the specific structure of the problem. Next, in section “[Results](#)”, computational results will be presented and we will end with conclusions in section “[Conclusion](#)”.

Multi-Commodity Network Design Problem

The main goal of this chapter is about efficiently and simultaneously routing transport means and scheduling transportation units. Without losing any generality, we will say vehicle if we mean a transportation mean and container if we mean the transportation unit. One of the models that could be used here is the Capacitated Fixed Charge Network Flow Problem from [8, 9, 15, 18]. We have a directed graph $G = (V, E)$ (or multi-graph) that contains all the nodes and arcs in the network. We have a set of commodities K that represent the bookings/orders. Every commodity $k \in K$ has, without loss of generality, one source node s_k and one sink node t_k . The parameters c_e , for $e \in E$, are the capacities of the arcs. The parameters $f_{e,k}$, for $e \in E, k \in K$, determine the per unit cost of commodity k on arc e . The parameters $d_{v,k} = d_k$ if $v = s_k, d_{v,k} = -d_k$ if $v = t_k$ and $d_{v,k} = 0$ otherwise, where d_k is the demand/size of commodity k . The variables $x_{e,k}$ depict the magnitude of the flow of commodity k on arc e . Finally, we define the design variables $y_e, \forall e \in E$, that are one if the service at link e is active and zero otherwise. In intermodal transport, these design variables are normally one if and only if the corresponding vehicle travels the corresponding link. Many possible arcs to travel are added for a vehicle, and after the optimisation process, it is decided which design variables are one, ergo, which routes the vehicles should travel. The graphs for these models are often time-space networks, but other graphs can also be used [19]. The optimisation problem now looks like

$$\min \sum_{k \in K} \sum_{e \in E} f_{e,k} x_{e,k} + \sum_{e \in E} g_e y_e \quad (8.1)$$

$$\text{s.t.} \quad \sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \quad \forall v \in V, \forall k \in K \quad (8.2)$$

$$\sum_{k \in K} x_{e,k} \leq c_e y_e \quad \forall e \in E \quad (8.3)$$

$$x_{e,k} \geq 0, y_e \in \{0, 1\} \quad \forall e \in E, \forall k \in K. \quad (8.4)$$

The first part of the objective function minimises the cost using the $f_{e,k}$. The second part is a link cost, if a vehicle travels a certain link $e \in E$, then a certain fixed cost g_e is added. The flow conservation constraints (8.2) make sure that the total amount of a commodity that enters the node also leaves the node, except for the sources and sinks. The capacity constraints (8.3) say that if an arc in the network is not travelled by a vehicle, then no commodity flow may be on that arc. If a vehicle does travel an arc, then the container flow on that arc can be at most the capacity of the edge.

This model is in some papers [5, 17, 20, 21] extended to include what are called design-balanced constraints:

$$\sum_{e \in \delta^+(v)} y_e - \sum_{e \in \delta^-(v)} y_e = 0 \quad \forall v \in V. \quad (8.5)$$

These constraints make sure that everywhere a vehicle arrives, it also leaves. This means that the routes for the vehicles are directed cycles. So the network we use should contain directed cycles. When working over a time-space network, an additional arc should be added from the sink of vehicle type w of set W to the source of vehicle type w to make sure that it is possible to have a directed cycle.

In [19], a similar continuous time ILP (Integer Linear Programming problem) is proposed. This model also has time variables and some more types of constraints. An extension of the design-balanced service network design problem is given in [14]. The model takes into account the usage of vehicles and the opening of corridors. In [1], another extension is derived, and in [13] a model that shares some resemblances is applied to freight car distribution in scheduled railways.

The problem we will propose is similar, though it contains a few key differences. First, we will consider two vehicle types: not flexible, high capacity, low-cost vehicles that need to be scheduled in advance (barges, train etc.) and flexible, low capacity, high-cost vehicles (trucks, transporters etc.). From here, we will refer to the first type as barge and to the second type as truck. Second, only the first part of the objective function of the service network design problem is used. Third, the flow conservation constraints for the vehicles are slightly different from the design-balanced constraints: in our model, the number of non-truck vehicles is pre-specified, and only for the non-truck vehicles, vehicle flow conservation constraints are added.

We call our problem the Integral Multi-Commodity Network Design (MCND) Problem. The model uses a time-space network, wherein the routes of the vehicles are not known in advance. The arcs in the time-space network are possible links that a vehicle can travel. For the trucks, we add an arc for every time stamp from every terminal to every terminal. Naturally, the travel time is taken into account.

We repeat this process for the first barge, second barge etc. Arcs corresponding to different vehicles (types) that run between the same time-space nodes are **not** merged. This way it is assured that all the links of all the possible routes they can take are included in the time-space network. The **Integral MCND problem** then is

$$\min \sum_k \sum_{e \in E} f_{e,k} x_{e,k} \quad (8.6)$$

$$\text{s.t. } \sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \quad \forall v \in V, \forall k \in K \quad (8.7)$$

$$\sum_{e \in \delta^+(v) \cap E_w} y_e - \sum_{e \in \delta^-(v) \cap E_w} y_e = b_{v,w} \quad \forall v \in V, \forall w \in W \setminus \{\text{truck}\} \quad (8.8)$$

$$\sum_k x_{e,k} \leq c_e y_e \quad \forall e \in E \setminus E_{\text{truck}} \quad (8.9)$$

$$\sum_k x_{e,k} \leq c_e \quad \forall e \in E_{\text{truck}} \quad (8.10)$$

$$x_{e,k} \geq 0, y_e \in \{0, 1\}, x_{e,k}, y_e \in \mathbb{Z} \quad \forall e \in E, \forall k \in K. \quad (8.11)$$

For all non-truck arcs e in the network, we have created a discrete design variable $y_e \in \mathbb{Z}_{\geq 0}$, determining if the arc is used (8.11). The y_e are binary variables. The container flows are modelled with the variables $x_{e,k}$ and still have to be integral and non-negative (8.11). We assume that trucks do not necessarily need to be used the whole day, whereas barges do have to be used the whole day. For the paths of the barges to make sense, we should add constraints that disallow the barge to teleport or travel multiple links at the same time. Flow conservation constraints for $w \in W \setminus \{\text{truck}\}$ (8.8) can do exactly this in the same way the flow conservation constraints (8.7) work for the commodities. In these constraints, we have $b_{v,w}$, which describe the time-space nodes that are the sink and source for a $w \in W \setminus \{\text{truck}\}$. In more detail, for such w , we have that $b_{v,w}$ equals b_w if v is the source node of w , $-b_w$ if v is the sink node of w and 0 otherwise, where b_w is the number of vehicles of type $w \in W \setminus \{\text{truck}\}$. This is normally 1 unless the vehicle reduction has been applied. The capacity of an arc is dependent on the number of vehicles that travel the arc (8.9). For the truck arcs, we have different capacity constraints (8.10). Normally, a vehicle has the same capacity the whole day. So we can then replace the c_e in the capacity constraints for the arcs by capacity constants for the vehicle, c_w . For the trucks, however, c_e is the maximum number of trucks that can be deployed on link $e \in E_{\text{truck}}$. We assume that a truck can carry exactly one container. Thus, the number of trucks that travel an arc $e \in E_{\text{truck}}$ is equal to the number of containers that are trucked on that arc $\sum_{k \in K} x_{e,k}$.

Variable Reductions

Reducing the number of variables of an ILP may reduce the computation time required to solve it. For that reason, we will look at several ways to remove variables from the MCND ILP, as proposed in the previous section. A simple approach is to arbitrarily remove arc variables. However, then we might remove arcs that would be used in an optimal solution of the original problem. So, in this section, we try to introduce variable reductions in a smart way that do not change the optimal solution value too much. Note that the used variables are indexed over the locations, time stamps, vehicles and commodities in the MCND problem. Reducing the size of those sets will reduce the number of variables. We will present the reduction in the next subsections, ordered using the set they reduce.

Commodity Reductions

Reduction A: Same Sink/Source In the model, we assume all the containers in one booking combined in one commodity. This way there are less variables than if all containers would be a separate commodity. It is however possible to reduce the number of commodities even more if the following condition holds. If multiple bookings have the same sink, then these bookings can be combined into one commodity [2]. This can be done also if they share the same source, see Fig. 8.1. In that figure, the values of $d_{v,k}$ are visualised for the sink and source nodes of the commodities. In the left figure, a commodity is a booking, and in the right figure a commodity is two bookings.

The problem with combining them if they have different sinks and sources is that a container of booking 0 can be transported to the sink of booking 1, if they are put in the same commodity.

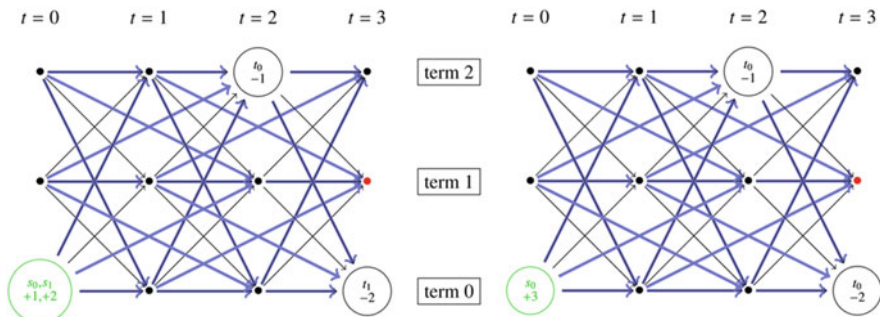


Fig. 8.1 Combined bookings shared source

If two bookings o_0 and o_1 have similar sinks, for example, if they have to be transported to the same destination location, then the same reduction is possible. If o_0 has to be there one time stamp earlier, then we can set the sink of o_1 to the same sink as that of o_0 . Note that by doing this the optimal solution may become worse. After this, we combine them in a single commodity.

Reduction B: Disjoint Time Frame Bookings In the ILP, arc variables are defined for a booking for every vehicle arc in the time-space network. Even those that start before the booking is released or end past its deadline. Suppose we have two commodities of which one has its deadline before the release time of the other one. Note that these bookings can be combined by putting them together in one commodity.

We can combine bookings in a greedy way: we start with the first one that is released and we add to the same commodity the first booking that is available after its deadline. We repeat this until it is no longer possible to add more bookings to the same commodity. After which we repeat this process for the next commodity. Clearly these bookings in the same commodity will not use the same arcs because there is no time stamp for which they are simultaneously available in the network. Every booking is available during a certain time frame.

Theorem *This greedy algorithm finds an optimal way to combine the bookings, that is, minimising the number of commodities, such that their time frames do not overlap.* \square

Proof For readability, we assume that none of the release times are equal, and however, the proof can be extended for cases where there are bookings with equal release times. Suppose we have an allocation that minimises the number of commodities by combining bookings in a certain way. In that allocation, we start with examining the first booking b_{i_1} that is released. If the next (in time) booking in the same commodity, b_{i_2} , is not the first one released after the first booking, b_{i_1} , then we swap the first booking released after it, b_{i_3} and everything in the same commodity as b_{i_3} later in time, with b_{i_2} and everything released after b_{i_2} on the same commodity as b_{i_2} . Then we repeat this process for b_{i_3} etc. until we are done for the commodity, and then we move to the first booking released that is not on a commodity that we already handled and do the same for that commodity, but we do not move the bookings that are on a commodity that is already ‘done’. The solution remains feasible and the number of commodities that are used does not increase. When we are done with all the commodities, we have found an allocation that is found by the greedy algorithm. \square

Vehicle Reductions

Reduction C: Same Vehicle Type In the MCND problem we have a set of vehicles $W = \{\text{truck, barge0, barge1, } \dots\}$. The trucks are already combined in the model,

and it is also possible to combine the barges in the model assuming they all have the same travel times and capacities. So then we get $W = \{\text{truck, barge}\}$. If there are barges of types A and B, we take $W = \{\text{truck, typeAbarge, typeBbarge}\}$. In the MCND model, the y_e variables are no binary variables anymore, but more general discrete variables. They keep track of the number of barges that take arc e . In the capacity constraints, these variables are multiplied with the capacities per barge to model the total barge capacity for a certain link.

If the barges are modelled individually, then for every barge a source and sink has to be given. With the reduction, it is possible to add multiple sources and sinks. So the barges still have the freedom to start from or end at different locations. Though, we can only specify the number of barges that have to arrive at a certain sink. It is not specified which individual barge has to arrive there, if there are multiple sinks. Furthermore, if too many sinks and sources are added, the number of possible paths for the barges might increase too much, also increasing the size of the feasible region.

An advantage of bundling different vehicles in the model together into a single vehicle type is that this is a way to avoid problems with symmetry and reduce the number of variables in the model. If the individual vehicles are modelled separately, many different solutions that are equivalent in practice have different variable allocations in the model. For instance, if there are ten identical barges that start at location 0 and one container needs to be moved from location 0 to location 1, then barge 0 can transport the container or barge 1 can transport it etc. As the barges are identical, these solutions are equivalent in practice. If these vehicles are put together in one index w in the model, then the container is not allocated to a specific barge in the model. That has to be done in a post-processing step.

Arc Reductions

Reduction D: Source/Sink Location In this reduction, we use the property that if some part of the route the commodity needs to be trucked, then it suffices to do that as soon as it is possible to do so. We also use the fact that it is always shorter to truck directly to a location than through another location.

In the MCND problem on a time-space network, some non-horizontal truck arc variables adjacent to a source location of a booking can be removed. It suffices to only add truck arc variables for a commodity from its origin location to every other location at its release time. The other non-horizontal truck arc variables adjacent to the source location can be removed. Similar things can be done for its sink location, though the arc from the source location to the sink location at the release time is never removed. Additionally, non-truck vehicle arc variables that go to the source location or leave from the sink location can be removed. In Fig. 8.2, we apply reduction D. The truck arcs are the thin arcs, and the barge arcs are the thick arcs. The truck arc variables that are removed for booking 0 are in red and the barge arc variables that are removed in blue.

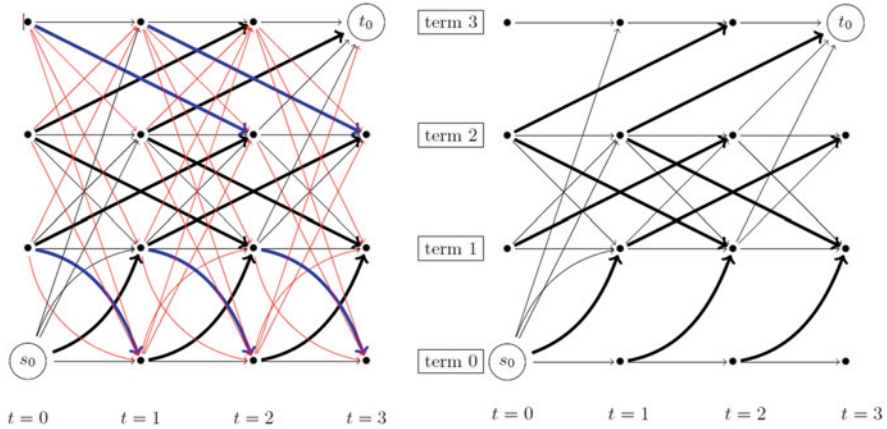


Fig. 8.2 Reduction D for MCND

Reduction E: Obsolete Barge Links Arcs in the time-space network that can never be travelled by some barge, because they start before the time the barge can be at the location, are removed. These links can never be taken by any container. Similarly barge arcs can be removed that depart too late at a location.

Location Reductions

Reduction F: Minimal Paths We call a $(loc1, loc2)$ path in a network minimal if the path has no sub-path that is a $(loc1, loc2)$ path. For every commodity k , we have that every path that is not a minimal (s_k, t_k) path in the space network can be removed. In our model, we can use that every location that is not on a minimal (s_k, t_k) path in the space network can be removed for commodity k . In Fig. 8.3, we see a space network with the waterway connections of several locations. Let $k \in K$ be a commodity with source location $\chi(s_k) = \text{Maasvlakte}$ and sink location $\chi(t_k) = \text{Hengelo}$, then we can conclude that arc variables that correspond to links that go to/from Delft do not have to be added for commodity k , if this reduction is applied.

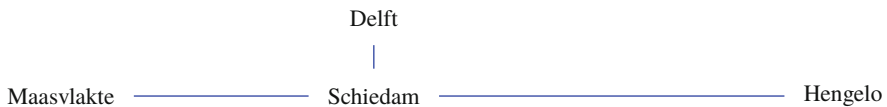


Fig. 8.3 Waterway connections

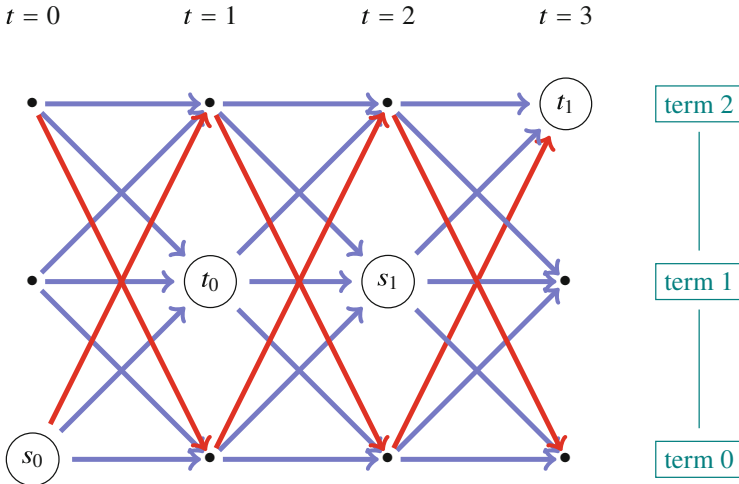


Fig. 8.4 Direct connection reduction

Reduction G: Direct Connection The network of the locations, waterways and roads can have a specific, recognisable structure. This structure can be used. If shipping from location $loc0$ to location $loc2$ means shipping through location $loc1$, then no arcs from $loc0$ to $loc2$ have to be added. It suffices to have arcs from $loc0$ to $loc1$ and from $loc1$ to $loc2$. See Fig. 8.4 for an example, there the red barge arcs are removed because of the structure of the waterways. We recommend taking a dense time grid with this reduction, because larger time steps may adversely affect the accuracy of the travel times between certain locations.

Reduction H: Locations in Between Every commodity k has an origin location $\chi(s_k)$ and a destination location $\chi(t_k)$. Let $d(\chi(s_k), \chi(t_k))$ be the Euclidean distance between $\chi(s_k)$ and $\chi(t_k)$, then we set all variables going to or from a location loc with $d(\chi(s_k), loc) > d(\chi(s_k), \chi(t_k)) + \delta$ and/or $d(\chi(t_k), loc) > d(\chi(s_k), \chi(t_k)) + \delta$ to zero. The δ should be chosen large enough to include locations that could be interesting for commodity k . Instead of the distance as the crow flies, it is also possible to use the trucking time for every pair of locations. This reduction can cut away optimal solutions. For example, in some problems, first trucking a container further away from your destination before shipping it to the destination would have given the optimal solution.

Time Reductions

Reduction I: Obsolete Time Reduction Let $v \in V$ be a time-space node, then we define $\tau(v)$ to be its *time* and $\chi(v)$ its *location*. Clearly a commodity can never

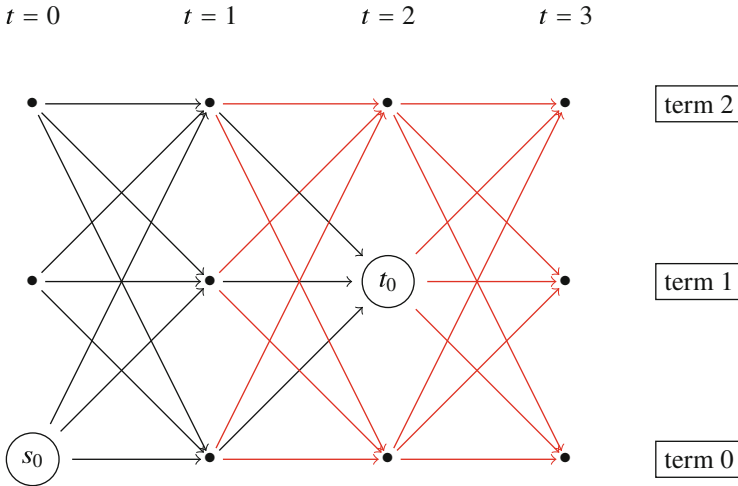


Fig. 8.5 Obsolete time reduction

take arcs that begin before its origin node time or end after its deadline. Instead of combining bookings in a commodity as in reduction B, one could remove those obsolete arcs entirely from the model. For every commodity $k \in K$, we remove all variables with $t + a(\text{loc}1, \text{loc}2, w)$ bigger than the time of its sink node $\tau(t_k)$, so $t + a(\text{loc}1, \text{loc}2, w) > \tau(t_k)$. We also remove all arcs with $t < \tau(s_k)$.

This reduction can even be enhanced by also removing variables with $t + a(\text{loc}1, \text{loc}2, w) = \tau(t_k)$ and $\text{loc}2 \neq \chi(t_k)$, where $a(\text{loc}1, \text{loc}2, w)$ is the travel time for vehicle (type) w from location $\text{loc}1$ to location $\text{loc}2$. Similarly, for the source, we obtain $t = \tau(s_k)$ and $\text{loc}2 \neq \chi(s_k)$.

In Fig. 8.5, we see an example of the truck arc variables; the basic reduction removes for commodity 0 in red.

Reduction J: Time Slot Reduction It is not always possible to go to a terminal. If the available time slots are known, then that information can be used to severely reduce the number of variables in the model. If an arc goes to a time-space node for which the location is not available at that time, then we instead let the arc go to the time-space node at that location that corresponds to the soonest time at which the location is available. If no such time exists, we remove the arc completely. If we get parallel arcs that correspond to the same vehicle type, then we merge them. We do a similar process for if the time-space node from which the arc leaves is not an available time-space pair.

Results

In this section, we present results of the reduction in calculation time by using some of the proposed network reductions on a test instance. The instance has eight terminals locations and two groups of six barges. All barges that belong to the same group have the same capacities, travel times and begin and end location. Therefore, all barges in the same group are modelled with one variable $w \in W$ in the model (Reduction C) unless stated otherwise. We assume an infinite number of trucks at every terminal, but restrictions may be added if that is desired. The travel times of the vehicles are based on data from practice and the truck travel times on data from Google Maps [16]. Every truck can carry exactly one (40 ft) container. We choose to look at a time span of 36 h with time steps of one hour, where 50 bookings and 100 containers are scheduled. We let the cost on the non-horizontal truck arcs be equal to the travel time of the arc. This models a company that owns barges and has to pay additional cost when trucking containers. We make sure that it is possible to truck a booking to its destination, so its deadline should be (at least one time stamp) later than its release date. Besides the terminal locations, there are two customer locations in the model. The customer locations are not reachable by barge. So in the time-space network, none of the barge arcs are incident to customer locations (which are not accessible by barge). Trains or other (types of) vehicles are initially not in the model but could easily be added. We solved our ILPs with IBM's CPLEX solver [12].

In all the experiments, reduction I is used, while it is not beneficial to add variables that cannot be non-zero in any feasible solution. The variables that are removed by reduction I and those that are removed by reduction E can never be used even if they are included in the model. The solver CPLEX removes many of those variables already automatically in the preprocessing phase. Therefore, these two reductions may influence the time for the preprocessing more than the time required for the actual solving. Though, an experiment in the past did show that reduction I also reduced the computation time of the solving phase. The other reductions will be turned on and off to see how they influence the computation time. In Table 8.1, the results are shown.

Reduction A is used to reduce the initial number of commodities $|K| = 50$ to $|K| = 39$. For this reduction, the computation time of the ILP with this reduction and without is compared, and we also split some commodity into sub-commodities to investigate the effect of taking a larger set $|K|$ on the computation time. Our set of vehicle types $W = \{\text{truck}, \text{typeAbarge}, \text{typeBbarge}\}$ is obtained by applying reduction C. If we only merge some of those barges of the same type or none of them, then we are using reduction C partially or not at all. In those cases, the set of vehicle (types) W is larger as we see in the table in the comparisons for reduction C. Reduction D is also implemented completely and partially to get some better insight in the effect of the reduction.

Table 8.1 Numerical results of reductions

Reduction	Active	Parameter	Comp. time	Solution
A	No	$K = 25$	7.12 s	2600 (opt.)
A	Yes	$K = 25 \rightarrow 20$	5.86 s	2600 (opt.)
A	No	$K = 50$	67.45 s	3760 (opt.)
A	Yes	$K = 50 \rightarrow 39$	61.16 s	3760 (opt.)
B	No		61.16 s	3760 (opt.)
B	Yes		43.35 s	3760 (opt.)
C	No	$ W = 6$	1667.61 s	3760 (opt.)
C	Yes	$ W = 5$	628.58 s	3760 (opt.)
C	Yes	$ W = 4$	183.51 s	3760 (opt.)
C	Yes	$ W = 3$	61.16 s	3760 (opt.)
D	No		117.61 s	3760 (opt.)
D	Yes	Sink incoming	61.16 s	3760 (opt.)
D	Yes	Sink in/out	64.58 s	3760 (opt.)
D	Yes	Complete	58.50 s	3760 (opt.)
F	No		129.98 s	3760 (opt.)
F	Yes		61.16 s	3760 (opt.)
G	No		>300 s	–
G	Yes		61.16 s	3760 (opt.)

Conclusion

Reduction A seems to help a little but clearly is most effective if there are many bookings with the same/similar sinks. The reduction may be less effective than expected, because adding multiple sources for a commodity may make the problem more difficult to solve. From theory, it is expected that reduction A is beneficial, if it is the number of commodities that blows up the computation time and many bookings have the same sink location. Reduction B surprisingly leads to better results, though it increases the number of variables because it was implemented together with reduction I. Reduction C is a very powerful tool. Reduction C does require a company to have many identical vehicles, though. Reduction D halves the computation time for our instance. Reduction F is effective, though we do need to do precalculations to use it. Reduction G is very effective. We do need however to know the structure of the waterways to use it.

Reduction H is expected to do well just like other location reductions. It is however possible that good solutions will be removed by using this reduction. Reduction J may be the most powerful reduction of them all. If the time windows at which locations can be visited are small, then many variables will disappear from the problem. The effectiveness of this reduction depends on the number of time stamps at which terminals are accessible. If there are few of those moments, then it will probably drastically reduce the computation time.

References

1. Andersen, J., Crainic, T. G., & Christiansen, M. (2009). Service network design with asset management: Formulations and comparative analyses. *Transportation Research Part C: Emerging Technologies*, 17(2), 197–207.
2. Babonneau, F., du Merle, O., & Vial, J. (2006). Solving large scale linear multicommodity flow problems with an active set strategy and proximal ACCPM. *Operations Research*, 54(1), 184–197 (2006). <https://doi.org/10.1287/opre.1050.0262>
3. Bai, R., Kendall, G., Qu, R., & Atkin, J. (2012). Tabu assisted guided local search approaches for freight service network design. *Information Sciences*, 189, 266–281.
4. Balakrishnan, A., Magnanti, T., & Wong, R. (1989). A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37(5), 716–740.
5. Chouman, M., & Crainic, T. (2012). *MIP-based matheuristic for service network design with design-balanced requirements*. CIRRELT.
6. Crainic, T., Frangioni, A., & Gendron, B. (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1), 73–99.
7. Crainic, T., Gendreau, M., & Farvolden, J. (2000). A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, 12(3), 223–236.
8. Ghamlouche, I., Crainic, T. G., & Gendreau, M. (2004). Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research*, 131(1–4), 109–133.
9. Hewitt, M., Nemhauser, G. L., & Savelsbergh, M. W. P. (2010). Combining exact and heuristic approaches for the capacitated fixed charge network flow problem. *Journal on Computing*, 22(2), 314–325.
10. Holmberg, K., & Hellstrand, J. (1998). Solving the uncapacitated network design problem by a Lagrangian heuristic and branch-and-bound. *Operational Research*, 46(2), 247–259.
11. Holmberg, K., & Yuan, D. (2000). A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operational Research*, 48(3), 461–481.
12. IBM. (2017). Cplex optimizer. <https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer>
13. Joborn, M., Crainic, T. G., Gendreau, M., Holmberg, K., & Lundgren, J. T. (2004). Economies of scale in empty freight car distribution in scheduled railways. *Transportation Science*, 38(2), 121–134.
14. Li, X., Wei, K., Aneja, Y., & Tian, P. (2017). Design-balanced capacitated multicommodity network design with heterogeneous assets. *Omega*, 67, 145–159.
15. Magnanti, T. L., & Wong, R. T. (1984). Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1), 1–55.
16. Page, L., & Pichai, S. (2018). Google maps. <https://www.google.nl/maps>
17. Pedersen, M., Crainic, T., & Madsen, O. (2009). Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science*, 43(2), 158–177.
18. Rodríguez-Martín, I., & Salazar-González, J. J. (2010). A local branching heuristic for the capacitated fixed-charge network design problem. *Computers and Operations Research*, 37(3), 575–581.
19. Sharypova, K. (2014). Optimization of hinterland intermodal container transportation. Ph.D. Thesis, Eindhoven University of Technology.
20. Vu, D. M., Crainic, T. G., & Toulouse, M. (2012). *A three-stage matheuristic for the capacitated multi-commodity fixed-cost network design with design-balance constraints*. CIRRELT.
21. Vu, D. M., Crainic, T. G., Toulouse, M., & Hewitt, M. (2014). Service network design with resource constraints. *Transportation Science*, 50(4), 1380–1393.

Chapter 9

Cutting Planes for Solving Logistic Flow Problems



K. Kalicharan

Abstract In logistic problems, an Integral Multi-Commodity Network Design Problem on a time-space network is often used to model the problem of routing transportation means and assigning freight units to those means. In Physical Internet and Synchromodal networks, an interactive planning approach is preferable, meaning that calculation times of a single planning step should be short. In this chapter, we provide ways to reduce the size of the problem formulation based on cutting planes, which are effective in reducing the computation time for Integer Linear Programming problem-based solution methods.

Introduction

As we indicated in the previous chapter, to allow for a more interactive use of this solution direction, short calculation times are crucial. Another way of doing this is to reduce the underlying problem using cutting planes techniques. We start again with the model formulated in section “[Multi-Commodity Network Design Problem](#)”, modelling this problem as an Integral Multi-Commodity Network Design (MCND) problem. We will present tailored cutting planes to reduce the computation time when solving the problem. In section “[Cutting Planes](#)”, the cutting planes are introduced. Next, in section “[Results and Conclusions](#)”, computational results and conclusions will be presented.

Cutting Planes

The feasible region of a Linear Programming (LP) problem may be unnecessarily large, which may lead to a large computation time. For instance, there may be many

K. Kalicharan (✉)
TNO, The Hague, The Netherlands

equivalent solutions satisfying the constraints of the LP problem. Furthermore, for ILP problems, in general, the feasible region defined without the integrality conditions contains many non-integral solutions. In practice, constraints are often added or changed to refine the feasible region making it easier to solve the problem. These constraints are called cutting planes or cuts. In this section, we define cuts as constraints that are added before or during the optimisation process, such that the optimal solution value of the problem does not change. An interesting heuristic for the integral MCND problem that uses cuts is discussed in [2].

General Cuts

For ILP problems, different general cuts exist. The cuts only remove non-integral solutions from the feasible region defined by the constraints of the problem (without integrality constraints). These general cuts are automatically added by solvers like CPLEX [4]. A well-known cut is the Gomory mixed integer cut (GMIC) for ILP problems [1, 6]:

$$\sum_{f_j \leq f} f_j x_j + \sum_{f_j > f} \frac{f(1-f_j)}{1-f} x_j \geq f, \quad f_j = a_j - [a_j].$$

A GMIC is often not only based on an inequality/row of the simplex tableau but can also be based on other valid inequalities. The cut is stronger than the fractional Gomory cut [1, 6]

$$\sum_{j=1}^n (a_j - [a_j]) x_j \geq a_0 - [a_0].$$

A zero-half cut [5] is basically adding two inequalities dividing by two and taking the floor. Example: suppose we have the inequalities $x_1 + 2x_2 \leq 3$ and $x_1 \leq 2$. Then we can combine them to $2x_1 + 2x_2 \leq 5$. We divide both sides by two $x_1 + x_2 \leq \frac{5}{2}$. Now, we take the floor at both sides and get $x_1 + x_2 \leq 2$.

Symmetry Breaking Cut

If we have barge 0 and barge 1 in the model, then they correspond to different variables in the model. Suppose barge 0 and barge 1 are identical and they have the same start node and end node. In that case, a solution with barge 0 taking path P_{j_1} and barge 1 taking path P_{j_2} is in practice equivalent with a solution in which barge 0 takes path P_{j_2} and barge 1 path P_{j_1} ceteris paribus. The symmetry leads to an unnecessary large feasible region and may result in a lengthy branch-

and-bound process. There are ways to deal with the symmetry. An option is to reformulate the problem. Applying reduction C is such an approach. Another option is to add symmetry breaking constraints to the model in advance. Adding too many cuts might be detrimental to the computation time of the algorithms, motivating our choice of only adding $|W| - 2$ path length cuts for the arc-based problems, where $W = \{\text{truck, barge0, barge1, barge2, } \dots \}$.

We use the fact that if we sum up all the y variables belonging to barge 0, we get the total path length of barge 0. The chosen numbering of the barges is then used to avoid some symmetry

$$\begin{aligned} \sum_{e \in E_{\text{barge0}}} y_e &\leq \sum_{e \in E_{\text{barge1}}} y_e \\ \sum_{e \in E_{\text{barge1}}} y_e &\leq \sum_{e \in E_{\text{barge2}}} y_e \\ &\vdots \end{aligned}$$

In these cuts, only the length of the paths is calculated, do if in the last example P_{j_1} and P_{j_2} would be paths of the same length, then the cuts will not block this symmetry. If the paths have different lengths, then the cuts will help out. Although normally optimal solution will be removed from the feasible region by adding these cuts, at least one optimal solution will always remain.

For the path-based problems, it is relatively easy to avoid symmetry. If $Q^{\text{barge0}} = Q^{\text{barge1}}$, then we number the paths of both barges the same way. Symmetric solutions can then be broken by adding constraints:

$$\sum_i i \zeta_i^{\text{barge0}} \leq \sum_i i \zeta_i^{\text{barge1}}.$$

Arc Residual Capacity Cut

In [7, 8], two cuts for multi-commodity problems are described that can be used for our MCND problem. One of them is the *arc residual capacity cut*:

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k - r'(\mu' - y_e), \tag{9.1}$$

with $\mu' = \lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil$ and $r' = \sum_{k \in K'} d_k - (\mu' - 1)c_e$ for $K' \subseteq K$ for $e \in E \setminus E_{\text{truck}}$. In the MCND model, constraints (8.9) hold, so for all $K' \subseteq K$ we have

$$\sum_{k \in K'} x_{e,k} \leq c_e y_e \quad \forall e \in E \setminus E_{\text{truck}}. \quad (9.2)$$

Note that a time-space graph has no cycles, together with constraints (8.7), (8.9) and (8.11), we know that

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k. \quad (9.3)$$

Theorem Let $e \in E \setminus E_{\text{truck}}$ and $K' \subseteq K$. Suppose y_e satisfies the integrality constraints (8.11) and constraints (9.2) and (9.3) are satisfied for e and K' , then (9.1) holds for e and K' . \square

Proof Let y_e be such that (8.11), (9.2) and (9.3) hold. If $y_e \geq \mu'$, then

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k \leq \sum_{k \in K'} d_k - r'(\mu' - y_e).$$

So, in this case, if (9.3) holds, then (9.1) holds.

If $y_e < \mu'$, then by (8.11) we have $y_e = \mu' - s$ where $s \geq 1$. It then follows that

$$\sum_{k \in K'} x_{e,k} \leq c_e y_e = c_e \mu' - c_e s \leq \sum_{k \in K'} d_k - r' s,$$

where the last inequality holds, because for $s = 1$ we have equality and

$$\begin{aligned} r' &= \sum_{k \in K'} d_k - (\mu' - 1)c_e = \sum_{k \in K'} d_k - \left(\lceil \sum_{k \in K'} \frac{d_k}{c_e} \rceil - 1 \right) c_e \\ &= \sum_{k \in K'} d_k - \lfloor \sum_{k \in K'} \frac{d_k}{c_e} \rfloor c_e = \sum_{k \in K'} \frac{d_k}{c_e} c_e - \lfloor \sum_{k \in K'} \frac{d_k}{c_e} \rfloor c_e \\ &= \left(\sum_{k \in K'} \frac{d_k}{c_e} - \lfloor \sum_{k \in K'} \frac{d_k}{c_e} \rfloor \right) c_e < c_e. \end{aligned}$$

\square

Theorem 9.1 We replace $\sum_{k \in K'} d_k$ by $\sum_{k \in K'} d_k + c_e$ in the arc residual capacity cut, and then we get

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k + c_e - r''(\mu'' - y_e), \quad (9.4)$$

with $\mu'' = \lceil \frac{\sum_{k \in K'} d_k + c_e}{c_e} \rceil$ and $r'' = \sum_{k \in K'} d_k + c_e - (\mu'' - 1)c_e$ for $K' \subseteq K$. Then, (9.4) is strictly dominated by (9.1) if $\frac{\sum_{k \in K'} d_k}{c_e} \notin \mathbb{Z}$. Furthermore, (9.4) is equivalent to (9.1) if $\frac{\sum_{k \in K'} d_k}{c_e} \in \mathbb{Z}$.

Proof We have

$$\mu'' = \lceil \frac{\sum_{k \in K'} d_k + c_e}{c_e} \rceil = \lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil + 1 = \mu' + 1$$

and

$$r'' = \sum_{k \in K'} d_k + c_e - (\mu'' - 1)c_e = \sum_{k \in K'} d_k - (\mu'' - 2)c_e = \sum_{k \in K'} d_k - (\mu' - 1)c_e = r'.$$

Then, (9.4) is equivalent to

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k + c_e - r'(\mu' + 1 - y_e) = \sum_{k \in K'} d_k - r'(\mu' - y_e) + c_e - r'.$$

In the case $\frac{\sum_{k \in K'} d_k}{c_e} \notin \mathbb{Z}$, we find

$$\begin{aligned} r' &= \sum_{k \in K'} d_k - (\mu' - 1)c_e = \sum_{k \in K'} d_k - \left(\lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil - 1 \right) c_e \\ &< \sum_{k \in K'} d_k - \left(\frac{\sum_{k \in K'} d_k}{c_e} - 1 \right) c_e = \sum_{k \in K'} d_k - \left(\frac{\sum_{k \in K'} d_k}{c_e} \right) c_e + c_e = c_e. \end{aligned}$$

So, with the above, we conclude that the arc residual cut (9.1) strictly dominates (9.4)

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k - r'(\mu' - y_e) < \sum_{k \in K'} d_k - r'(\mu' - y_e) + c_e - r'.$$

If $\frac{\sum_{k \in K'} d_k}{c_e} \in \mathbb{Z}$, then we get

$$\begin{aligned} r' &= \sum_{k \in K'} d_k - (\mu' - 1)c_e = \sum_{k \in K'} d_k - \left(\lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil - 1 \right) c_e \\ &= \sum_{k \in K'} d_k - \left(\frac{\sum_{k \in K'} d_k}{c_e} - 1 \right) c_e = \sum_{k \in K'} d_k - \left(\frac{\sum_{k \in K'} d_k}{c_e} \right) c_e + c_e = c_e. \end{aligned}$$

So in that case we conclude that the arc residual cut (9.1) is equivalent to (9.4)

$$\sum_{k \in K'} x_{e,k} \leq \sum_{k \in K'} d_k - r'(\mu' - y_e) = \sum_{k \in K'} d_k - r'(\mu' - y_e) + c_e - r'.$$

□

Corollary 9.1 *The right-hand side of the arc residual capacity cut (9.1),*

$$\sum_{k \in K'} d_k - \left(\sum_{k \in K'} d_k - \left(\lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil - 1 \right) c_e \right) \left(\lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil - y_e \right),$$

is not (monotonically) increasing in $\sum_{k \in K'} d_k$.

□

Proof We first write the right-hand side as a function of $\sum_{k \in K'} d_k$

$$g : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}$$

$$g\left(\sum_{k \in K'} d_k\right) = \sum_{k \in K'} d_k - \left(\sum_{k \in K'} d_k - \left(\lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil - 1 \right) c_e \right) \left(\lceil \frac{\sum_{k \in K'} d_k}{c_e} \rceil - y_e \right).$$

Let $A \in \mathbb{Z}_{\geq 0}$ such that $\frac{A}{c_e} \in \mathbb{Z}$ and define $B := A - s$ for a $s \in \mathbb{Z} \cap [1, c_e - 1]$. We have $A - c_e < B < A < B + c_e < A + c_e$. Suppose g is monotonically increasing, then we would have

$$g(A - c_e) \geq g(B) \leq g(A) \leq g(B + c_e) \leq g(A + c_e). \quad (9.5)$$

From Theorem 9.1 follows

$$g(A - c_e) = g(A) = g(A + c_e) \quad (9.6)$$

$$g(B) < g(B + c_e). \quad (9.7)$$

From (9.5) and (9.6) follows

$$g(A - c_e) = g(B) = g(A) = g(B + c_e) = g(A + c_e),$$

but this contradicts with (9.7). So we conclude the right-hand side of the arc residual capacity cut is not monotonically increasing in $\sum_{k \in K'} d_k$. □

The structure of the time-space graph can be used to reduce the right-hand side of the arc residual capacity cut (9.1). Although this does not always help to find a stronger cut (Corollary 9.1), it still finds stronger cuts very often (Theorem 9.1). For an arc $e \in E$, no commodities should be included in the $K' \subseteq K$ for the arc residual capacity cut, which consist of bookings that cannot use that link. If we exclude these commodities, then the left-hand side does not change as $x_{e,k} = 0$ if commodity k

cannot use arc e . The right-hand side is likely to decrease (Theorem 9.1), so if this is the case, the cut will become stronger.

Cutset Cut

For every truck arc $e \in E_{\text{truck}}$, we add the variable $y_e \in \mathbb{Z}_{\geq 0}$ and the constraint

$$\sum_{k \in K} x_{e,k} = y_e \quad \forall e \in E_{\text{truck}}. \tag{9.8}$$

The second cut suitable for the MCND problem described in [7, 8] is the cutset cut. Let $k \in K$ be a commodity that consists of one booking. Then, it has a source node s_k . For the node, we can add the cutset cut

$$\sum_{e \in \delta^+(s_k)} y_e \geq 1. \tag{9.9}$$

In Fig. 9.1, an example is visualised, where only half a barge is deployed to transport the container of a commodity at the start. In scenarios like that, the cutset cut can be used to avoid the fractional solution that is found for the LP problem relaxation.

Constraints (8.9) imply

$$x_{e,k} \leq \sum_{k_1 \in K} x_{e,k_1} \leq y_e c_e \Rightarrow \sum_{e \in \delta^+(s_k)} y_e c_e \geq \sum_{e \in \delta^+(s_k)} x_{e,k} \quad \forall e \in E, \forall k \in K.$$

Constraints (8.7) and (8.11) imply

$$\sum_{e \in \delta^+(s_k)} x_{e,k} = d_k \quad \forall e \in E, \forall k \in K.$$

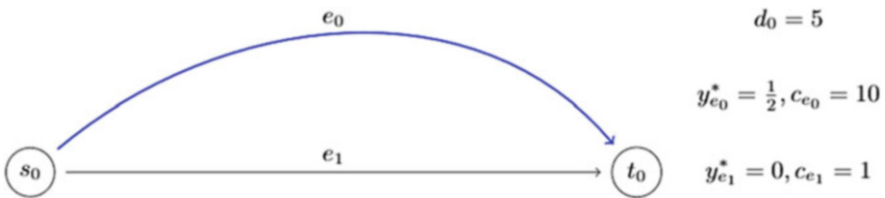


Fig. 9.1 Cutset cut example and LP problem relaxation solution y^*

Substituting that and using the maximum arc capacity $c_{\max} := \max_{e \in E} c_e$, we get

$$c_{\max} \sum_{e \in \delta^+(s_k)} y_e \geq \sum_{e \in \delta^+(s_k)} y_e c_e \geq d_k \Leftrightarrow \sum_{e \in \delta^+(s_k)} y_e \geq \frac{d_k}{c_{\max}} \quad \forall k \in K.$$

Constraint (8.11) implies $y_e \in \mathbb{Z}_{\geq 0} \forall e \in E$. We can use this to get the enhanced cutset cut

$$\sum_{e \in \delta^+(s_k)} y_e \geq \lceil \frac{d_k}{c_{\max}} \rceil \quad \forall k \in K. \quad (9.10)$$

The demand d_k and the maximum capacity c_{\max} are always integral, so (9.10) is stronger than (9.9). This cut can be generalised.

Let $G = (V, E)$ be a directed graph. Suppose we have a subset of the nodes $S \subseteq V$. We denote the complement of S as $\bar{S} := V \setminus S$. The partition of the nodes $C := (S, \bar{S})$ is called a *cut*, to avoid ambiguity we will call this a *graph cut* in this chapter. Then, $\delta(S, \bar{S}) = \{(i, j) \in E \mid i \in S, j \in \bar{S}\}$ is called the *cutset* of the graph cut (S, \bar{S}) . The cuts in this section are named after it.

This allows us to define the general cutset cut

$$\sum_{e \in \delta(S, \bar{S})} y_e \geq \lceil \frac{\sum_{k \mid s_k \in S, t_k \notin S} d_k}{c_{\max}} \rceil \quad \forall S \subseteq V. \quad (9.11)$$

If we take $S = \{s_{k_1}\}$ for a $k_1 \in K$, then we get an enhanced cutset cut back (9.10) (if there is no $k_2 \in K$ with $k_2 \neq k_1$ and $s_{k_1} = s_{k_2}$). If there are multiple commodities with the same source, then sufficient capacity needs to be installed to handle the sum of their demands. So then (9.11) is stronger than (9.10). Suppose we have a graph cut (S, \bar{S}) , then one could even say that sufficient capacity needs to be installed on the arcs of $\delta(S, \bar{S})$ to be able to transport at least the sum of the demands of the commodities that have their source in S and sink in \bar{S} . Every container of those commodities has to be transported from a node in S to a node in \bar{S} . Consequently, every such container is transported by at least one arc in $\delta(S, \bar{S})$. This reasoning gives some intuition why (9.10) can be generalised to (9.11). If the design variables y_e are binary variables, then we will use the flow cover cuts

$$\sum_{e \in Q} y_e \geq 1 \quad \forall Q \subseteq \delta(S, \bar{S}) \text{ with } \sum_{e \in Q} c_e < \sum_{k \mid s_k \in S, t_k \notin S} d_k, \forall S \subseteq V. \quad (9.12)$$

There are a huge number of node subsets $S \subseteq V$. So adding all cuts (9.11) is in general impractical. We can add some general cutset cuts during the branch-and-cut process. Finding violated cuts is called the *separation problem*. An $s_k - t_k$ cut for a $k \in K$ is a graph cut $C = (S, \bar{S})$ with $s_{k_1} \in S$ and $t_{k_1} \in \bar{S}$. Suppose we have a non-integral solution (x^*, y^*) found in one of the nodes of the branch-and-cut tree. To find violated cuts, we take the following steps:

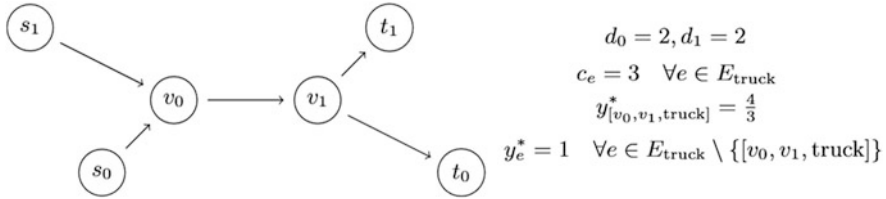


Fig. 9.2 Undiscovered violated cutset cut example

1. For all commodities $k_1 \in K$, repeat the following steps.
2. Set $S := \{s_{k_1}\}$ and $T := \{t_{k_1}\}$.
3. Put the values y_e^* as capacity on the arcs of the time-space graph.
4. Find a minimal $s_{k_1} - t_{k_1}$ cut by solving the max flow problem[3] on the time-space graph with s_{k_1} the source and t_{k_1} the sink (according to the max flow—min cut theorem[3]).
5. Check for the S constructed if the corresponding cut (9.11) is violated, if so add it to the ILP problem.

This way at a node in the branch & cut tree at most $|K|$ max flow problems need to be solved. The max flow problem is similar to the min cost flow problem.

Although the method can discover many violated cutset cuts, there are also some it cannot discover. The step-by-step plan above gives two minimum graph cuts in the graph of Fig. 9.2 with five truck arcs: the graph cut with $S = \{s_0\}$ and the graph cut with $S = \{s_1\}$. The corresponding general cutset cuts are not violated, because $y_{[s_1, v_0, \text{truck}]} \geq 1$ and $y_{[s_0, v_0, \text{truck}]} \geq 1$ hold (with equality). The general cutset cut with $S = \{s_0, s_1, v_0\}$, $y_{[v_0, v_1, \text{truck}]} \geq 2$ however, is violated.

Repeatedly solving max flow problems might take too much time in practice. So better options may be to add cuts (9.10) to the model. This can be done in advance or if they are violated during the branch and cut. Other useful cuts may be (9.11) for $S := \{s_k, (\tau(s_k) + 1, \chi(s_k)), \dots, (\tau(s_k) + r, \chi(s_k))\}$ for all $k \in K$ for some small $r \in \mathbb{Z}_{\geq 0}$. If S and \bar{S} are both large sets, then the cut is normally not effective, because in that case there is a high chance that too many vehicles go from nodes in S to nodes in \bar{S} rendering the corresponding cut (9.11) useless.

Strong Cut

Strong cuts from [2] are defined as

$$x_{e,k} \leq d_k y_e \quad \forall e \in E, \forall k \in K. \tag{9.13}$$

These are only useful if all containers of a commodity are going through an arc $e \in E \setminus E_{\text{truck}}$. Then the corresponding strong cut ensures that at least one vehicle

should be used for that link. These cuts are less effective, when used in combination with vehicle and commodity reductions. With Reduction C, the design variables are non-binary making the cuts weaker. In case commodity reductions are used and/or commodities have high demands, then the chance is smaller that all containers of the commodity take the same arc. So then the cut is also less effective.

We have general integral design variables when vehicle reductions are used. In that case, we can make more effective cuts. If $x_{e,k} = d_k$, then we want to have that $y_e \geq \lceil \frac{d_k}{c_e} \rceil$. This means that we need to have $\lceil \frac{d_k}{c_e} \rceil d_k$ for the LHS (left-hand side). Furthermore, if $x_{e,k} = \lfloor \frac{d_k}{c_e} \rfloor c_e$, then we want to have $y_e \geq \lfloor \frac{d_k}{c_e} \rfloor$, so that means $\lfloor \frac{d_k}{c_e} \rfloor d_k$ for the LHS. Suppose $\frac{d_k}{c_e} \notin \mathbb{Z}$. Then we can define a linear function for the LHS that goes through the two points described:

$$ax_{e,k} + b \leq d_k y_e \quad \forall e \in E, \forall k \in K, \quad (9.14)$$

with $a = \frac{\lceil \frac{d_k}{c_e} \rceil d_k - \lfloor \frac{d_k}{c_e} \rfloor d_k}{d_k - \lfloor \frac{d_k}{c_e} \rfloor c_e} = \frac{d_k}{d_k - \lfloor \frac{d_k}{c_e} \rfloor c_e}$ and $b = \lceil \frac{d_k}{c_e} \rceil d_k - ad_k$. We divide by d_k to get

$$ax_{e,k} + b \leq y_e \quad \forall e \in E, \forall k \in K, \quad (9.15)$$

with $a = \frac{1}{d_k - \lfloor \frac{d_k}{c_e} \rfloor c_e}$ and $b = \lceil \frac{d_k}{c_e} \rceil - a$. For the substitution, we use $\mu' = \lceil \frac{d_k}{c_e} \rceil$ and $r' = d_k - (\mu' - 1)c_e$ and we simplify to get

$$x_{e,k} \leq d_k - r'(\mu' - y_e) \quad \forall e \in E, \forall k \in K. \quad (9.16)$$

These are arc residual capacity cuts. This has a few implications. We do not have to show that this enhanced cut does not cut away integral solutions, because that is already shown for arc residual capacity cuts. It is not necessary to add strong cuts, because if they are effective, $x_{e,k} = d_k$, then the arc residual capacity cut forces the same bound for y_e and the arc residual capacity cut can be even stronger for integral design variables. These arc residual capacity cuts seem to be useful if $x_{e,k} \in \{d_k, \lfloor \frac{d_k}{c_e} \rfloor c_e\}$, but for most other values of $x_{e,k}$ it gives a non-integral lower bound for y_e . For the interested reader, we refer to [2] for another cut, the flow pack cut.

Results and Conclusions

In this section, we present the first results of the cutting planes in calculation time on a simple test instance. The instance has eight terminal locations and two groups of six barges. All barges that belong to the same group have the same capacities, travel times and begin and end location. We assume an infinite number of trucks at every terminal, but restrictions may be added if that is desired. The travel times

Table 9.1 Numerical results of cuts

Cut	Active	Parameter	Comp. time	Solution
Symmetry breaking cut	Yes	$ W = 4$	292.32 s	3760 (optimal)
Symmetry breaking cut	No	$ W = 4$	259.27 s	3760 (optimal)
Symmetry breaking cut	Yes	$ W = 5$	641.11 s	3760 (optimal)
Symmetry breaking cut	No	$ W = 5$	665.19 s	3760 (optimal)
Strong cut	Yes		101.22 s	3760 (optimal)
Arc residual capacity cut	Yes	All commodities	>300 s	3850
Arc residual capacity cut	Yes	Per commodity	54.58 s	3760 (optimal)
Cutset cut	Yes	Counter = 0	61.16 s	3760 (optimal)
Cutset cut	Yes	Counter = 1	58.42s	3760 (optimal)
Cutset cut	Yes	Counter = 2	47.83 s	3760 (optimal)
Cutset cut	Yes	Counter = 3	100.97 s	3760 (optimal)

of the vehicles are based on data from practice and the truck travel times on data from Google Maps. Every truck can carry exactly one (40 ft) container. We choose to look at a time span of 36 h with time steps of one hour, where 50 bookings and 100 containers are scheduled. We let the cost on the non-horizontal truck arcs be equal to the travel time of the arc. This models a company that owns barges and has to pay additional cost when trucking containers. We make sure that it is possible to truck a booking to its destination, so its deadline should be (at least one time stamp) later than its release date. Besides the terminal locations, there are two customer locations in the model. The customer locations are not reachable by barge. So in the time-space network none of the barge arcs are incident to customer locations (which are not accessible by barge). Trains or other (types of) vehicles are initially not in the model but could easily be added. We solved our ILP problems with IBM's CPLEX solver.

For the cutset cut, the counters determine the number of cutset cuts we take and which cutset cuts. If the counter is zero, we only take the cutset based on the sink of the commodity. If the counter is one, we also take the cutset cut with the sink and the time-space node with the same location as the sink one time stamp earlier etc.

The results are depicted in Table 9.1. The symmetry breaking cut does not show its effectiveness in this example. We expect that the instance is too small for this cut to be effective. The strong cut and the arc residual capacity cut are put together in one part of the table, because the arc residual capacity cut is a stronger version of the strong cut. Using the strong cut brings the calculation time back from more than 300–101.22 s, still finding the optimal solution. Using the arc residual capacity cut per commodity is very effective. This may be because many commodities send all of their containers over one path. One could experiment with other sets of commodities to reach even better results. Using the cut for all commodities may lead to the case, with high probability, that this includes commodities that do not use the specific arc, making, probably, this cut weaker. Using this cut for more than two commodities is only recommended if you know that the arc is used by many

commodities and preferably which commodities. The cutset cut only helps a little here. More experiments with different types of cutsets and other instances should be done to see the full impact of this cut.

For further research, we recommend testing the cuts on a variety of test instances to understand the best fit of the cuts on specific cases. This would also show whether the symmetry breaking cut does indeed perform on bigger instances. Also, the effective use of (which) cutset cuts should follow from this analysis.

References

1. Aardal, K., Weismantel, R., & Wolsey, L. A. (2002). Non-standard approaches to integer programming. *Discrete Applied Mathematics*, 123(1–3), 5–74.
2. Chouman, M., & Crainic, T. (2012). *MIP-based matheuristic for service network design with design-balanced requirements*. CIRRELT.
3. Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., & Schrijver, A. (1998). *Combinatorial optimization*. Wiley Interscience.
4. IBM. (2017). Cplex optimizer. <https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer>
5. IBM. (2017). Zero-half cuts. www.ibm.com
6. Letchford, A. N., & Lodi, A. (2002). Strengthening Chvátal–Gomory cuts and Gomory fractional cuts. *Operations Research Letters*, 30(2), 74–82.
7. Magnanti, T. L., Mirchandani, P., & Vachani, R. (1995). Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1), 142–157.
8. Marchand, H., Martin, A., Weismantel, R., & Wolsey, L. (2002). Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1–3), 397–446.

Part III

Synchromodal Logistics as Selfish Systems

In the previous part, the synchromodal planning problem was assumed to have global information and search for a global optimisation, giving a “social” problem following the definition of Fig. 1.1. However, in practice, multiple parties act independently from each other, each trying to get the best solution from their viewpoint. These parties can be various, smaller or modality specific, LSPs or even individual containers. Where the solution of the “social” case is at least as good (or better than) the resulting “selfish” solution, it is interesting to look how the “selfish” system can be motivated to find and use the global optimum and how the benefit, and the individual losses, can be divided fairly among all parties. This will be the topic of the chapters in this part.

In Chap. 10, the focus is on how players in a “selfish” system will cooperate, under different levels of shared information. Next, Chaps. 11 and 12 will focus on how network pricing can be used to force the “selfish” system to find the “social” solution and how the profits and losses are distributed in a fair way, using toll schemes.

Chapter 10

Optimising Routing in an Agent-Centric Sychromodal Network with Shared Information



M. A. M. De Juncker

Abstract This chapter focuses on sychromodal planning problems in which information is shared between all agents in the system and they choose their routes based on an individual optimisation objective. We show the effect of the information availability by developing three different methods to determine the optimal paths, to motivate logistic players to cooperate in a sychromodal system.

Introduction

Freight transportation is growing and so is the need for an efficient organisation of hinterland transport services. The main element of a sychromodal transportation [19] is the integration of transport service on different modalities with real-time availability of information. Changes have to be made to the network in order to create a sychromodal system. Among others, there is need for an integrated network and service design, an integrated operation and control, contracts that allow synchronised transport, a stronger collaboration and a mind shift in planning and control.

In Chap. 1, sychromodal planning problems are classified in two directions: available information and the degree of control and optimisation. Both can take either a local view, where only own information is known and optimisation is for an individual objective, or a global view, where information is available for the entire network and the optimisation is aimed at a shared goal. If the information is available globally but every agent only optimises their own objective, the approach is called *selfish*. Information is a broad term, some of the information is public, which means every agent can get this information. Other information is private and has to be shared between different stakeholders in the network. This sharing can be difficult to achieve, since the stakeholders need to be willing to share their private information to competitors and clients. In this chapter, we focus on agent-centric

M. A. M. De Juncker (✉)
TNO, The Hague, The Netherlands

sychromodal networks, where each agent is selfish and wants to optimise its own objective function. The public information we encounter is information about the occupancy of different links in the network. This means that we know, at any point in time, how many agents are on every link of the network. One can already see examples of this public information being used for road networks. For example: route guiding systems already have an option to recalculate your shortest path based on information about current congestion in the network. Private information, on the other hand, does require different stakeholders to cooperate. The private information we encounter is information about upcoming orders. Logistic service providers normally know what orders are going to arrive in the near future. If these logistic service providers would be willing to share this information with all stakeholders, all stakeholders can react upon this information. This means that agents also have information about the, probable, future occupancy of certain links. Note that much research is known on agent based road traffic, looking for a user equilibrium. In these networks centrally controlled optimisation is not possible. In logistics we assume both approaches are possible and therefore worthwhile investing.

This chapter investigates the effect of different information availability in these kind of networks to show the logistic players the value of cooperating. We assume cases where only public information is available and cases where agents have access to both public and private information. We present three models for the analysis, using different methods to generate paths through the network for the agents:

- *Model 1*: Naive implementation with public information. We assume that each agent checks the public information at departure and will react accordingly. This means they will take the shortest path for the current state of the network. After the choice is made, they will not deviate.
- *Model 2*: Very similar to the first, except the fact that agents *do* switch routes before reaching their destination. This means that at each decision point, i.e., an intermediate node in the network, they again check the state of the network and reroute if necessary.
- *Model 3*: Assumes full information. Each agent knows the future destinations and routes of other agents for a certain planning horizon. This includes public as well as private information. With this information, the algorithm seeks to find the optimal routes for all agents that arrive somewhere in the planning horizon.

In section “[Literature Review](#)” we give an overview of the literature on Dynamic Traffic Assignment models and on the effect of information in road networks. In section “[Models](#)” we describe the used models for the analysis and the underlying simulation techniques. Next, in section “[Results](#)” we discuss the results we found. In the final section, section “[Conclusions](#)”, we derive the conclusions of our work and mention recommended further research.

Literature Review

In Dynamic Traffic Assignment (DTA) problems one can take one of two approaches in choosing objectives:

1. *System optimal (SO)*: in this case one wants to optimise a system objective; e.g., congestion or average travel time. This also means that all vehicles are controlled by a central controller.
2. *User equilibrium (UE)*: here every vehicle in the system wants to optimise an individual performance measure; e.g., travel time or costs.

There exist models for these DTA problems, but there is no model that provides a universal solution for general networks [15]. In DTA models, there is a trade-off between traffic realism and the theoretical guarantee of properties such as existence, uniqueness and stability.

Peeta and Ziliaskopoulos [15] give an overview of the different modelling approaches to DTA problems. These approaches can be divided into four categories: *mathematical programming*, *optimal control*, *variational inequality* and *simulation-based*. We mention interesting literature on all four approaches. We also describe the advantages and disadvantages of using certain approaches.

Mathematical programming DTA models aim to formulate the problem as a mathematical discrete-time program. The first formulation was by Merchant and Nemhauser [12]. Birge and Ho [4] extended this model to the stochastic case by allowing for random demand desires. Jansen [9] describes the UE DTA problem as a mathematical program. However, all formulations are non-convex because of the FIFO requirement. While there is enough literature on non-convex optimisation, in a DTA context analytical and computational tractability are lost for general networks. Together with the difficulty to prohibit holding back of traffic, mathematical programming formulations lack efficient solutions for realistic instances. Carey and Subrahmanian [5] illustrate some of the issues that arise because of the FIFO requirement and the holding back of traffic in mathematical programming formulations. An overview on dynamic dispatching problems with stochastic requests can be found in [20].

Optimal control theory DTA formulations are continuous time problems. Here the origin-destination rates are assumed to be known continuous functions of time. For the formulations we refer to [8, 18] and [17]. The main issue with optimal control formulations is that there is no efficient solution algorithm. As mentioned before, these formulations also have no explicit constraints for the FIFO requirement and the holding back of traffic. Therefore, new research has been done in variational inequality formulations.

Variational inequality formulations were introduced by Dafermos [6]. Variational inequality formulations are used in equilibrium problems. One defines an inequality involving a functional, which has to be solved for all possible values of a variable. Nagurney [13] provides a summary of variational inequality formulations and addresses various equilibrium problems in network economics, under which the

traffic network equilibrium. Variational inequality formulations can handle more realistic traffic scenarios, but the approaches are computationally intensive. Also, the problems with the FIFO requirement and the holding back of traffic remain.

Simulation-based DTA models use a traffic simulator in order to handle realistic traffic scenarios. The main issue with simulation-based models is that theoretical insights cannot be gathered from the models. The solution methods in these simulation-based models often use the traffic simulator as part of the solution. This is called the *predictive-iterative method*, where the simulator is used in each iteration to predict future traffic conditions given a certain route assignment. Based on these predictions a new route assignment is determined and so on. One of these iterative models is described by Peeta and Mahmassani [14]. A similar iterative approach for a UE DTA is taken by Kaufman, Smith and Wunderlich [10]. These models are much more realistic than the analytic ones and, therefore, widely used in analyses. However, deployment in real life is only feasible if the algorithms are computational efficient. Ben-Akiva et al. propose DynaMIT in [3] (and its route guidance in [2]), which uses a demand and supply simulator to generate UE route guidance under a rolling horizon framework.

Next to DTA models there is also literature available on the effect of information in road networks, see, for example, the papers by Mahmassani and Jayakrishnan [11] and Dia [7], both describes a modelling framework to analyse the effect of in-vehicle real-time information. The framework consists of a *simulation component* and a *user decisions component*.

Models

In this section we describe the models used for the analysis. Firstly, in section “[Assumptions](#)”, we state the assumptions for all models. Then, section “[Description of Simulation](#)” describes the simulation we developed for the agents moving through a network. How the routing is done in the simulation is decided using one of the three models. Section “[Public Information Models](#)” describes models 1 and 2. These heuristics assume that the agents in the network only know the occupancy of the links in the network up to the current time. Section “[Full Information Model](#)” describes model 3. This model relies on the assumption that there is perfect knowledge. This means that all agents also know how many agents are due to arrive in the future.

Assumptions

First, we assume to have a transportation network, where nodes are locations and links are connections between locations which can be various modalities, such as trucks (roads), trains and barges. In the remainder of the chapter we assume by

agents containers in the network that have to be transported from a certain origin to a certain destination using one or several modalities. Next, in the models we assume certain properties as stated below.

Assumption 10.1 All nodes can be reached by truck.

Assumption 10.2 The costs of travelling links are non-negative.

Assumption 10.3 All information is available to all agents.

For the first two models, Assumption 10.3 means that the occupancy on all links is known up to the departure time of the agent. For model 3, we assume full information. Therefore, Assumption 10.3 means that the occupancy on all links is known as well as future orders, i.e., containers that want to travel from an origin to a destination within the network, and their routes.

In section “[Description of Simulation](#)” we mention how we calculate time- and state-dependent travel times for our network. There are also some specific assumptions on the travel times.

Assumption 10.4 The travel times of roads only depend on the occupancy of the link.

Assumption 10.5 The departure times of the trains and barges are known.

Assumption 10.6 The capacities of all trains and barges are known.

Assumption 10.7 The travel times of trains and barges over a certain link are constant.

Description of Simulation

The simulation used in all models is an event-driven simulation, where each new event triggers a change in the network. Possible events are:

- Request route: a new agent asks for a route from an origin to a destination.
- Enter link: an agent will traverse a certain link in the network.
- Leaving link: when an agent reaches its destination or an intermediate node, he leaves the link.

The simulation handles the events one by one until a certain end time. The entire duration of the simulation is referred to as the planning horizon. In this simulation one can keep track of all kinds of performance measures: individual travel times, average travel times, occupancy on roads, etc.

The travel times in the simulation are time- and state-dependent. However, we assume that we know how many agents occupy certain links and at what time they want to traverse the link. This means that the time and state is fixed for the calculation of the travel times.

The travel times for roads are calculated using the approach of Akçelik [1]. He describes Davidson's function, which is a general-purpose travel-time formula for transport planning purposes. To overcome some issues, Akçelik proposes an alternative formulation, described in Eq. (10.1).

$$t = t_0 + 0.25T_f \left(z + \left(z^2 + \frac{8J_A \cdot x}{C \cdot T_f} \right)^{0.5} \right), \quad (10.1)$$

where

- t = average travel time per unit distance,
- t_0 = min. (free-flow) travel time per unit distance,
- T_f = flow/analysis period,
- J_A = a delay parameter,
- $z = x - 1$
- $x = \frac{v}{C}$, degree of saturation,
- v = demand flow rate,
- C = capacity.

This function assumes a constant demand pattern and no initial queue at the start of the flow period. The travel time is defined as experienced by all vehicles *arriving* during the specified flow period.

The author also proposes some parameters for this travel-time function representing various road classes. We adjust these parameters for trucks using freeways in the Netherlands. Since the maximum velocity for trucks on freeways is 80 km/h, we choose the following parameters: $v_0 = 80$ and $J_A = 0.1$. The capacity is chosen with respect to the other parameters in our simulation.

For trains and barges the travel time is calculated differently. As they will leave at certain departure times and have a specified capacity, we have to calculate the time it has to wait for the next available departure. We assume fixed departure times, capacities and travel times.

Let us assume we have a container at time t that wants to travel over a link representing rail or waterway. Given departure time t_D and travel time t_T , the total travel time for the container at time t is given by $t_D - t + t_T$. However, the capacity of the train or barge, denoted by C , is not yet accounted for. The next C containers have to wait for the next departure, t_{D+1} and so on.

Public Information Models

The first two models are loosely based on how drivers in a road network can adjust their route in current traffic. We assume that the knowledge of the network is available to all agents, but only up to the current time.

In the first model, all agents will know the state of the network and act upon this information. However, once this choice has been made, it will not be altered anymore. In the second heuristic, agents can decide to switch routes as conditions in the network change.

Model 1: Minimum-Cost Routing Without Rerouting

The first model tries to find the minimum-cost route in a greedy way. For all containers that request a route, the shortest path is calculated with a dynamic shortest-path algorithm. Here we use the algorithm described in the paper by Ramalingam and Reps [16]. They obtain a new dynamic single-source shortest-path problem, which can be extended to a dynamic all-pairs shortest-path problem. All containers that request a route, are thus given a route based on the current conditions. The containers will follow this route and do not adjust on a later time. The first model is described in Algorithm 1. Note that this is the entire algorithm, including the simulation.

Algorithm 1 Heuristic 1: minimum-cost routing without rerouting

```

1:  $t = 0$ 
2: while  $t < \text{planning horizon}$  do
3:    $event = \text{first event from event queue}$ 
4:    $t = \text{time of event}$ 
5:   if  $event$  is an request route event then
6:     Calculate shortest path under current conditions
7:     Give this route to the current agent
8:     Add enter link event for time  $t$  on first link
9:   if  $event$  is a enter link event then
10:    Calculate travel time,  $t_T$ 
11:    Add leave link event for time  $t + t_T$ 
12:    Increase occupancy on this link by 1
13:    if agent is not yet at its destination then
14:      Add enter link event for the next link for time  $t + t_T$ 
15:   if  $event$  is an leave link event then
16:     Decrease occupancy on this link by 1

```

Model 2: Minimum-Cost Routing with Rerouting

An improvement on the previous model is the rerouting of agents. As each agent traverses the network, it encounters intermediate nodes. However, for the simulation it does not matter if the agent arriving at that node arrived from outside the network or from another node. Therefore, we can recalculate the shortest path for each agent. Since the conditions in the network have likely changed, so may have the shortest

path. This model is described in Algorithm 2. Again this algorithm includes the simulation.

Algorithm 2 Model 2: minimum-cost routing with rerouting

```

1:  $t = 0$ 
2: while  $t <$  planning horizon do
3:    $event =$  first event from event queue
4:    $t =$  time of event
5:   if  $event$  is a request route event then
6:     Calculate shortest path under current conditions
7:     Give this route to the current agent
8:     Add enter link event for time  $t$  on first link
9:   if  $event$  is a enter link event then
10:    Calculate travel time,  $t_T$ 
11:    Add leave link event for time  $t + t_T$ 
12:    Increase occupancy on this link by 1
13:    if agent is not yet at its destination then
14:      Create an request route event for the next node at time  $t + t_T$ 
15:   if  $event$  is an leave link event then
16:     Decrease occupancy on this link by 1

```

Full Information Model

The third model calculates the optimal route for each agent in a certain planning horizon. For a given planning horizon with full information, a user equilibrium is reached between all agents.

Model 3: Full Information, User Equilibrium Routing

For all classes of agents in the network, i.e., all origin-destination pairs, we know all arrivals in the planning horizon. For each of the links we should know the time- and state-dependent travel-time function. These travel-time functions are already discussed in section “[Description of Simulation](#)”. We should also know the costs of traversing a link. We focus on the user equilibrium case which uses Wardrop’s User Equilibrium Condition [21]: the system has a *user equilibrium* if no agent can improve his/her experienced travel time by unilaterally switching routes (for a given departure time).

This model is an iterative procedure which switches between a simulation of events under given route assignments and calculating new route assignments for differences in the travel times and costs. By rerouting the agents in each iteration we want to reach a fixed point in this system. In this fixed point the travel times and costs are not altered anymore and thus also the shortest paths are not altered

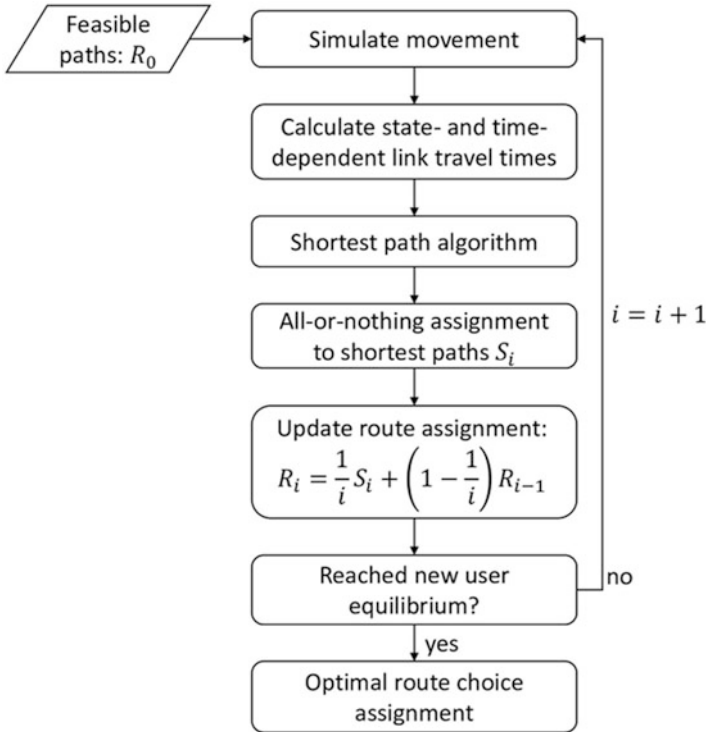


Fig. 10.1 Flow chart describing the general solution algorithm

anymore. Kaufman et al. [10] describe that this fixed point solution satisfies the UE condition.

In Fig. 10.1 one can see the global idea of our solution. The steps of the algorithm are described in further detail below. The algorithm is based on the algorithm for traffic networks described by Peeta and Mahmassani [14].

- **Step 1.** The iteration counter i is set to 0. For all origin-destination pairs and all time steps we calculate feasible paths. We used Dijkstra’s algorithm to create these paths with the free-flow travel times and costs. The first route assignment is sending all incoming agents over their shortest path. This route assignment is denoted by R_0 .
- **Step 2.** Perform simulation of the planning horizon by sending the agents over their path assignments R_i . In this simulation we log the changes throughout time, obtaining the number of agents on each link on each time. We also keep track of the individual travel times of the agents.
- **Step 3.** Compute the new time- and state-dependent travel times with the information gathered from the simulation. How we calculate these travel times is described in section “Description of Simulation”.

- **Step 4.** Compute the new time-dependent shortest paths for all origin-destination pairs, using the algorithm proposed by Ziliaskopoulos and Mahmassani [22]. This algorithm calculates the time-dependent shortest paths from all nodes in a network to a given destination node (denoted by N). Note that this is a discrete-time algorithm, thus it calculates the shortest paths for each time step over a given time horizon. It is based on Bellman's principle of optimality.
- **Step 5.** We create an auxiliary route assignment by the *all-or-nothing assignment*. The all-or-nothing assignment basically sends all agents from a certain origin-destination pair that want to depart at a certain time step over the same route. The auxiliary route assignment we create, denoted by S_i , is sending all agents on their shortest path calculated in the previous step.
- **Step 6.** Then we calculate the new route assignment. To reach convergence, the new route assignment is a combination between the old route assignment and the auxiliary route assignment calculated in the previous step. A new route assignment is calculated with the use of the Method of Successive Averages (MSA):

$$R_i = \frac{1}{i}S_i + \left(1 - \frac{1}{i}\right)R_{i-1}. \quad (10.2)$$

- **Step 7.** The convergence criterion is based upon how much the occupancy on the links changes from one simulation to the next. We keep the log from the previous simulation and the one from this simulation and check the difference in occupancy on each link. If this difference is less than 5%, the convergence criterion is met.
- **Step 8.** If the convergence criterion is met, terminate the algorithm. Otherwise repeat from step 2 with the new route assignments.

It is important to know that the shortest-path algorithm in step 4 is discrete and, therefore, the planning horizon is divided into multiple time steps. At the end of the algorithm we know the shortest paths for all agents at each time step. To cope with this fact in step 2, we group all arrivals in the same time step and handle them as if they all occur at the beginning of this time step.

Results

We developed three different models: two based on public information and one based on full information. We elaborate on some small networks that highlight differences between and the performance of those models. The first network can be found in Fig. 10.2. Here one can see the length of the links and the capacity of each link (length/capacity). For each of these links the costs increase with the amount of agents in the network. One can understand that when a link is over-utilised, i.e., there are more agents on the link than the capacity allows, these costs increase steeply.

Fig. 10.2 Example 1, with length and capacity of nodes (notation: length/capacity)

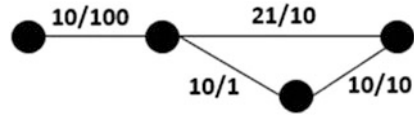
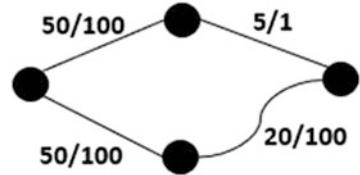


Fig. 10.3 Example 2, with length and capacity of nodes (notation: length/capacity)



When there is enough capacity for the amount of agents, the costs are similar to the free-flow travel costs. The travel times follow from Eq. (10.1), using $t_0 = 1/6.666$ and $J_a = 0.1$.

In this first example we want agents to travel from the left of this network to the right. This means they all first need to traverse a link that has a length of 10 and a capacity of 100. Afterwards, they can choose between one route with length 21 and capacity 10 or a route comprised of two links: one with length 10 and capacity 1 and one with length 10 and capacity 10.

In the second example, Fig. 10.3, agents also need to travel from left to right. There are two routes available, both comprised of two links. Both routes start with a link with length 50 and capacity 100. The first route then has a link with length 20 and capacity 100, while the other route has a link with length 5 and capacity 1.

In both examples we assume that 10 agents want to traverse the network. Firstly, we assume that all these agents want to traverse the network at the same time. For example 1, these results can be found in Fig. 10.4 and for example 2 in Fig. 10.5.

In Fig. 10.4, we can see that model 1 performs very poorly compared to the other two. The reason for this behaviour is that it looks at the network and sees the route via the link with capacity 1 as the shortest path, since this path has length 30 and the other path has length 31. All agents depart at the same time; therefore, no agents are on the link with the small capacity and thus the shortest route will seem to be the same for all agents. Therefore, all agents will travel via the lower route. However, when the agents do arrive at the second link in their route, all the other agents also want to traverse that route, which means there is an enormous extra cost for this link. Model 2 does not show this behaviour, since the shortest path is recalculated at the first intermediate node. Here the first agent will travel the route with length 30, but the next agent sees that the link is already in use and will, therefore, use the link with length 21. This recalculation of the shortest path is not done in heuristic 1. The solution algorithm also divides the agents, such that they all have a low cost of travelling.

Figure 10.5 shows the costs for example 2. Here both models with public information perform badly. The reason for this is that both models see the route on the upper path of the network as the shortest path at the departure time of the agents. Therefore, both models send all agents on the top link with length 50. Model

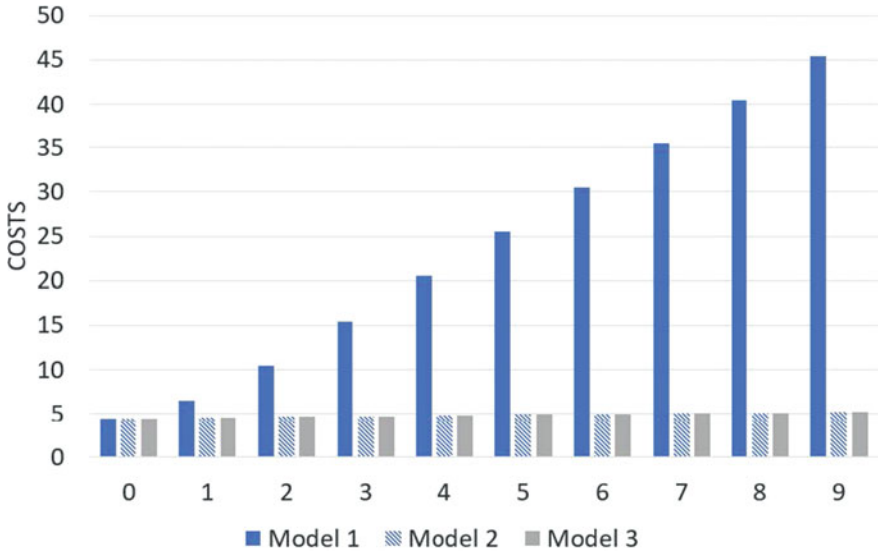


Fig. 10.4 Costs for each agent 0, ..., 9 in example 1 with clustered travelling

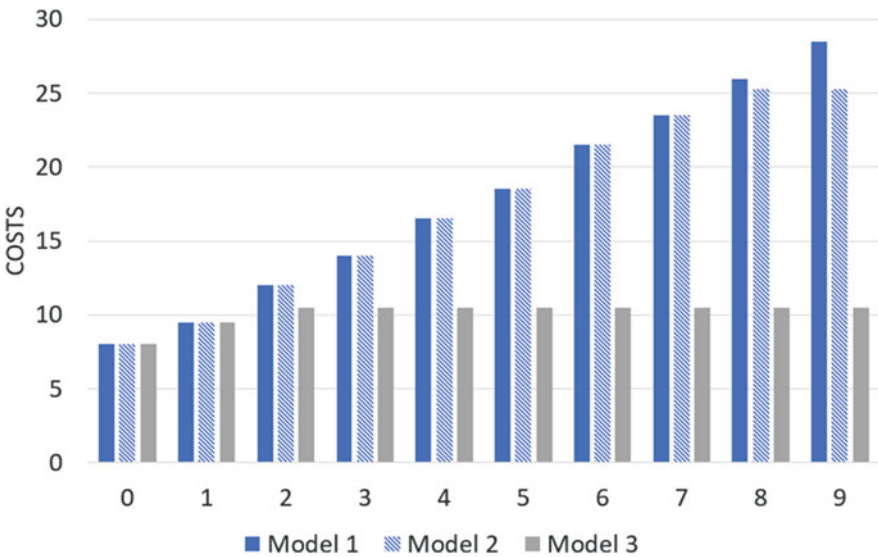


Fig. 10.5 Costs for each agent in example 2 with clustered travelling

1 does not have a possibility to reroute and, therefore, all agents will be stuck on this route. This creates a situation where all agents want to traverse a route of capacity 1. This leads to extra costs. Model 2 *does* have a possibility to reroute, but only at the intermediate node. This means that at this point agents would have to travel back in

Fig. 10.6 Example 3

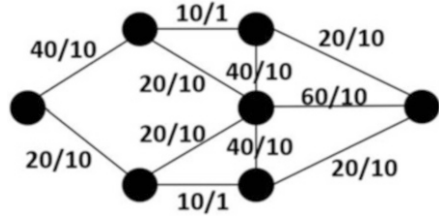


Table 10.1 Results for example 3

	Model 1	Model 2	Model 3
Mean travel time	11.4	10.4	10.1
Max travel time	19.0	12.0	13.1

order to avoid the link with capacity 1. But this means extra costs of at least a link with length 50. Therefore, most agents will still be routed on the link with capacity 1. However, one can see that agents 8 and 9 have a lower cost for model 2. This means that at this point it is actually beneficial to go back to the first node and take the bottom route. The solution algorithm has full information and thus knows that sending all agents over the top route will lead to trouble. Therefore, the solution algorithm already divides the agents over the top and bottom route from the start.

Now we show the results for a somewhat larger example as depicted in Fig. 10.6. For this network, given a number of 80 departs per time step (of 5 min) travelling from left to right, repeated for 10 time steps. Again, arriving at a certain arc, the travel time on that arc is calculated using Eq. (10.1).

The results of the 3 models in this network are shown in Table 10.1. We see that indeed model 1 performs worst and model 3 performs best on average. However, we see that model 2 has better worst case statistics.

Conclusions

In this chapter we investigated the effect of different information availability in agent-centric sychromodal networks. For the optimal routing with only public information we developed our two models. Obviously, model 2, which allows for rerouting, outperforms model 1. The reason for this is that agents (containers) are able to change their route when new information enters the network. Model 2 makes use of all public information that is available. The optimal routing with full information could be determined with the third model. An iterative process is used to find optimal routes for the entire planning horizon. The models were tested on small examples and on a larger one. We mention the advantages and disadvantages of each model below.

- *Model 1*: performs well if the arrivals of the orders are spread out over the planning horizon. It needs time to see the congestion of the network build up. The

model performs poorly on networks where orders need to travel a large distance. For all orders, the routes are determined at their departure time. That means that the more time it takes for an order to move through the network, the more can change in the network. This will result in sub-optimal routes.

- *Model 2*: performs better when the orders are spread out over the planning horizon. Although it does have the availability to reroute certain containers, the availability of alternative routes plays a huge role on its performance. If one of the routes seems like a short path at departure time, the other routes need to be reachable from intermediate nodes on that route. If there are no alternative routes available, the rerouting will not result in smaller costs. However, this model does not perform worse when the orders in the network need to travel longer.
- *Model 3*: results in optimal routes for all discrete cases. This means that the parameters of the network and the network itself have less effect on this method. However, real life instances are continuous and the discrete nature of the simulation-based solution method is a disadvantage. Note that this problem can be solved by taking smaller time steps within the algorithm.

In further research the models should be applied to real life data to show the economic effect of information sharing in synchromodal transportation networks. Next, research should be done on ways to reach the global optimum in an agent based (user equilibrium reaching) network, i.e., by introducing tolls or other kind of (artificial) road pricing.

References

1. Akcelik, R. (1991). Travel time functions for transport planning purposes: Davidson's function, its time dependent form and alternative travel time function. *Australian Road Research*, 21(3)
2. Ben-Akiva, M., Bierlaire, M., Bottom, J., Koutsopoulos, H., & Mishalani, R. (1997). Development of a route guidance generation system for real-time application. In *8th IFAC Symposium on Transportation Systems, TRANSP-OR-CONF-2006-063*.
3. Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H., & Mishalani, R. (1998). DynaMIT: A simulation-based system for traffic prediction. In *DACCORD Short Term Forecasting Workshop* (pp. 1–12).
4. Birge, J., & Ho, J. (1993). Optimal flows in stochastic dynamic networks with congestion. *Operations Research*, 41(1), 203–216.
5. Carey, M., & Subrahmanian, E. (2000). An approach to modelling time-varying flows on congested networks. *Transportation Research Part B: Methodological*, 34(3), 157–183.
6. Dafermos, S. (1980). Traffic equilibrium and variational inequalities. *Transportation Science*, 14(1), 42–54.
7. Dia, H. (2002). An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. *Transportation Research Part C: Emerging Technologies*, 10(5), 331–349.
8. Friesz, T., Luque, J., Tobin, R., & Wie, B. (1989). Dynamic network traffic assignment considered as a continuous time optimal control problem. *Operations Research*, 37(6), 893–901.
9. Janson, B. (1991). Convergent algorithm for dynamic traffic assignment. *Transportation Research Record* 1328.

10. Kaufman, D., Smith, R., & Wunderlich, K. (1998). User-equilibrium properties of fixed points in dynamic traffic assignment. *Transportation Research Part C: Emerging Technologies*, 6(1), 1–16.
11. Mahmassani, H., & Jayakrishnan, R. (1991). System performance and user response under real-time information in a congested traffic corridor. *Transportation Research Part A: General*, 25(5), 293–307.
12. Merchant, D., & Nemhauser, G. (1978). A model and an algorithm for the dynamic traffic assignment problems. *Transportation Science*, 12(3), 183–199.
13. Nagurney, A. (2013). *Network economics: A variational inequality approach* (Vol. 10). Springer.
14. Peeta, S., & Mahmassani, H. S. (1995). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, 60(1), 81–113.
15. Peeta, S., & Ziliaskopoulos, A. (2001). Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1(3), 233–265.
16. Ramalingam, G., & Reps, T. (1996). An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2), 267–305.
17. Ran, B., Boyce, D., & LeBlanc, L. (1993). A new class of instantaneous dynamic user-optimal traffic assignment models. *Operations Research*, 41(1), 192–202.
18. Ran, B., & Shimazaki, T. (1989). A general model and algorithm for the dynamic traffic assignment problems. In *Transport Policy, Management & Technology Towards 2001: Selected Proceedings of the Fifth World Conference on Transport Research* (Vol. 4).
19. Tavasszy, L., Behdani, B., & Konings, R. (2015). Intermodality and synchromodality. SSRN.com.
20. Ulmer, M. W., Heilig, L., & Voß, S. (2017). On the value and challenge of real-time information in dynamic dispatching of service vehicles. *Business & Information Systems Engineering*, 59(3), 161–171.
21. Wardrop, J. (1900). Some theoretical aspects of road traffic research. In *Inst Civil Engineers Proc London* (Vol. 36, pp. 325–378).
22. Ziliaskopoulos, A., & Mahmassani, H. (1993). Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. In *Transportation research record* (pp. 94–94).

Chapter 11

User Equilibrium in a Transportation Space-Time Network



L. A. M. Bruijns

Abstract This chapter focuses on synchronodal planning problems in which information is shared between all agents in the system and they choose their routes based on an individual optimisation objective. We show the effect of the information availability by developing three different methods to determine the optimal paths, to motivate logistic players to cooperate in a synchronodal system.

Introduction

The agents in such a synchronodal network can be logistic service providers or clients controlling the stream of their containers, or intelligent containers or other smart logistic units themselves. In this chapter, we again focus on a ‘selfish’ system as presented in Fig. 1.1 and a mathematical model using a Space-Time Networks (STN), as presented in Sect. “[Modelling the Problem as a MCMC Flow Problem on a Space-Time Network](#)”. On this STN a nonnegative integral Minimum Cost Multi-Commodity Flow problem (MCMCF) is solved to get the overall optimal (social) solution. However, if links have capacity constraints and there are multiple agents travelling or sending their commodities over the network, some agents may not receive the shortest or most economical path. They may be unhappy (in a selfish model) with the total solution, even when this solution is the optimal solution for all agents together, a system optimal solution. Note that all kind of modalities (or combinations) can be modelled using this approach. To reach a solution in which all agents are satisfied and do not want to change their paths, we would actually need a ‘User Equilibrium’ (UE) solution. However, in most cases this UE is overall a worse solution than the overall optimal ‘System Optimal’ (SO) solution. There is an expected gain [22] for the total system in case of cooperation, reaching a system optimal solution. Swamy [23] shows that selfish, here meaning locally optimising, systems have their price: they prove that, in traffic assignment problems, travel times

L. A. M. Bruijns (✉)
TNO, The Hague, The Netherlands

induced by selfish agents might be the same as the total travel time incurred by optimally routing twice as much traffic and indicate that adding central control or incentives gives an overall improvement of the system. However, in networks with high load the performance might not suffer too much, as can be found in [13]. So optimising the total network and then sharing the benefits from an overall optimal solution between all agents is beneficial for all. On the other hand, it is not easy, as it requires a mental shift to get to give up control.

In this work we propose for the first time a definition of a UE solution in a logistic STN. We then provide a method how to change the arc weights of the STN to create and find a UE solution in the modified STN, by adding tolls, that equals the system optimal solution. Note that the practical implementation is far away, but this can be used to propose a reallocation of costs in which the benefit of the social optimal, with respect to the UE in the original STN, is shared in a fair way. In terms of Fig. 11.1, we want to get the ‘social’ solution in a ‘selfish’ network. For the second part, changing the arc weights to create a UE solution that equals the SO solution, we propose the following algorithm. The first step in this ‘all toll algorithm’ is to calculate the SO based on the path costs of agents travelling from their origin to their destination. The next step is to calculate tolls that are added to the paths in the network. These tolls are used to adjust the path costs, such that we can offer the agents a choice of tolled paths. Now, when the agent gets assigned its cheapest tolled paths, those paths are in the SO solution and the solution is UE as well. The solution is a UE because the offered path costs are the cheapest option according to the information available for the agent, the new tolled STN. In the next section we discuss the literature on User Equilibria and toll systems in traffic assignment problems. To the best of the authors’ knowledge no literature exists for UE in freight logistic networks. In Sect. “[User Equilibrium in STN](#)” the definition of UE in STN is given and a method is presented to find a UE that equals the solution of the system optimal. The method is illustrated by two examples in Sect. “[Numerical Examples](#)”. We conclude with some remarks and directions for future research.

Literature Review

Most of the literature about User Equilibria is based on network congestion, where travel times on roads depend on occupancy of travelling arcs, as in traffic assignment problems. Van Essen et al. [5] give a proper review of ways to force a UE into a System Optimum (SO) by diffusing travel information to stimulating some agents to travel non-selfishly to achieve cheaper total costs. Peeta and Mahmassani [13] investigate both the SO and the UE Time-Dependent Traffic Assignment. They show that the more goods have to be transported, the more the solutions of the two models differ from each other. Bar-Gera [1] provides a solution method for the UE traffic assignment problem which is computationally efficient, memory conserving and an origin-based solution method. Xu et al. [18] propose a stochastic UE for a passenger transport network.

Miyagi et al. [12] consider a traffic assignment problem from the view of game theory. They assume drivers have knowledge of the network and a Nash Equilibrium (which corresponds to a UE) is achievable. Wagner [16] shows that the existence of a Nash Equilibrium is guaranteed under some natural assumptions on the travel time models. Also Wang and Yang [17] show the equality of Nash Equilibrium and UE. Levy et al. [10] consider selfish agents in a traffic assignment problems and apply properties of game theory on traffic problems. They start from finding a UE solution, in which all agents take the best route for themselves, based on their route choice experiences in the past. The question then is if it is possible to obtain a System Optimal solution, in which agents are still selfish.

The relationship between the UE and the System Optimal can be examined by the Price of Anarchy [15], a system often used in both economics and game theory, that measures how the efficiency of a system degrades due to selfish behaviour of its customers. Bar-Gera et al. [2] consider the UE problem with the focus on spreading flow over the network (not time-dependent). They also introduce several criteria which can be taken into consideration for choosing UE solution methods. Their most important addition to the subject is the condition of proportionality: the same proportions apply to all travellers facing a choice between a pair of alternative paths, regardless of their origins and destinations.

Corman et al. [3] consider the application of multimodal transport to provide a UE solution, with the choice of modality based on the wishes of the agents. They assume that agents have access to a system for publishing demand and offering transportation possibilities. Moreover, they assume everybody has access to truck transportation, so transport is always possible, regardless of the fact that other modalities are not available. They define every agent as one unit of transport, which has to choose one specific mode for the whole travel distance. The goal is to assign agents to modes in such a way that no agent will change its departure time and its route (and thus will not change its mode), to provide all agents a sufficient route.

One commonly used approach for creating a UE is by using tolls. Hearn and Ramana [8] make use of a toll pricing system by adding a toll term to the cost function for each arc. They also describe the Robinhood formulation, in which the sum of all tolls must be zero, so that there is no profit for the system. In this case they calculate the toll after a System Optimal solution is found. According to Florian and Hearn [6], the application of those types of toll is hard to implement on traffic networks regarding variable travel times, although the selective use of negative tolls to influence route choice of users might have some appeal. Yang and Han [7] investigate the use of tolls with the help of the price of anarchy. Yang and Zhang [21] constructed an anonymous link toll system to add traveller-dependent tolls. They concluded that there exist nonnegative links tolls identical to all users to decentralise the Wardropian System Optimum as a UE-CN (Cournot-Nash) mixed equilibrium, and the valid toll set is made up of a convex set of linear equalities and inequalities. They use nonnegative tolls only. Yang and Huang [19] state that Value Of Time (VOT) is a very important concept in transportation system modelling. The VOT of an order is a constant which denotes the importance of that agent. Didi-Biha, Marcotte and Savard [4] also use nonnegative tolls. Their goal is to maximise the toll

revenue for the highway authority while the users of the network want to minimise their travelling costs. They introduce their bi-level programming Toll Optimisation Problem, both arc, arc-path and path based. Yang [20] proved the existence of a Pareto refunding scheme that returns the congestion pricing revenues to all users to make everyone better off. This Pareto refunding scheme refunds class-specific and OD-specific toll revenue equally to all users in the same Origin-Destination pair in the same user class. User Equilibria in (multi-) agent environments are also described as consensus seeking agents [11, 14].

User Equilibrium in STN

In this chapter we propose the use of tolls on paths within as STN. For convenience we will use the terms agent for the controller of (at least) one unit of transport. This agent can send an order (multiple units) for transportation within the network and as unit we will say container. A Physical Internet system or another self-organising system with smart units will fit within this method. For each order there may be multiple paths to travel by within the STN. We propose the following definition for a UE within an STN:

Definition 11.1 (User Equilibrium) A UE is reached when each agent can use their cheapest paths.

This is obviously not always possible when concerning only the initial networks, so we need to adjust the initial network using the path tolls to reach this UE. We will assume that agents are not familiar with the path costs in the initial STN, they only have knowledge of the tolled path costs. In this section, our goal is to find a Path Tolloed UE. Our approach is to first find an SO solution and then add tolls to paths, which create a new cost scheme for paths. When we offer the STN with the adjusted path costs to the agents, they can selfishly choose routes, and the outcome of their path choice will correspond with the path to order assignment within the SO. The difference between the two networks provides insight in the offered fairness by the solution.

The way of finding tolls that give us a UE solution in an initial SO problem is described in Algorithm 1, which is partly based on the solution algorithms used by Hearn and Ramana [8] and Jiang and Mahmassani [9]. The difference with the framework of Hearn and Ramana is that we do not define the toll set, because in our approach there is no need to obtain this total set. The difference with Jiang and Mahmassani is that we apply tolls on paths instead of updating path assignment.

The Space-Time Network is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, consisting of a set of nodes $v \in \mathcal{V}$ and a set of directed arcs $a \in \mathcal{A}$ (Table 11.1). Each arc a is a link between two nodes, an origin node v_1 and an end node v_2 : $a = (v_1, v_2)$, along which a container can travel. We use x_a to denote the number of units of flow along arc a . An Origin-Destination-pair (OD-pair) w is a pair of two nodes, origin location w_O and destination location w_D , so $w = (w_O, w_D)$, which is not necessarily an arc.

Table 11.1 Used notation

$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	STN Graph
\mathcal{V}	Set of nodes of the STN
\mathcal{A}	Set of directed arcs of the STN
x_a	The number of units of flow along arc a
$w = (w_O, w_D)$	Origin-Destination-pair (OD-pair) or order between origin location w_O and destination location w_D
\mathcal{W}	Set of all OD-pairs/orders
d_w	Demand of order w
p	Path, order of adjacent arcs, in the STN
\mathcal{P}_w	Set of all paths for OD-pair w
\mathcal{P}	Total path set
f_p	Path flow of path p
c_a	Cost of arc a
C_w^p or C^p	Path cost (of order w)
m_a or m_p	Capacity of arc a or path p
δ_{ap}	Indicator whether arc a is in path p
k_w	Cost of shortest possible path of order w in the empty STN
r_w	Cost ratio of order w
$r_{\beta w}$	Tolled cost ratio of order w
$h_{in,w}$	Set of paths used in SO solution by order w
$h_{out,w}$	Set of paths not used in SO solution by order w
$NP - \beta$	Optimisation problem for finding the tolls
η_a and Γ_a	Bottleneck sets of paths and orders for arc a
\mathcal{A}_η	Bottleneck set
\mathcal{Q}_w	Set of cheapest paths for order w
\mathcal{V}_s	Connected components s , which is a list of visited nodes in connected component s

The number of containers an order wants to transport from w_O to w_D is denoted by d_w , the demand of order w . A path p consists of a sequence of (non-horizontal) adjacent arcs between two nodes. In our problem we only consider paths between origin and destination nodes. f_p denotes the path flow of path p (always integer), with $p \in \mathcal{P}_w$, $w \in \mathcal{W}$, where \mathcal{P}_w is the set of all paths for OD-pair w and \mathcal{W} is the set of all OD-pairs. The total path set is $\mathcal{P} := \bigcup_{w \in \mathcal{W}} \mathcal{P}_w$. The costs of an arc a are denoted by c_a and the path costs of path p are denoted by C_w^p or C^p . The capacity of an arc is denoted by m_a and the capacity of a path is denoted by m_p . The available arcs in a path are denoted by

$$\delta_{ap} = \begin{cases} 1 & \text{if } a \text{ is contained in } p, \forall a \in \mathcal{A}, p \in \mathcal{P}, \\ 0 & \text{otherwise.} \end{cases}$$

After finding the SO solution, we want to find path tolls (β_w^p) such that each agent is satisfied with its route, and thus a UE is achieved. We only use tolls to obtain both an SO and UE solution, so we do not need to make profit on the tolls. We will search for tolls that are as low as possible and we require that all tolls paid or received by agents sum up to zero. We will now go through the proposed algorithm. Finding the path tolls starts with an SO solution (Step 1), solving of which results in the optimal flows \underline{f}_p (Step 2). Now, define the set of paths used in the SO solution (Step 3) by

$$h_{in,w} := \left\{ p \mid \underline{f}_p > 0, p \in \mathcal{P}_w \right\},$$

and the sets of all other paths (which are not in the SO solution) by

$$h_{out,w} := \left\{ p \mid \underline{f}_p = 0, p \in \mathcal{P}_w \right\}.$$

We then solve a Non-linear Programming Problem NP- β that consists of an objective function that minimises the path tolls of a certain path set, and a set of constraints. To realise low tolls on paths in $h_{out,w}$ we will minimise the tolls added to paths which are not in the SO solution, so we use as the objective function:

$$\sum_{w \in \mathcal{W}} \sum_{p \in h_{out,w}} |\beta_w^p|.$$

To let the tolls sum up to zero we use the constraint:

$$\sum_{p \in h_{in,w}} \beta_w^p \underline{f}_p = 0.$$

So, if there are tolls needed to obtain a UE, there will be one or multiple agents who need to pay toll, as well as there are one or multiple agents who receive toll. This last group thus has a discount on the routes which we want those agents to take. We do not want the toll received by an agents to be higher than the initial path cost (which would mean that an agents does not have to pay but only receives money for choosing a certain path), so we use the constraints:

$$C_w^p + \beta_w^p \geq 0 \quad \forall p \in \mathcal{P} \quad \iff \quad \beta_w^p \geq -C_w^p \quad \forall p \in \mathcal{P}.$$

Now, the NP- β (step 4) consists of the following constraints:

$$\sum_{w \in \mathcal{W}} \sum_{p \in h_{in,w}} \beta_w^p \underline{f}_p = 0 \tag{11.1}$$

$$\beta_w^i - \beta_w^j \leq C_w^j - C_w^i \quad \forall (i, j), i \in h_{in,w}, j \in h_{out,w} \quad \forall w \in \mathcal{W} \tag{11.2}$$

$$\beta_w^p \geq -C_w^p \quad \forall p \in \mathcal{P}, \tag{11.3}$$

where Constraint (11.1) ensures all tolls on paths used in the SO solution sum up to zero, Constraint (11.2) ensures the paths used in the SO solution for one order, have equal or lower costs than the paths for that order which are not in the SO solution, and Constraint (11.3) ensures no tolled cost can become negative.

The NP- β in Algorithm 1 Step 4 is non-linear, which makes this problem hard to solve. We therefore use the equivalent linear formulation of the problem:

$$\begin{aligned} \min \quad & \sum_{w \in \mathcal{W}} \sum_{p \in h_{out,w}} \gamma_w^p \\ \text{s.t.} \quad & \sum_{w \in \mathcal{W}} \sum_{p \in h_{in,w}} \beta_w^p \underline{f}_p = 0 \\ & \beta_w^i - \beta_w^j \leq C_w^j - C_w^i \quad \forall (i, j), i \in h_{in,w}, j \in h_{out,w} \quad \forall w \in \mathcal{W} \\ & \beta_w^p \geq -C_w^p \quad \forall p \in \mathcal{P} \\ & \beta_w^p \leq \gamma_w^p \quad \forall p \in \mathcal{P} \end{aligned} \tag{11.9}$$

$$- \beta_w^p \leq \gamma_w^p \quad \forall p \in \mathcal{P} \tag{11.10}$$

$$\gamma_w^p \geq 0 \quad \forall p \in \mathcal{P}, \tag{11.11}$$

where γ_w^p replaces the absolute value variable $|\beta_w^p|$ with Constraints 11.9–11.11.

Solving the NP- β (step 5) leads to toll that can be used to change the path costs (Step 6). The desired outcome of Algorithm 1 is that the solution to the SO- β problem is equal to the initial SO problem (Step 7). The resulting path costs are the only costs that are showed to the agents, so the agents do not have any knowledge about the initial STN and those path costs.

Numerical Examples

We illustrate the algorithm by solving two examples. In the first example there are three locations, $\mathcal{V} = \{1, 2, 3\}$, and five time steps. We have two connections between location $l = 1$ and $l = 2$ and two between $l = 2$ and $l = 3$. Those arcs all have capacity $m_a = 1$, and $m_a = \infty$ for waiting arcs. We have two orders, order 1 and 2 both start at location 1, order 1 has to go to $l = 2$ and order 2 to $l = 3$. Every node column shows a time step and each arc has cost $c_a = 1$. The two possible solutions are given in Fig. 11.1, with s_w denoting the starting point and e_w denoting the end point for order w .

In Fig. 11.1a we see the SO solution, resulting from Step 1 and Step 2, that is the solution where the total costs are minimised. Here order 1 is delivered first with cost $C_1^a = 2$ and therefore order 2 can only take path bd with cost $C_2^{bd} = 5$. In Fig. 11.1b a solution is given where both orders pay cost 4, that is path b for order 1, and path ac for order 2.

Algorithm 1 Calculating path tolls

1: Create SO problem:

$$\begin{aligned}
 & \min \sum_{p \in \mathcal{P}} C_w^p f_p \\
 & \text{s.t. } x_a = \sum_{p \in \mathcal{P}} \delta_{ap} f_p \quad \forall a \in \mathcal{A} \\
 & \quad \sum_{p \in \mathcal{P}_w} f_p = d_w \quad \forall w \in \mathcal{W} \\
 & \quad x_a \leq m_a \quad \forall a \in \mathcal{A} \\
 & \quad f_p \in \mathbb{N}_0 \quad \forall p \in \mathcal{P} \\
 & \quad x_a \in \mathbb{N}_0 \quad \forall a \in \mathcal{A}
 \end{aligned} \tag{11.4}$$

2: Solve SO problem, output: path flow vector \underline{f} .

3: Create two lists for each order w : $h_{in,w} = \{p \mid \underline{f}_p > 0, p \in \mathcal{P}_w\}$, $h_{out,w} = \{p \mid \underline{f}_p = 0, p \in \mathcal{P}_w\}$.

4: Create NP- β :

$$\min \sum_{w \in \mathcal{W}} \sum_{p \in h_{out,w}} |\beta_w^p| \tag{11.5}$$

$$\text{s.t. } \sum_{w \in \mathcal{W}} \sum_{p \in h_{in,w}} \beta_w^p \underline{f}_p = 0 \tag{11.6}$$

$$\beta_w^i - \beta_w^j \leq C_w^j - C_w^i \quad \forall (i, j), i \in h_{in,w}, j \in h_{out,w} \quad \forall w \in \mathcal{W} \tag{11.7}$$

$$\beta_w^p \geq -C_w^p \quad \forall p \in \mathcal{P} \tag{11.8}$$

where Constraint (11.6) ensures the total toll sum of the chosen paths to be zero, Constraint (11.7) ensures that the paths in the SO solutions are the ones with cheapest costs $C_{\beta_w}^p$ and Constraint (11.8) ensures no path can have a negative $C_{\beta_w}^p$ cost.

5: Solve NP- β , output: β_w^p .

6: Add tolls β_w^p to the SO problem, SO- β :

$$\begin{aligned}
 & \min \sum_{p \in \mathcal{P}} (C_w^p + \beta_w^p) f_p \\
 & \text{s.t. } x_a = \sum_{p \in \mathcal{P}} \delta_{ap} f_p \quad \forall a \in \mathcal{A} \\
 & \quad \sum_{p \in \mathcal{P}_w} f_p = d_w \quad \forall w \in \mathcal{W} \\
 & \quad x_a \leq m_a \quad \forall a \in \mathcal{A} \\
 & \quad f_p \in \mathbb{N}_0 \quad \forall p \in \mathcal{P} \\
 & \quad x_a \in \mathbb{N}_0 \quad \forall a \in \mathcal{A}
 \end{aligned}$$

7: Solve SO- β problem, output path flow vector \underline{f} .

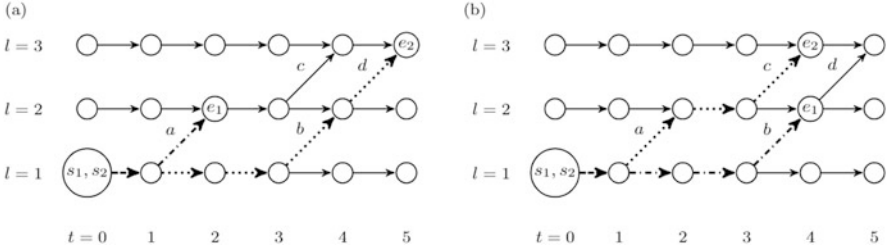


Fig. 11.1 STN with two orders, with $m_a = 1$ for all arcs between two different locations, $m_a = \infty$ otherwise. (a) The SO solution. (b) The alternative solution

We can see that each order has its own preferable solution, that is the one in which they can travel via arc a , which is in the cheapest path for both orders. We have path costs

$$C_1^a = 2, C_1^b = 4, C_2^{ac} = 4, C_2^{ad} = 5 \text{ and } C_2^{bd} = 5.$$

and the path sets following from the SO solution as obtained in Algorithm 1 in Step 3:

$$h_{in,1} = \{a\}, h_{in,2} = \{bd\}, h_{out,1} = \{b\}, h_{out,2} = \{ac, ad\}.$$

The tolls given by Step 4 and Step 5 are

$$\beta_1^a = 1, \beta_2^{bd} = -1,$$

so all tolls on paths $p \in \bigcup_{w \in \mathcal{W}} h_{out,w}$ are zero and so is the objective value of the NP- β . The best solution of the NP- β is indeed the solution as obtained from Algorithm 1 Step 5:

$$\beta_1^a = 1, \beta_1^b = 0, \beta_2^{ac} = 0, \beta_2^{ad} = 0, \beta_2^{bd} = -1$$

and with those tolls we obtain the path costs:

$$C_{\beta 1}^a = 3, C_{\beta 1}^b = 4, C_{\beta 2}^{ac} = 4, C_{\beta 2}^{ad} = 5 \text{ and } C_{\beta 2}^{bd} = 4,$$

so both orders can travel via their cheapest paths, so both an SO and a UE are obtained.

In the second example we have three orders, all with different demand: $d_1 = 3$ from location 1 to 2, $d_2 = 3$ from location 1 to 3 and $d_3 = 1$ from location 2 to 3. An SO solution is given in Fig. 11.2, with s_w and e_w denoting the start end point of order w , respectively. All travelling arcs have capacity 1, except for arcs a , c and f , which have capacity $m_a = m_c = m_f = 2$, which we graphically show by multiple arcs between a pair of nodes.

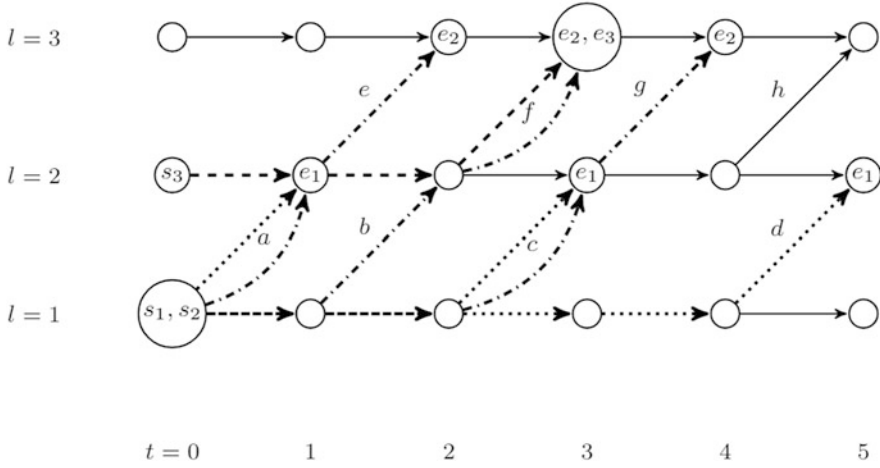


Fig. 11.2 STN with two orders, $d_w \geq 1 \forall w \in \mathcal{W}$, with $m_a = m_c = m_f = 2$, $m_{a_i} = 1$ for $a_i \in \mathcal{A} \setminus \{a, c, f\}$, $m_{a_i} = \infty$ on waiting arcs. The denoted solution is SO

We have path costs

$$\begin{aligned}
 C_1^a &= 1, & C_1^b &= 2, & C_1^c &= 3, & C_1^d &= 5, \\
 C_2^{ae} &= 2, & C_2^{af} &= 3, & C_2^{ag} &= 4, & C_2^{ah} &= 5, & C_2^{bf} &= 3, & C_2^{bg} &= 4, & C_2^{bh} &= 5, & C_2^{cg} &= 4, & C_2^{ch} &= 5, \\
 C_3^e &= 2, & C_3^f &= 3, & C_3^g &= 4, & C_3^h &= 5,
 \end{aligned}$$

The path sets following from the SO solution are:

$$\begin{aligned}
 h_{in,1} &= \begin{Bmatrix} a, c, d \\ 1, 1, 1 \end{Bmatrix}, & h_{out,1} &= \begin{Bmatrix} a, b, c \\ 1, 1, 1 \end{Bmatrix}, \\
 h_{in,2} &= \begin{Bmatrix} ae, bf, cg \\ 1, 1, 1 \end{Bmatrix}, & h_{out,2} &= \begin{Bmatrix} af, ag, ah, bf, bg, ch \\ 2, 1, 1, 1, 1, 1 \end{Bmatrix}, \\
 h_{in,3} &= \begin{Bmatrix} f \\ 1 \end{Bmatrix}, & h_{out,3} &= \begin{Bmatrix} e, f, g, h \\ 1, 1, 1, 1 \end{Bmatrix}.
 \end{aligned}$$

Table 11.2 $h_{out,w}$

Order	1			2						3			
$p \in h_{out,w}$	a	b	c	af	ag	ah	bf	bg	ch	e	f	g	h
Initial path costs	1	2	3	3	4	5	3	4	5	2	3	4	5
Tolls	$2\frac{1}{3}$	$1\frac{1}{3}$	$\frac{1}{3}$	0	0	0	0	0	0	0	-1	0	0
Resulting path costs	$3\frac{1}{3}$	$3\frac{1}{3}$	$3\frac{1}{3}$	3	4	5	3	4	5	2	2	4	5

Table 11.3 $h_{in,w}$

Order	1			2			3
$p \in h_{in,w}$	a	c	d	ae	bf	cg	f
Initial path costs	1	3	5	2	3	4	3
Tolls	$2\frac{1}{3}$	$\frac{1}{3}$	$-1\frac{2}{3}$	1	0	-1	-1
Resulting path costs	$3\frac{1}{3}$	$3\frac{1}{3}$	$3\frac{1}{3}$	3	3	3	2

We see that none of the orders can travel via their cheapest paths, so we need tolls to create a UE. Solving the NP- β gives us

$$\beta_1^a = 2\frac{1}{3}, \beta_1^c = \frac{1}{3}, \beta_1^d = -1\frac{2}{3}, \beta_2^{cg} = -1, \beta_2^{ae} = 1, \beta_3^f = -1, \beta_1^b = 1\frac{1}{3}.$$

Note that path $b \in h_{out,1}$, so the toll on that path is not actually paid (Tables 11.2 and 11.3).

Conclusions

The goal of this chapter was to provide a method to obtain a User Equilibrium in a logistic, intermodal or synchronomodal Space-Time Network (STN), in which we transport containers for multiple agents. We defined a UE as the solution where each agent can send its containers via its cheapest paths. We expanded this goal to also finding a solution of assigning containers to modes where the solution is System Optimal and by adding tolls a UE simultaneously. The first step in all toll algorithms is to calculate the SO based on the path costs of containers travelling from their origin to their destination. The next step is to calculate tolls that are added to the path or order costs, depending on what kind of tolls we considered.

When applying path based tolls, we assume agents do not know the path costs of the initial network (and thus also do not know their initial cheapest paths). Here the tolls are used to adjust the path costs, such that we can offer the agents a choice of tolled paths. Then when the agent gets assigned its cheapest tolled paths, those paths are in the SO solution and the solution is UE as well. The solution is UE because the offered path costs are the cheapest option according to the information available for the agent. We succeeded in finding an approach to obtain both an SO and a UE solution on an STN.

For further research, we propose to take due dates into account. When we do this, it can be the case that orders will arrive too late compared to this due date. We then need to add a penalty function to the cost objective function in order to minimise the number of orders arriving too late. With the tolls, it is possible to share the penalty costs by all orders who are causing the lateness of the delayed orders. Another aspect that should be looked at is fairness of the UE solution. In the presented approach a UE is found and the benefit of the SO is shared between the agents. We do not know, however, whether this sharing is done in the fairest way. This will be the topic of the next chapter.

References

1. Bar-Gera, H. (1999). *Origin-based algorithms for transportation network modeling*. Ph.D. thesis, Chicago: University of Illinois at Chicago.
2. Bar-Gera, H., Boyce, D., & Nie, Y.M. (2012). User-equilibrium route flows and the condition of proportionality. *Transportation Research Part B: Methodological*, 46(3), 440–462.
3. Corman, F., Viti, F., & Negenborn, R. R. (2017). Equilibrium models in multimodal container transport systems. *Flexible Services and Manufacturing Journal*, 29(1), 125–153.
4. Didi-Biha, M., Marcotte, P., & Savard, G. (2006). Path-based formulations of a bilevel toll setting problem. In *Optimization with Multivalued Mappings* (pp. 29–50). Springer
5. van Essen, M., Thomas, T., van Berkum, E., & Chorus, C. (2016). From user equilibrium to system optimum: a literature review on the role of travel information, bounded rationality and non-selfish behaviour at the network and individual levels. *Transport Reviews*, 36(4), 527–548.
6. Florian, M., & Hearn, D. (2003). Network equilibrium and pricing. In *Handbook of Transportation Science* (pp. 373–411). Springer
7. Han, D., & Yang, H. (2008). The multi-class, multi-criterion traffic equilibrium and the efficiency of congestion pricing. *Transportation Research Part E: Logistics and Transportation Review*, 44(5), 753–773.
8. Hearn, D., & Ramana, M. (1998). Solving congestion toll pricing models. In P. Marcotte, & S. Nguyen (Eds.), *Equilibrium and Advanced Transportation Modelling*. Centre for Research on Transportation.
9. Jiang, L., & Mahmassani, H. (2013). Toll pricing: Computational tests for capturing heterogeneity of user preferences. *Transportation Research Record: Journal of the Transportation Research Board*, 2343(1), 105–115.
10. Levy, N., Klein, I., & Ben-Elia, E. (2016). Emergence of cooperation and a fair system optimum in road networks: A game-theoretic and agent-based modelling approach. *Research in Transportation Economic*, 68, 46–55.
11. Liu, C. L., & Liu, F. (2012). Dynamical consensus seeking of second-order multi-agent systems based on delayed state compensation. *Systems & Control Letters*, 61(12), 1235–1241.
12. Miyagi, T., & Peque, G. C. (2012). Informed-user algorithms that converge to Nash equilibrium in traffic games. *Procedia—Social and Behavioral Sciences*, 54, 438–449. <https://doi.org/10.1016/j.sbspro.2012.09.762>. Proceedings of EWGT2012—15th Meeting of the EURO Working Group on Transportation, September 2012, Paris.
13. Peeta, S., & Mahmassani, H. S. (1995). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, 60(1), 81–113.
14. Ren, W., & Beard, R. W. (2005). Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5), 655–661.
15. Roughgarden, T. (2006). Selfish routing and the price of anarchy. OPTIMA-2007.

16. Wagner, N. (2012). *The dynamic user equilibrium on a transport network: Mathematical properties and economic applications*. Ph.D. thesis, Université Paris-Est.
17. Wang, C., & Tang, Y. (2017). The discussion of system optimism and user equilibrium in traffic assignment with the perspective of game theory. *Transportation Research Procedia*, 25, 2974–2983.
18. Xu, W., Miao, L., & Lin, W. H. (2012). Stochastic user equilibrium assignment in schedule-based transit networks with capacity constraints. In *Discrete Dynamics in Nature and Society*
19. Yang, H., & Huang, H. J. (2005). Fundamentals of user-equilibrium problems. In *Mathematical and Economic Theory of Road Pricing* (pp. 13–46). Amsterdam: Elsevier.
20. Yang, H., & Huang, H. J. (2005). Social and spatial equities and revenue redistribution. In *Mathematical and Economic Theory of Road Pricing* (pp. 203–238). Amsterdam: Elsevier.
21. Yang, H., & Zhang, X. (2008). Existence of anonymous link tolls for system optimum on networks with mixed equilibrium behaviors. *Transportation Research Part B: Methodological*, 42(2), 99–112 .
22. Roughgarden, T., & Tardos, E. (2002). How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2), 236–259.
23. Swamy, C. (2007). The effectiveness of stackelberg strategies and tolls for network congestion games. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1133–1142). Society for Industrial and Applied Mathematics.

Chapter 12

Fair User Equilibrium in a Transportation Space-Time Network



L. A. M. Bruijns

Abstract Central in this chapter is a transportation network, in which containers are transported for multiple agents. This network is modelled by a Space-Time Network, in which the travel time of modalities is fixed and independent of the occupancy of the network. To find the best allocation of containers to paths in this network, a flow problem can be solved. The System Optimal solution found then is the solution in which the total costs of the network are minimised. This paper introduces the idea of a fair User Equilibrium solution in such problem. The proposed approach changes the network, using a toll scheme, such that the fair User Equilibrium Solution in this changed network equals the System Optimal solution in the original network. This can be used to fairly redistribute the cost of the network among the users.

Introduction

In the previous chapter we showed how a UE solution can be created in an STN, where the UE solution is equal to the SO solution. The idea here is that optimising the total network and then sharing the benefits from an overall optimal solution between all agents is beneficial for all. This sharing was done by defining tolls on paths in the STN network. This approach is partly based on the solution algorithms used by Hearn and Ramana [2] and Jiang and Mahmassani [3]. In general, tolls can be assigned to orders and paths in a STN. Assigning tolls to the orders occurs after obtaining the SO solution, to create a UE solution in which costs are divided over the orders. Assigning tolls to paths occurs after obtaining the SO solution and creates tolled path costs on the STN, in which the paths of the SO solution have cheapest (tolled) paths costs and thus a UE solution is obtained.

L. A. M. Bruijns (✉)
TNO, The Hague, The Netherlands

In a system where all agents make their own decisions, this approach might look very theoretical. However, in practice this approach can be helpful for a LSP in organising and pricing its system and services. The LSP has a system with bottlenecks and priorities certain orders and clients. How can he do that, without having clients complaining, if they would have total knowledge about the system, about their service and pricing in comparison with other clients. Creating a UE would partially solve this, the clients cannot find a better solution themselves, however, it would be even more interesting to have an approach to find a *fair* UE solution: their realised service level might be lower, but they get a fair compensation for that. In this paper the fairness of the UE solution in a logistic STN is defined for the first time and a method is presented to find a fair UE in an order-based STN and how to translate this to a fair UE solution in a path-based STN.

In the next section, the general STN is defined and a definition for fairness is proposed, when creating a User Equilibrium in an STN. Fairness will be defined for a group of agents or flows over the STN that share a bottleneck link in their shortest or cheapest path. For this, in Sect. “[Finding Connected Components in STN](#)”, an algorithm is presented to find connected components in an STN. With these connected components, the fair order-based toll scheme will be created in Sect. “[Tolls on Orders](#)” and the existence of a toll set that realises the fair redistribution is proved. Next, in Sect. “[Path Tolls Based on Order Fairness](#)”, the fair path-based toll scheme is created and again the existence of a solution is proved. We end with some conclusions and ideas for future research.

Fair User Equilibrium in STN

In this chapter, a way to construct a fair User Equilibrium within an STN is proposed. All used notation can be found in Table 11.1. The Space-Time Network that represents the transportation network is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. Each arc $a \in \mathcal{A}$ is a link between two nodes, an origin node v_1 and an end node v_2 , both in \mathcal{V} : $a = (v_1, v_2)$, along which a transportation unit, for example, a container, can travel. The variable x_a is used to denote the number of units of flow along arc a . An Origin-Destination-pair (OD-pair) or order w is a pair of two nodes, origin location w_O and destination location w_D , so $w = (w_O, w_D)$, which is not necessarily an arc. The number of containers that an order wants to transport from w_O to w_D is denoted by d_w , the demand of order w . A path p consists of a sequence of (non-horizontal) adjacent arcs between two nodes. Here only paths between origin and destination nodes are considered. f_p denotes the path flow of path p (always integer), with $p \in \mathcal{P}_w$, $w \in \mathcal{W}$, where \mathcal{P}_w is the set of all paths for OD-pair w and \mathcal{W} is the set of all OD-pairs/orders. The total path set is $\mathcal{P} := \bigcup_{w \in \mathcal{W}} \mathcal{P}_w$. The costs of an arc a are denoted by c_a and the path costs of path p are denoted by C_w^p or C^p . The capacity of an arc is denoted by m_a and the capacity of a path is denoted by m_p . The available

arcs in a path are denoted by

$$\delta_{ap} = \begin{cases} 1 & \text{if } a \text{ is contained in } p, \forall a \in \mathcal{A}, p \in \mathcal{P}, \\ 0 & \text{otherwise.} \end{cases}$$

For each order w there may be multiple paths to travel by within the STN. The definition of the previous chapter for a UE within an STN is used again:

Definition User Equilibrium in a transportation STN: A UE is reached when each agent can use their cheapest paths.

Next, the concept of fairness is added to this User Equilibrium. For this the cost ratio

$$r_w = \frac{C_w}{k_w}$$

is used to calculate how much the cost of the path in the solution for order w (C_w) is, compared to the cheapest costs he could have paid when he would be the only order on the network (k_w). Also the tolled cost ratio

$$r_{\beta w} = \frac{C_{\beta w}}{k_w} = \frac{C_w + \beta_w}{k_w}$$

is introduced, which denotes the ratio of a tolled solution, where β_w denotes the toll paid by order w .

Definition Fair User Equilibrium in a transportation STN: A fair UE is reached when each agent can use their cheapest paths and all agents that would use some same bottleneck arc in the original STN, have the same tolled cost ratio.

The overall approach to find the tolls is equal to the approach in the previous chapter. It starts with finding an SO solution, which results in the optimal flows \underline{f}_p . Now, define the set of paths used in the SO solution by $h_{in,w} := \{p \mid \underline{f}_p > 0, p \in \mathcal{P}_w\}$, and the sets of all other paths (which are not in the SO solution) by $h_{out,w} := \{p \mid \underline{f}_p = 0, p \in \mathcal{P}_w\}$. Then an new Problem NP- β can be solved that consists of an objective function that minimises the path tolls of a certain path set under a set of constraints.

Finding Connected Components in STN

For future use, it is important to know which agents use, in their shortest or cheapest path, the same (congested) connection or which orders share a bottleneck link in the original STN. This means that all connected components in the STN have to

be found. The proposed method is presented in Algorithm 1. There, the connected components are found by constructing a graph G , consisting of orders $w \in \mathcal{W}$, where two orders in this graph can share an arc when those orders contain a joint bottleneck. To obtain these arcs, the cheapest paths per order are found (in Step 4). For each arc a , check if this arc is a bottleneck, that is when the amount of cheapest paths on that arc is higher than the capacity of that arc (this happens in Step 6), assuming every order travels via its cheapest path. If in this step bottlenecks are found and bottleneck sets are created, in which paths (bottleneck sets η_a) and orders (bottleneck sets Γ_a) are listed per bottleneck arc a . Also an arc in G is added between each pair of orders that are in the same bottleneck set. Then, for each cheapest path of an order that is in the SO solution, this path is fixed (so state that this path is taken by that order) and then recalculate the cheapest paths for all remaining orders (Step 15). If new bottlenecks arise, arcs are added between each pair of orders in this bottleneck set to the graph G . The fixing process is iterated until no new bottlenecks arise. Then in Step 37 the connected components are found in graph G . This set can be used to compare only the (tolled) cost ratio of orders that are in the same connected component, and to make sure that for each connected component the tolls sum up to zero such that orders only pay/receive for bottlenecks that influence the route choice for them.

Tolls on Orders

In this section, a method is presented to find tolls on orders to create a fair UE. Next, the existence of a (unique) solution will be proven.

Finding a User Equilibrium

Here, the goal is to find a fair order-tolled User Equilibrium. The same structure as presented in the previous chapter for path tolled UE, is used here to obtain fair order tolls. First, the SO problem is solved in an STN. Second, the costs are adjusted by adding tolls. Note that here tolls are assigned to the total costs of orders, and thus the value of the tolls does not influence the path choices of customers. The tolls assigned to orders will make a fair redistribution of the costs of all orders using the network.

The way of finding tolls that provide a UE solution in an initial SO problem is described in Algorithm 2. First, the SO is defined and solved in step 1 and 2 in Algorithm 2. In step 3 and 4, the alternative problem is formulated, minimising the differences in cost ratio within connected components, and solved to get a fair payment regulation and a UE is reached. The sum of all tolls paid and received by all customers within a connected component has to be zero to ensure the total system has no profit or loss. The problem in step 3 can be linearised easily. Adding the tolls

Algorithm 1 Finding connected components in STN

```

1: Let  $\mathcal{Q}_w = \emptyset \forall w \in \mathcal{W}$  be the set of cheapest paths per order  $w$ ,  $\mathcal{A}_\eta = \emptyset$  the bottleneck set.
2: Create graph  $G$  with nodes  $V = \mathcal{W}$  and arc set  $E = \emptyset$ .
3: for  $w \in \mathcal{W}$  do
4:   Find cheapest paths  $q_{w,n}$ ,  $n \in \{1, \dots, d_w\}$ ,  $q_{w,n} \in \mathcal{P}_w$ .
5:   Add paths  $q_{w,n}$  to  $\mathcal{Q}_w$ .
6: for  $a \in \mathcal{A}$  do
7:   if  $o_a > m_a$  ( $o_a$  the occupancy on arc  $a$ ) then
8:     Create bottleneck set:  $\eta_a = \{q_{w,n} \mid \delta_{aq_{w,n}} = 1, q_{w,n} \in \mathcal{Q}_w, \forall w \in \mathcal{W}\}$ .
9:     Create order set:  $\Gamma_a = \{w \mid q_{w,n} \in \eta_a, q_{w,n} \in \mathcal{Q}_w\}$ .
10:     $\mathcal{A}_\eta := \mathcal{A}_\eta \cup a$ .
11:    for  $w_1, w_2 \in \Gamma_a$  do
12:      Add  $e = (w_1, w_2)$  to  $E$ .
13:   else
14:      $\eta_a = \emptyset$ .
15: for  $a_1 \in \mathcal{A}_\eta$  do
16:   for  $q_{w,n} \in \eta_{a_1}$  and  $q_{w,n} \in \bigcup_{w \in \mathcal{W}} h_{in,w}$  do
17:     Fix  $q_{w,n}$ : update  $o_a = \delta_{aq_{w,n}} f_{q_{w,n}} \forall a \in \mathcal{A}$ .
18:     Recalculate the cheapest paths for remaining orders and update  $\mathcal{Q}_w$ .
19:     Define  $l = (q_{w,n})$  (a list of all previous fixed paths) and  $\mathcal{A}_\eta^l = \emptyset$ .
20:     if  $o_{a_2} > m_{a_2}$  for  $a_2 \notin \mathcal{A}_\eta$  then
21:       Create new bottleneck set  $\eta_{a_2}$ .
22:       Create order set:  $\Gamma_{a_2} = \{w \mid q_{w,n} \in \eta_{a_2}, q_{w,n} \in \mathcal{Q}_w\}$ .
23:       Define  $\mathcal{A}_\eta^l := \mathcal{A}_\eta^l \cup a_2$ .
24:       for  $w_1, w_2 \in \Gamma_{a_2}$  do
25:         Add  $e = (w_1, w_2)$  to  $E$ .
26:     for  $a_2 \in \mathcal{A}_\eta^l$  do
27:       for  $r_{w,n} \in \eta_{a_2}$  and  $r_{w,n} \in \bigcup_{w \in \mathcal{W}} h_{in,w}$  do
28:         Fix  $r_{w,n}$ : update  $o_a = \sum_{q_{w,n} \in \text{fixed paths}} \delta_{aq_{w,n}} f_{q_{w,n}} \forall a \in \mathcal{A}$ 
29:         Recalculate the cheapest paths for remaining orders and update  $\mathcal{Q}_w$ .
30:         if  $o_{a_2} > m_{a_2}$  for  $a_2 \notin \mathcal{A}_\eta$  then
31:           Create new bottleneck set  $\eta_{a_2}$ .
32:           Create order set:  $\Gamma_{a_2} = \{w \mid q_{w,n} \in \eta_{a_2}, q_{w,n} \in \mathcal{Q}_w\}$ .
33:            $l := (l, q_{w,n})$ .
34:           Define  $\mathcal{A}_\eta^l := \mathcal{A}_\eta^l \cup a_2$ .
35:           for  $w_1, w_2 \in \Gamma_{a_2}$  do
36:             Add  $e = (w_1, w_2)$  to  $E$ .
37:    $s = 1$ .
38:    $\mathcal{V}_s$  is the set of visited nodes in connected component  $s$ .
39:   for  $w_1 \in \mathcal{V} \setminus \bigcup_{1 \leq j \leq s} \mathcal{V}_j$  do
40:      $\mathcal{V}_s := \{w_1\}$ .
41:     for  $w_2 \in \bigcup_{v \in \mathcal{V}_1} N_G(v) \setminus \bigcup_{1 \leq j \leq s} \mathcal{V}_j$  do
42:        $\mathcal{V}_s := \mathcal{V}_s \cup w_2$ .
43:      $s := s + 1$ .
44:    $k := s - 1$  are the number of connected components of  $G$ .
45:    $\mathcal{V}_s$  are the connected components of graph  $G$ .

```

Algorithm 2 Calculating order tolls

1: Create SO problem:

$$\begin{aligned}
 & \min \sum_{p \in \mathcal{P}} C_w^p f_p \\
 & \text{s.t. } x_a = \sum_{p \in \mathcal{P}} \delta_{ap} f_p \quad \forall a \in \mathcal{A} \\
 & \quad \sum_{p \in \mathcal{P}_w} f_p = d_w \quad \forall w \in \mathcal{W} \\
 & \quad x_a \leq m_a \quad \forall a \in \mathcal{A} \\
 & \quad f_p \in \mathbb{N}_0 \quad \forall p \in \mathcal{P} \\
 & \quad x_a \in \mathbb{N}_0 \quad \forall a \in \mathcal{A}
 \end{aligned}$$

2: Solve SO problem, output: $C_w = \sum_{p \in \mathcal{P}_w} C_w^p f_p \quad \forall w \in \mathcal{W}$.

3: Create the nonlinear programming problem $\overline{\text{NP}}\text{-}\beta$ (minimising over the absolute value of the difference of ratios for all pairs of orders in the same set of connected components):

$$\begin{aligned}
 & \min \sum_{s=1}^k \sum_{w_1, w_2 \in \mathcal{V}_s} \left| \frac{C_{w_1} + \beta_{w_1}}{k_{w_1}} - \frac{C_{w_2} + \beta_{w_2}}{k_{w_2}} \right| \\
 & \text{s.t. } \sum_{w \in \mathcal{W}} \beta_w = 0 \\
 & \quad \sum_{w \in \mathcal{V}_s} \beta_w = 0 \quad \forall 1 \leq s \leq k \\
 & \quad \beta_w \geq -C_w \quad \forall w \in \mathcal{W}
 \end{aligned}$$

where the objective function minimises the ratio differences for all pairs of orders. Note that we sum twice over the set of orders: when $w_1 = w_2$, that term of the objective becomes zero. The constraints ensures the total toll sum to be zero, and that no path can have a negative C_{β_w} cost.

4: Solve $\overline{\text{NP}}\text{-}\beta$, output: β_w .

5: Add tolls β_w to the SO problem, $\text{SO}\text{-}\beta$:

$$\begin{aligned}
 & \min \sum_{w \in \mathcal{W}} \left(\sum_{p \in \mathcal{P}_w} C_w^p f_p + \beta_w \right) \\
 & \text{s.t. } x_a = \sum_{p \in \mathcal{P}} \delta_{ap} f_p \quad \forall a \in \mathcal{A} \\
 & \quad \sum_{p \in \mathcal{P}_w} f_p = d_w \quad \forall w \in \mathcal{W} \\
 & \quad x_a \leq m_a \quad \forall a \in \mathcal{A} \\
 & \quad f_p \in \mathbb{N}_0 \quad \forall p \in \mathcal{P} \\
 & \quad x_a \in \mathbb{N}_0 \quad \forall a \in \mathcal{A}
 \end{aligned}$$

6: Solve $\text{SO}\text{-}\beta$ problem, output path flow vector \underline{f} .

to the SO problem (step 5 and 6) defines a new SO problem, which solution (in flows) should be the same as the original SO problem.

Existence of Solutions

Now the question arises whether every NP- β of Algorithm 2 has a (unique) solution. For each component \mathcal{V}_s , there are $\frac{1}{2}|\mathcal{V}_s|(|\mathcal{V}_s| - 1)$ terms in the objective, and we can state that the best solution possible has objective value 0, meaning that each term has to be equal to zero, leading to:

$$k_i\beta_j - k_j\beta_i = k_jC_i - k_iC_j. \quad (12.1)$$

A generalisation can be made of the optimal solution for tolls on orders: Given the problem with m orders, the linear system can be created $A\boldsymbol{\beta} = \mathbf{b}$ with A a block diagonal matrix:

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_m. \end{bmatrix}$$

So, A consists of m block matrices, one for each component $\mathcal{V}_s = \{1, \dots, n\} \forall 1 \leq s \leq m$, with $n = |\mathcal{V}_s|$ the number of orders in \mathcal{V}_s :

$$A_i = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_{n-1} \\ \mathbf{1}_{1,n} \end{bmatrix} \quad \text{with } D_j = \begin{bmatrix} \mathbf{0}_{1,j-1} & k_{j+1} & -k_j & 0 & \cdots & 0 \\ \mathbf{0}_{1,j-1} & k_{j+2} & 0 & -k_j & \ddots & \vdots \\ \mathbf{0}_{1,j-1} & \vdots & \vdots & \ddots & \ddots & 0 \\ \mathbf{0}_{1,j-1} & k_n & 0 & \cdots & 0 & -k_j \end{bmatrix}, \quad 1 \leq j \leq n-1, \quad (12.2)$$

and $\mathbf{1}_{1,n}$ is a row vector of n ones, $\mathbf{0}_{1,j-1}$ a row vector of $j-1$ zeros and $\mathbf{0}_{1,0} := \emptyset$.

The vector \mathbf{b} consists of vectors \mathbf{b}_s for each component $\mathcal{V}_s = \{1, \dots, n\} \forall 1 \leq s \leq m$ with $n = |\mathcal{V}_s|$:

$$\mathbf{b}_i = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ 0 \end{bmatrix} \quad \text{with } z_j = \begin{bmatrix} k_j C_{j+1} - k_{j+1} C_j \\ k_j C_{j+2} - k_{j+2} C_j \\ \vdots \\ k_j C_n - k_n C_j \end{bmatrix} \quad (12.3)$$

where A is a $\sum_{s=1}^m \left(\frac{1}{2} |\mathcal{V}_s| \cdot (|\mathcal{V}_s| - 1) + 1 \right) \times |\mathcal{W}|$ matrix and \mathbf{b} has length $\sum_{s=1}^m \left(\frac{1}{2} |\mathcal{V}_s| \cdot (|\mathcal{V}_s| - 1) + 1 \right)$. To draw some conclusions from this equality $A\boldsymbol{\beta} = \mathbf{b}$, first some definitions from Linear Algebra are introduced:

Definition Row-echelon form: A matrix is in row-echelon form if:

- All zero rows have been moved to the bottom.
- The leading nonzero element (also called a pivot) in any row is farther to the right than the leading nonzero element in the row just above it.
- In each column containing a leading nonzero element, the entries below that leading nonzero element are 0.

The elementary row operations can be applied to modify the matrix until a row-echelon form is obtained.

Theorem 1 ([1, Theorem 1.7]) *Let $A\mathbf{x} = \mathbf{b}$ be a linear system, and let $[A|\mathbf{b}] \sim [H|\mathbf{c}]$, where H is in row-echelon form.*

1. *The system $A\mathbf{x} = \mathbf{b}$ is inconsistent if and only if the augmented matrix $[H|\mathbf{c}]$ has a row with all entries 0 to the left of the partition and a nonzero entry to the right of the partition.*
2. *If $A\mathbf{x} = \mathbf{b}$ is consistent and every column of H contains a pivot, the system has a unique solution.*
3. *If $A\mathbf{x} = \mathbf{b}$ is consistent and some column of H has no pivot, the system has infinitely many solutions, with as many free variables as there are pivot-free columns in H .*

The problem $A\boldsymbol{\beta} = \mathbf{b}$ can be reduced to $H\boldsymbol{\beta} = \mathbf{c}$, where H is in row-echelon form. When solving NP- β with two orders and one connected component \mathcal{V}_1 , one can easily verify that there always exists a unique solution by reducing the initial system:

$$\begin{bmatrix} [cc|c]k_2 & -k_1 & k_1C_2 - k_2C_1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} [cc|c]k_2 & -k_1 & k_1C_2 - k_2C_1 \\ 0 & k_1 + k_2 & k_2C_1 - k_1C_2 \end{bmatrix}.$$

This shows that Statement 2 of Theorem 1 holds, because $k_w > 0 \forall w \in \mathcal{W}$.

Then the solution is

$$\beta_2 = \frac{k_2C_1 - k_1C_2}{k_1 + k_2},$$

$$\beta_1 = \frac{k_1C_2 - k_2C_1}{k_2} + \frac{k_1}{k_2}\beta_2 = \frac{k_1C_2 - k_2C_1}{k_2} + \frac{k_1(k_2C_1 - k_1C_2)}{k_2(k_1 + k_2)} = \frac{k_1C_2 - k_2C_1}{k_1 + k_2}.$$

We only need to show that the constraints $\beta_w \geq -C_w$ are satisfied for all $w \in \mathcal{W}$:

$$\begin{aligned}
 -C_2 \leq \beta_2 &= \frac{k_2 C_1 - k_1 C_2}{k_1 + k_2} \iff 0 \leq \frac{k_1 (C_1 + C_2)}{k_1 + k_2} \\
 -C_1 \leq \beta_1 &= \frac{k_1 C_2 - k_2 C_1}{k_1 + k_2} \iff 0 \leq \frac{k_2 (C_1 + C_2)}{k_1 + k_2}.
 \end{aligned}$$

These constraints are always satisfied, since $k_1, k_2, C_1, C_2 \geq 0$. This problem can be generalised to a problem with n orders with m components: Suppose there is an NP- β problem with n orders, and $\mathcal{V}_i = \{1, \dots, n\}$, so $|\mathcal{V}_i| = n$ for all $1 \leq i \leq m$. This gives the problem $A\beta = b$, with A being a block matrix. Now, the existence of a unique solution for each sub-problem $A_i\beta = b_i$ is proved, and so that a unique solution exists for the original problem.

The row-echelon form consists of the first row of each matrix D_j , meaning the rows $[\mathbf{0}_{1,j-1} \ k_{j+1} \ -k_j \ \mathbf{0}_{1,n-(j+1)}] \forall 1 \leq j \leq n-1$. All other rows in the matrices D_j can be written as a linear combination of rows $[\mathbf{0}_{1,j-1} \ k_{j+1} \ -k_j \ \mathbf{0}_{1,n-(j+1)}]$, so those rows are equal to a zero row in the row-echelon form.

For block matrix A_i and corresponding vector b_i , the following row-echelon form $[H_i|c_i]$ can be obtained:

$$H_i = \begin{bmatrix} k_2 - k_1 & 0 & \dots & \dots & \dots & 0 \\ 0 & k_3 & -k_2 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & k_4 & -k_3 & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 & k_{n-1} & -k_{n-2} & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & k_n & -k_{n-1} \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & -\sum_{i=1}^n k_i \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 \end{bmatrix}, \quad c_i = \begin{bmatrix} k_1 C_2 - k_2 C_1 \\ k_2 C_3 - k_3 C_2 \\ k_3 C_4 - k_4 C_3 \\ \vdots \\ k_{n-2} C_{n-1} - k_{n-1} C_{n-2} \\ k_{n-1} C_n - k_n C_{n-1} \\ \sum_{j=1}^{n-1} k_j C_n - \sum_{j=1}^{n-1} k_n C_j \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Concluding, using Theorem 1 shows that for each toll problem with one connected component, there is a unique toll solution, because each column in H_i contains a pivot. Then, it follows that each row in block diagonal matrix H has a pivot in each column, and for the zero rows, c contain a zero in that row number, so together with the inequality constraints $\beta_w \geq -C_w \forall w \in \mathcal{W}$ are satisfied, Theorem 1 holds. It can be concluded that if a solution exists for a toll order problem given the inequality constraints, the equality constraints provide a unique solution.

Path Tolls Based on Order Fairness

In the previous chapter it was shown how path tolls can be found to create new tolled costs for the STN, in which the customers can choose their own travelling paths, and the path costs are constructed in such a way that they will choose paths that minimise the total cost of the network (SO) and are the cheapest paths for themselves (UE). It is shown how the realised costs can be redivided fairly over all orders, such that the use of the network is paid by all customers, and no customer is more harmed regarding costs than others, when they are using the same part of the network. When adding tolls to paths, as in the previous chapter, it may feel like the customers are being misled, because the tolled path costs can differ a lot from the initial path costs, when taking the costs of the original networks into account. For this, the constraints of fairness, as used in the objective function of order tolls, can be coupled to make the path-based solutions a more fair User Equilibrium, by dividing the costs over all orders in the network, instead of assign higher path costs to certain orders only. In this section the procedure is explained.

Finding a User Equilibrium

We start with the SO problem as presented in the previous chapter. To find path tolls, first the SO problem is solved, as presented step 1 of Algorithm 2. With the SO solution found, the next step is calculating the path tolls. Therefore the NP- β of the previous chapter is used, but with the extra constraints ($\forall w_1, w_2 \in \mathcal{V}_s, \forall s \in \{1, \dots, k\}$):

$$\begin{aligned} r\beta_{w_1} = r\beta_{w_2} &\iff \frac{C_{w_1} + \beta_{w_1}}{k_{w_1}} = \frac{C_{w_2} + \beta_{w_2}}{k_{w_2}} \\ &\iff \frac{\sum_{p \in \mathcal{P}_{w_1}} (C_{w_1}^p + \beta_{w_1}^p) f_p}{k_{w_1}} = \frac{\sum_{p \in \mathcal{P}_{w_2}} (C_{w_2}^p + \beta_{w_2}^p) f_p}{k_{w_2}}, \end{aligned}$$

with order sets \mathcal{V}_s as in Sect. “Fair User Equilibrium in STN”. Again, the objective can be linearised easily.

Existence of Solutions

The question arises again whether a combined toll solution can be obtained. For the inequality constraints always a valid solution can be found:

$$\begin{aligned} \beta_w^i &\geq -C_w^i \quad \forall i \in \mathcal{P}_w \\ \beta_w^i - \beta_w^j &\leq C_w^j - C_w^i \quad \forall (i, j), i \in h_{in,w}, j \in h_{out,w}. \end{aligned}$$

The first set of constraints provides a lower bound for all path tolls, and the last set of constraints gives an upper bound for all paths tolls for paths $p \in \bigcup_{w \in \mathcal{W}} h_{in,w}$, the path tolls for the other paths are unbounded. We can always find a toll vector β that satisfies those inequality constraints. If $C_w^j - C_w^i < 0$ for some (i, j) , $i \in h_{in,w}$, $j \in h_{out,w}$, set $\beta_w^j := C_w^i - C_w^j$ for all those paths j and $\beta_w^p := 0$ for all other paths. This shows the solution space is non-empty.

Now, it has to be investigated if the equality constraints provide a valid solution in combination with the inequality constraints. The equality constraints are:

$$\sum_{w \in \mathcal{W}} \sum_{p \in h_{in,w}} \beta_w^p = 0 \quad (12.4)$$

$$\sum_{w \in \mathcal{V}_s} \sum_{p \in h_{in,w}} \beta_w^p = 0 \quad \forall s \in \{1, \dots, k\} \quad (12.5)$$

$$\frac{\sum_{p \in \mathcal{P}_{w_1}} (C_{w_1}^p + \beta_{w_1}^p) \underline{f}_p}{k_{w_1}} = \frac{\sum_{p \in \mathcal{P}_{w_2}} (C_{w_2}^p + \beta_{w_2}^p) \underline{f}_p}{k_{w_2}} \quad \forall w_1, w_2 \in \mathcal{V}_s, \quad \forall s \in \{1, \dots, k\}. \quad (12.6)$$

Constraint (12.4) is superfluous, because it is equal to summing up Constraints (12.5):

$$\sum_{w \in \mathcal{W}} \sum_{p \in h_{in,w}} \beta_w^p = 0 \iff \sum_{s=1}^k \sum_{w \in \mathcal{V}_s} \sum_{p \in h_{in,w}} \beta_w^p = 0.$$

If $d_w = 1 \quad \forall w \in \mathcal{W}$, this problem corresponds to finding solutions in . If $d_w \geq 1 \quad \forall w \in \mathcal{W}$, constraint (12.6) can be rewritten for a pair of two orders i and j , $i, j \in \mathcal{W}$ (we assume $h_{in,i} = \{p_1, \dots, p_k\}$, $h_{in,j} = \{q_1, \dots, q_l\}$):

$$\begin{aligned} \frac{C_i + \beta_i}{k_i} = \frac{C_j + \beta_j}{k_j} &\iff k_j \beta_i - k_i \beta_j = k_i C_j - k_j C_i \\ \iff k_j \sum_{p \in \mathcal{P}_i} \beta_i^p \underline{f}_p - k_i \sum_{p \in \mathcal{P}_j} \beta_j^p \underline{f}_p &= k_i \sum_{p \in \mathcal{P}_j} C_j^p \underline{f}_p - k_j \sum_{p \in \mathcal{P}_i} C_i^p \underline{f}_p. \end{aligned}$$

The tolls can be calculated per connected component.

- If $\mathcal{V}_s = \{1\}$, then constraint (12.5) states: $\sum_{p \in h_{in,1}} \beta_1^p \underline{f}_p = 0$. So no tolls are added to the STN, because this order can take its cheapest paths, because it does not use any bottleneck arcs, which indicates there are no issues in travelling via its cheapest paths.

- If $\mathcal{V}_s = \{1, 2\}$, then constraint (12.5) states :

$$\beta_i + \beta_j = 0 \iff \beta_i = -\beta_j \iff \sum_{p \in \mathcal{P}_i} \beta_i^p \underline{f}_p = - \sum_{p \in \mathcal{P}_j} \beta_j^p \underline{f}_p.$$

We can use this to continue our rewriting of constraint (12.6):

$$\begin{aligned} (k_i + k_j) \sum_{p \in \mathcal{P}_i} \beta_i^p \underline{f}_p &= k_i \sum_{p \in \mathcal{P}_j} C_j^p \underline{f}_p - k_j \sum_{p \in \mathcal{P}_i} C_i^p \underline{f}_p \\ \iff \sum_{p \in \mathcal{P}_i} \beta_i^p \underline{f}_p &= \frac{k_i \sum_{p \in \mathcal{P}_j} C_j^p \underline{f}_p - k_j \sum_{p \in \mathcal{P}_i} C_i^p \underline{f}_p}{k_i + k_j}. \end{aligned}$$

The equality always has a solution because $k_i + k_j \neq 0$.

- If $\mathcal{V}_s = \{1, 2, 3\}$, then based on (12.5):

$$\sum_{p \in \mathcal{P}_1} \beta_1^p \underline{f}_p + \sum_{p \in \mathcal{P}_2} \beta_1^p \underline{f}_p + \sum_{p \in \mathcal{P}_3} \beta_3^p \underline{f}_p = 0,$$

and (12.6):

$$\begin{aligned} k_2 \sum_{p \in \mathcal{P}_1} \beta_1^p \underline{f}_p - k_1 \sum_{p \in \mathcal{P}_2} \beta_2^p \underline{f}_p &= k_1 \sum_{p \in \mathcal{P}_2} C_2^p \underline{f}_p - k_2 \sum_{p \in \mathcal{P}_1} C_1^p \underline{f}_p \\ k_3 \sum_{p \in \mathcal{P}_1} \beta_1^p \underline{f}_p - k_1 \sum_{p \in \mathcal{P}_3} \beta_3^p \underline{f}_p &= k_1 \sum_{p \in \mathcal{P}_3} C_3^p \underline{f}_p - k_3 \sum_{p \in \mathcal{P}_1} C_1^p \underline{f}_p \\ k_3 \sum_{p \in \mathcal{P}_2} \beta_2^p \underline{f}_p - k_2 \sum_{p \in \mathcal{P}_3} \beta_3^p \underline{f}_p &= k_2 \sum_{p \in \mathcal{P}_3} C_3^p \underline{f}_p - k_3 \sum_{p \in \mathcal{P}_2} C_2^p \underline{f}_p. \end{aligned}$$

This problem corresponds to the order toll problem, where the constraints, as just observed, are contained in the objective function (see (12.1)), and so a solution exist for each problem (see). In this case there are infinitely many solutions, because the row-echelon form of the problem satisfies Statement 3 of Theorem 1. We will show this, but first introduce some extra notation: with x_{j-1} we denote the number of paths for orders 1 up until $j - 1$: $x_j = \left| \bigcup_{1 \leq i \leq j} \mathcal{P}_i \right|$, and we denote path flow found in the SO solution by f_w^p instead of \underline{f}_p . We assume $l = |\mathcal{P}_j| \forall j \in \mathcal{W}$.

and the vector

$$c_i = \begin{bmatrix} k_1 C_2 - k_2 C_1 \\ k_2 C_3 - k_3 C_2 \\ \vdots \\ k_{n-1} C_n - k_n C_{n-1} \\ \sum_{j=1}^{n-1} k_j C_n - \sum_{j=1}^{n-1} k_n C_j \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The number of pivots in H_i equals the number of orders, which is less than the number of path toll variables, so according to Statement 3 of Theorem 1: if a solution exists, there are infinitely many solutions for this toll problem.

Conclusions and Future Research

The goal of this work was to provide a method to obtain a fair User Equilibrium solution in a Space-Time Network used for representation and optimisation of a logistic network. As extra constraint we wanted the solution to be both System Optimal as well as User Equilibrium.

In order to provide an SO solution in which a UE is reached as well, we applied tolls to the network costs. But in order to obtain a User Equilibrium, we need to define when a User Equilibrium is reached. For order tolls, we say a UE is reached when all extra costs (compared to the cheapest path costs) made in the network are divided over the orders in a fair way, concerning the ratio of the paid costs compared to the cheapest path cost. For fair path tolls, we say a UE is reached when all orders can travel via their cheapest paths, and the extra costs in the network are divided in a fair way over the orders.

The first step in all toll algorithms is to calculate the SO solution based on the path costs of orders travelling from their origin to their destination. The next step is to calculate tolls that are added to the path or order costs, depending on what kind of tolls we considered. We succeeded in finding an approach to obtain both an SO and a UE solution on an STN. For order tolls we showed there always exists a unique toll solution and for fair path tolls there always exist infinitely many solutions.

For future research we recommend to look at the scalability and computational effectiveness of the proposed methodology. As indicated, the methodology can be used by a LSP to divide the capacity and to price the services in a fair way. We propose to bring this methodology in practice in such a case and perform a case study.

References

1. Fraleigh, J., Beauregard, R., & Katz, V. (1995). Linear Algebra. In *No. v. 1 in Featured Titles for Linear Algebra*. Addison-Wesley.
2. Hearn, D., & Ramana, M. (1998). Solving congestion toll pricing models. In Marcotte P., & Nguyen S. (Eds.). *Equilibrium and advanced transportation modelling*. Centre for Research on Transportation.
3. Jiang, L., & Mahmassani, H. (2013). Toll pricing: Computational tests for capturing heterogeneity of user preferences. *Transportation Research Record: Journal of the Transportation Research Board*, 2343(1), 105–115.

Part IV

Applications

In this last part, we will look at four special use cases. Chapter 13 presents a combined schedule and container assignment in a network design problem under uncertainty. This case has the special feature that the planner has to obtain slots at each terminal the barge has to visit. However, the confirmation of these time slots is late in time and may deviate from the requested ones, which brings uncertainty in the scheduling process.

Chapter 14 presents the problem of a 2-stage delivery chain with time windows. An arrival has to be in a certain time interval, at the expense of waiting time or penalties if the time limits are exceeded. This chapter looks at the optimal placement of those time intervals in a specific case of a barge visiting two ports in sequence. For the second port, a possible delay or penalty should be incorporated.

Chapter 15 looks at the problem of an LSP controlling the means of transport and responsible for the assignment of orders to these means. Here the demand, i.e., the arrival of orders, is uncertain until the moment of unveiling. The combination of uncertainty and the huge number of controllable items make the problem difficult to solve. That is why it is proposed in this chapter to split the problem in two parts, first creating a general schedule and thereafter assigning containers to trains and barges following the schedule or assign them to a truck.

Finally, Chap. 16 addresses synchromodal planning at operational level from the perspective of a logistics service provider and studies an optimisation problem with simultaneous vehicle routing and container-to-mode assignment, having uncertain data. This problem belongs to the fourth quadrant of Fig. 1.4. Here, a robust formulation is proposed to eliminate the uncertain parameters from the objective function and constraints.

Chapter 13

Simulation Approach for Container Assignment under Uncertainty



W. J. de Koning

Abstract In this chapter an online optimisation approach is proposed which can be used to find an appropriate combined schedule and container assignment in a Network Design Problem under uncertainty. For this, a simulation based approach on a multi-period time window is proposed, moving forward on the time window after each decision made, assuming that the status of the system is updated as soon as the stochastic and unknown elements become deterministic and known. This approach provides new insight and knowledge into synchronodal and multimodal planning problems. The results of the approach are compared to the results of three simpler online optimisation methods and to the solution of the offline approach where all information is known.

Introduction

In recent years, a remarkable growth is noticeable in the number of containers that have to be transported from one place to another by different kinds of resources, e.g., trucks, trains and barges. A set of these resources linked together with the purpose of transporting freight (or people) from one place to another is called a logistics network. A logistics network is usually run by a logistics service provider (LSP), who faces the problem of delivering the right amount of freight in the right place at the right time. Due to the ever-growing complexity of these networks, an LSP needs efficient tools to support his decisions in order to strive for the optimal network performance at minimal cost [9]. The decision making process could be classified into three levels: strategic, tactical and operational [4, 16]. The operational level is concerned with short term decisions that need to be made by local management. The most important operational decisions relate to scheduling the transport and maintenance services, routing and dispatching of vehicles and allocating freight to transport modes.

W. J. de Koning (✉)
TNO, The Hague, The Netherlands

In this work we look at the challenge faced by a Dutch LSP, responsible for the transportation of containers from the eastern part of the Netherlands to the port of Rotterdam and vice versa. Every day, multiple barges depart from the single inland terminal in the east to different deep-sea terminals within the port of Rotterdam. The orders arrive randomly in time at the planner; this being the first source of uncertainty. At each decision moment, a planner has to decide which containers to allocate to which barges and/or trucks. This decision has to be made in such a way that the network performance of the Dutch LSP is optimised over time. The planner needs to obtain a slot at each terminal the barge has to visit. The terminals within the Rotterdam region are controlled by other agents, who confirm the requested calls from the LSP with a delay of approximately half a day. Most of the time, these confirmed slots may deviate from the requested ones, giving the second source of uncertainty. This means that we have two elements of uncertainty in this problem: the requested appointment times that have to be confirmed and the orders (of containers to be shipped) that have not been announced yet. This online assignment problem under uncertainty falls under synchromodal transportation problems. The framework of Chap. 2 would summarise the problem as:

$$\overline{R}, [RD], [RDT] \mid \overline{D}, [D2R], \widehat{DRD}, \widehat{DDD} \mid \text{selfish}(1+) \mid \text{isolated.}$$

This problem has not been studied earlier. However, there exists work on related problems. The work of Rivera and Mes [13, 15] also addresses future assignments. The authors formulate the container-to-mode assignment problem as a Markov Decision Process (MDP) and approximate the solution by means of Approximate Dynamic Programming (ADP). A future simulation approach in logistics can be found in [11], which is based on the general approach presented in [12, 14]. To account for uncertain release times of the containers a simulation algorithm is used when making online decisions whether to assign a container to a barge or not. The general underlying problems are the Multi-Commodity Network Design Problem (MCNDP) [5, 7, 10] and Multi-Commodity Minimum Cost Flow Problem (MCMCFP) [3], which disregard the stochastic nature of some of the problem's elements. The paper by Fragkos et al. [6] generalises the basic MCNDP to a multi-period setting, where demand for each commodity expands dynamically over a discrete time window. In the paper of Han et al. [8] a robust scenario approach is presented for the vehicle routing problem (VRP) with uncertain travel times. The work of Chiscop [2] also proposes a robust formulation, to be able to do simultaneous vehicle routing and container-to-mode assignment including uncertainty in the release times.

In this chapter, in order to address the presence of uncertainty, a multi-period time window (MPTW) approach is introduced. This extends the traditional multi-commodity network design problems by introducing a multi-period time window setting. Although such multi-period network design problems can provide useful input for strategic and tactical decisions, finding their optimal solution is computationally challenging. Our approach solves iteratively the planning problem,

using a simulation approach, by moving forward on the MPTW after each decision made, assuming that the status of the system is updated as soon as the stochastic and unknown elements (i.e., orders) become deterministic and known, respectively, combining [6, 11].

We attempt to obtain new insights and knowledge into synchromodal planning problems, including stochastic elements, by proposing a simulation based approach on a multi-period time window. To the best of our knowledge, this research is the first to address both vehicle routing (explicitly) and a container-to-mode assignment (implicitly), including uncertainty in the pick-up and delivery appointments, by generating potential future scenarios in order to obtain the best decision(s) that is resistant to change. The uncertainty element in our model is based on probability distributions, which has the benefits of incorporating distributional information and hence results in less moderate solutions than the classical robust optimisation approaches where probability distributions are ignored.

In Sect. “[Problem Description](#)” we present the problem and make some assumptions to create a computationally tractable mathematical model. Then in Sect. “[Simulation Approach](#)” we present our simulation approach for this problem. Section “[Results](#)” presents some benchmark problems and the results for a case study. We end with some conclusions and ideas for further research.

Problem Description

We use a specific network structure in this chapter. This network structure is relatively simple, however, it is close to the daily operations of a Dutch LSP. We assume three types of terminals in our problem, see Fig. 13.1: one inland terminal, T_{Origin} and one container terminal T_{Rot} in the Port of Rotterdam, both operated by the LSP and t deep-sea terminals $I = (T_1, T_2, \dots, T_t)$, operated by other agents. Every day, multiple barges depart from T_{Origin} to different deep-sea

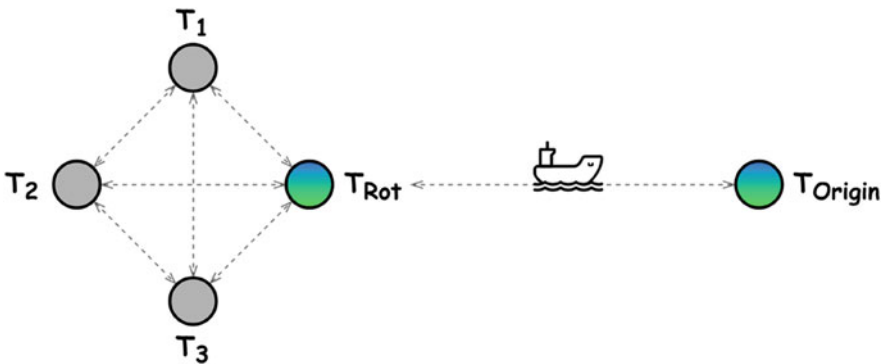


Fig. 13.1 The transportation network under consideration with $t = 3$

terminals within the port of Rotterdam. We consider the transportation of freight from (i) T_{Origin} to multiple deep-sea terminals, denoted by outland orders, and from (ii) deep-sea terminals back to T_{Origin} , denoted by inland orders. Notice that no freight needs to be shipped between any two deep-sea terminals within the port of Rotterdam. The terminals within the port region are denoted by ‘region D ’. Besides the use of a limited number of barges, we assume that there is an unlimited number of trucks that can be used for urgent freight that cannot be transported by barge. We propose to use a MPTW approach combined with simulation. We need to simplify our mathematical model by making several assumptions. The MPTW is divided into a finite number of time steps (i.e., discrete approach), where each time step corresponds to 3 h in real-life. The multi-period time window starts at time step 0 and covers 9 days, until time step 72. At each decision moment, we have a Controlled Time Window (CTW), which is the next time step, and the Single-Period Time Window (SPTW), which is the time window containing all the relevant information known, i.e., the planning horizon of 32 time steps. In Fig. 13.2 the MPTW approach is visualised. At the start of each decision moment a decision has to be made for the upcoming CTW, based on the information available in the concerned SPTW. This information could be both deterministic and stochastic. When inland orders become known, the barge planners request an appointment, 12 time steps in advance relative to the requested appointment time. This requested appointment is confirmed 4 time steps later, i.e., 8 time steps in advance. Outland orders become known 12 time steps in advance relative to the release time. The corresponding requested appointment is confirmed in the same way. The confirmed appointment time could be scheduled at the requested

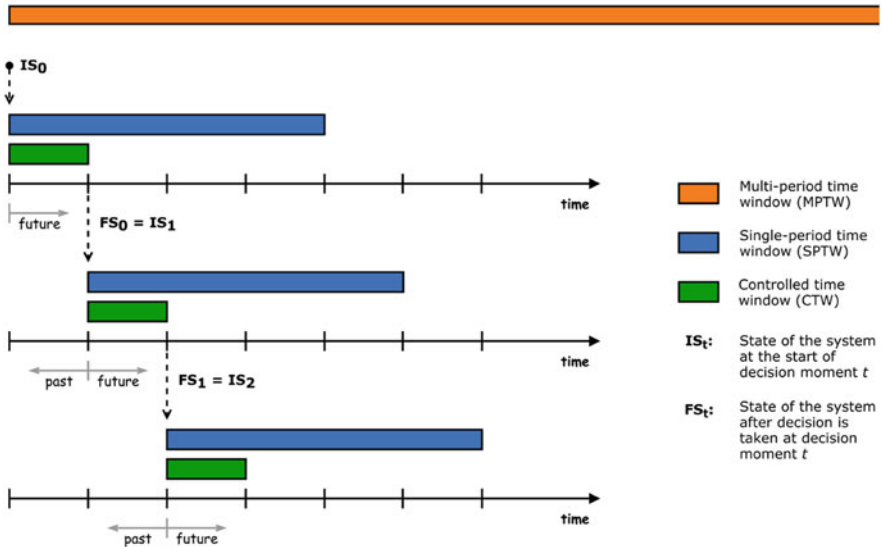


Fig. 13.2 Solving the planning problem using a multi-period time window approach

appointment time (0), earlier (at most one time step) or later (one until five time steps), with probability distribution $p = (p_{-1}, p_0, p_{+1}, p_{+2}, p_{+3}, p_{+4}, p_{+5})$.

This means that the SPTW can be divided into three parts relative to the requested appointment times:

1. Orders having a requested appointment time in the interval $[0, 8]$ that is confirmed already.
2. Orders having a requested appointment time in the interval $(8, 12]$ that is not confirmed yet.
3. Outland orders having a pick-up time in the interval $[0, 12]$ and a requested appointment time strictly greater than 12.

The three deep-sea terminals in the Rotterdam region may only be visited in case of a confirmed appointment. The number of barges that may visit an appointment is restricted to one. Observe that this restriction does only apply to barges. The travel times of the barges are known and fixed, 1 time step in the Port area and 7 time steps between the origin and the container terminal. The travel times of the trucks are also known and fixed, 1 and 2 time steps, respectively. At any point in time, an unlimited number of trucks is available at every terminal, which can (i) transport containers from the pick-up location directly to their destination or to the container terminal or (ii) transport containers from the container terminal to their destination. We charge costs for the use of trucks. Containers could be temporarily stored or switch vehicles at the container terminal T_{Rot} . However, handling time is taken into account for the unloading and loading process, i.e., one time step for each processing. Handling time is taken into account for the unloading process at the origin T_{Origin} , i.e., one time step.

Using the agreed time at the client's warehouse for (un)loading and the requested/confirmed pick-up and delivery times at the corresponding terminals, the orders K of the clients could be split into inland orders K^{in} and outland orders K^{out} . The properties of these orders are depicted in Table 13.1.

As mentioned before, the travel time of the long-haul trip between the single inland terminal and the port of Rotterdam is around 24 h. Since the other agents confirm the requested calls only 24 h in advance, a planning may become subject to changes when beneficial. For example, at some point in time, the LSP assigns outland order $k \in K^{out}$ to barge B , located at the origin, while the order is not confirmed yet. At that time, based on the requested pick-up and delivery times, the LSP benefits the most when the barge first picks up inland order $k' \in K^{in}$, then

Table 13.1 Properties of inland and outland orders

Inland orders:	Outland orders:
Pick-up location	Delivery location
Requested/confirmed pick-up time	Pick-up time
Delivery time	Requested/confirmed delivery time
Size of the order	Size of the order

delivers the outland order and finally picks up inland order $k'' \in K^{in}$. However, when time passes by, the requested times are confirmed and might deviate. Based on the real-time data it might be more beneficial to unload the outland order at the container terminal T_{Rot} and deliver the order at the delivery appointment by truck, such that the barge is able to visit some other confirmed appointments. Since the LSP has the ability to change the plan when beneficial, using multiple modes of transport, the problem described coincides with a synchromodal planning problem.

Simulation Approach

We propose an algorithm in which future scenarios are simulated for the requested appointment times, given their probability vector p . A future scenario (or realisation) is not a specific forecast of the future, but a plausible description of what might happen. These scenarios are generated by sampling from the probability distribution of the uncertainty elements. By analysing various possible future scenarios, the planning and decision making process will be more efficient. For example, given two potential decisions for some barge (i.e., routing and container assignment), say decision I and II, decision I might perform better for certain scenarios, while decision II achieves better results for some other scenarios. Within the decision making process, the goal is to find the best decision(s) for each CTW that is resistant to change, i.e., feasible and (sub)optimal for every potential future scenario that has been simulated, such that a proper solution is obtained for the entire MPTW. The approach is visualised in Fig. 13.3.

Start of the Algorithm

The decision making process consists of both routing of the barges and container assignment to the barges (and trucks) such that the total cost is minimised over the entire multi-period time window. Since the problem is twofold, the decision space grows rapidly. To be able to manage this immense space, the container-to-mode assignment is not included in the decision space explicitly, but is taken into account afterwards, implicitly. By using a flowchart of the algorithm, shown in Fig. 13.3, the simulation based model is presented and explained. Due to complexity reasons, the length of a CTW is set to one, i.e., $d = 1$.

Although a MPTW consisting of 72 time steps is solved, it is sufficient to consider only 55 decision moments. At the 55th decision moment (i.e., at time step $t = 54$) no uncertainty is involved anymore. Just by the construction of the in- and outland orders, every requested appointment time is scheduled before or at time step 62, implying that every order is confirmed at time step 54. Therefore, the remaining interval [54, 72] can be solved offline. In the flowchart, the decision moment is

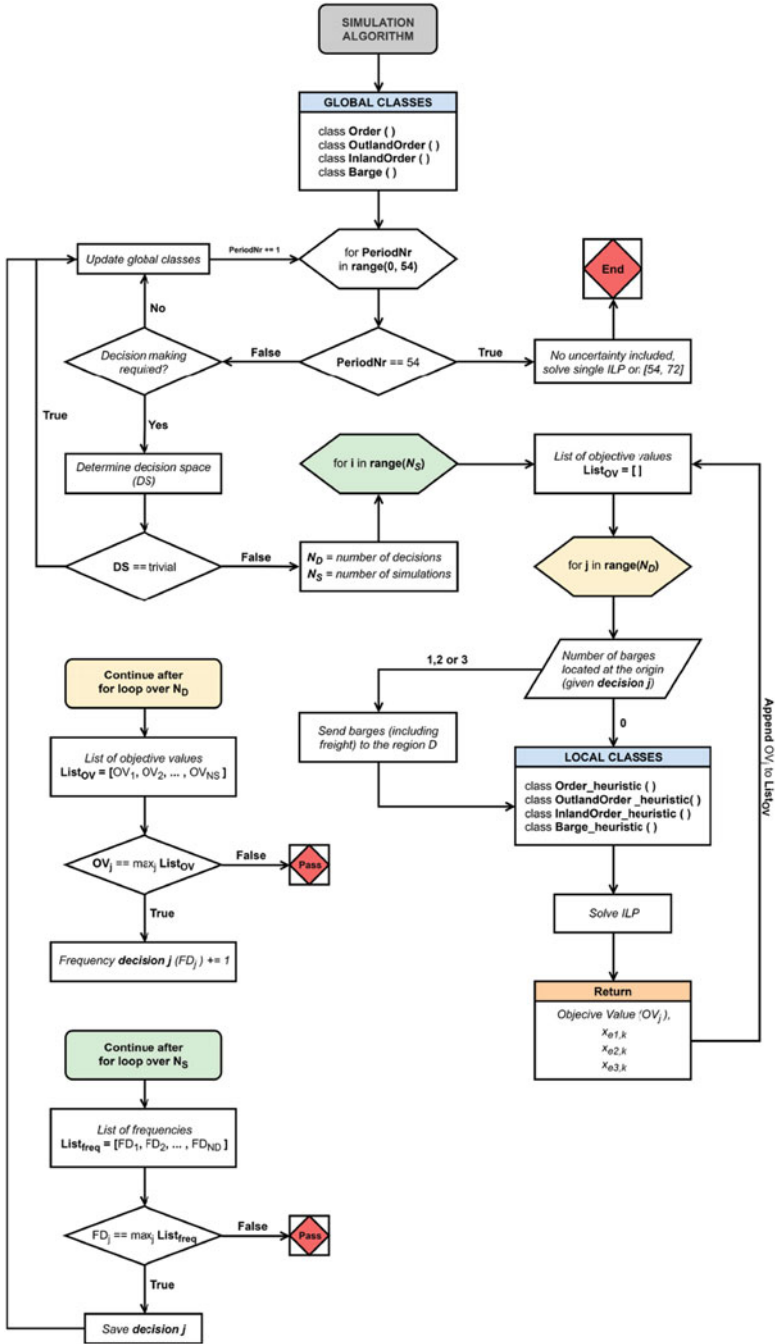


Fig. 13.3 Flow chart of the simulation algorithm

denoted as period number (PeriodNr), but it is equivalent terminology. For each period number less than 54, the algorithm checks if a decision has to be made at all, which is almost always the case. Only if all barges are on their way from the origin to the container terminal or vice versa, the decision moment can be skipped until a barge arrives at one of the locations.

Decision Space

As an example, we use a set of three barges. At each decision moment t_a , each barge could be located at the origin (t_a, T_{Origin}) , the container terminal (t_a, T_{Rot}) or one of the deep-sea terminals $(t_a, T_i)_{i \in I}$ in the region D . Additionally, a barge could be on the move from the origin to the container terminal or vice versa, in case the initial node of the barge is (t, T_{Rot}) or (t, T_{Origin}) , respectively, for some $t_a + 1 \leq t \leq t_a + 6$. In the latter case no decision has to be made for the barge under consideration. We distinguish between trivial and non-trivial decisions. Trivial decisions occur when there is only one direction for a barge, non-trivial decisions have several options to choose from.

If a barge is located at the origin, two possible decisions can be made. The barge could depart to the container terminal or stay another time step at the origin. If a barge is located at the container terminal, at most five possible decisions can be made. The barge could depart to the origin, it could stay at the container terminal or the barge could visit one of the three deep-sea terminals in case an appointment is scheduled at the upcoming time step. In case a barge is located at one of the deep-sea terminals, at most four possible decisions can be made. The barge could depart to the container terminal or it could visit one of the three deep-sea terminals in case an appointment is scheduled at the upcoming time step. From a theoretical point of view it could happen that 5^3 decisions could be made. However, in practice such a scenario would never happen. Moreover, due to symmetry, the number of decisions can be reduced drastically. During the performed experiments, on average 3.14 decisions could be made at each non-trivial decision moment, having a maximum of 20 decisions. In the model, a decision is denoted as a triple $(decision_{B1}, decision_{B2}, decision_{B3})$, where the i -th element corresponds to the (potential) upcoming location of barge i . Observe that the notation excludes the container-to-mode assignment.

Trivial Decisions

Quite often there is only one possible direction for a barge. Decisions could be trivial in different ways. In case no appointment is scheduled at the upcoming time step, a barge located at one of the deep-sea terminals can only return to the container terminal. Moreover, in case a pick-up appointment was scheduled at the current location, the order has to be (partially) assigned to the barge based on its remaining

capacity. In case a barge, located at the region D , does have some containers on board corresponding to an outland order having its delivery appointment at the upcoming time step, the barge is obliged to visit the appointment. Observe that decisions belonging to delivery appointments must be taken at an earlier stage. Furthermore, the decision is trivial when a barge is located at the origin and its previous location was the container terminal, i.e., the barge just arrived. Since a barge must stay at least one time step at the origin to unload the containers on board (and possibly load some new containers), the decision is fixed.

Decision (t_a, T_{Rot}) to $(t_a + 1, T_{Rot})$

Although at first sight this might seem a trivial decision, it is not. At the container terminal a barge is allowed to unload or load some containers, implying that the decision includes the assignment of the containers located at the container terminal and on board of the barge itself. Even orders may be split into suborders, which causes some extra difficulties. Even if there is only one possible decision, the simulation process has to be done to reveal what containers to load and unload. For each potential decision, in which at least one barge decides to stay at the container terminal, we keep track of how frequently an order is transported by that barge. If so, the order could be transported fully or partially. Therefore, the number of containers per order is recorded as well. In the end, orders that occur in more than half of the simulations are taken into account in the final decision. The actual quantity equals the average number of containers transported by barge, rounded to the nearest integer.

Decision $(t_a, Origin)$ to $(t_a + 7, T_{Rot})$

At the origin, the only prerequisite is that a barge has to wait at least one time step after arrival (to unload the containers on board). After this, however, no loading time is taken into account, implying that the assignment is not based on the freight on board of the barge. Hence we need to determine what orders are, fully or partially, transported to the container terminal by the barge under consideration. As will be described in Sect. “[Solving the ILP](#)”, the ILP used to find the (estimated) objective value for each possible decision is a heuristic excluding the container flow from the origin to the container terminal. The container-to-mode assignment has to be done manually. Just as for the previous decision, we keep track of the frequency an order is transported by that barge, including the number of containers per order. In the end, orders that occur in more than $\alpha\%$ of the simulations are taken into account in the final decision, where α is a predefined threshold value. This percentage has to be significantly higher than before, because we want to ensure that no conflicting appointments could occur, which might lead to infeasibility. During the experiments, α is set to 95, 90, and 85.

Remaining Decisions

The container assignment for the remaining decision space is straightforward. A concrete example is if a barge is located at the container terminal T_{Rot} and there is a single potential decision: departure to the origin. In that case, only the containers on board of that barge are forced to be transported to the origin. Just like for the delivery appointments mentioned above, the assignment of the containers to the barge has to be done at an earlier stage. In a similar way, the container-to-mode assignment for other decisions is based on the freight on board of the barge.

Solving the ILP

If the decision is non-trivial, a pre-specified number of simulations, denoted by N_S , is performed to seek the best decision(s) that is resistant to change. For every future scenario (or simulation) and every decision, an ILP, representing the SPTW problem, has to be solved. In case the number of simulations and decisions increases, the computational time significantly increases. Therefore a less time consuming heuristic is preferred. The output of each ILP does not have to be precise, because we are only interested in finding the best decision(s), given a possible future scenario. Therefore, an estimated objective value is sufficient. The time-space graph can be modified in various ways as can be found in Chaps. 8 and 9.

Results

In order to benchmark the Simulation Approach, we conducted experiments for a set of randomly generated instances. In this section the benchmarked results of Solution Approach are presented, interpreted and compared to the lower bounds computed for the benchmark solution methods. We will first shortly introduce how the instances are generated and which benchmark solution methods are used.

Design of Experiments

For the experiment we used $t = 3$, the network as depicted in Fig. 13.1. In the experiment we use a set of randomly generated instances, each containing a specific realisation of orders. For each instance a number of orders is generated, setting the pick-up location, the pick-up time, the delivery location, the delivery time and the size of the order. The pick-up location (for inland orders) and the delivery location (for the outland orders) are chosen from T_1 , T_2 and T_3 with equal probability. For inland orders the requested pick-up time comes from a Uniform distribution $\{1, 59\}$

and the confirmed pick-up time follows from the probability vector as defined earlier. The due time is uniformly chosen among discrete values between 13 and 20 time steps after the requested pick-up time. For outland orders the pick-up time comes from a Uniform distribution $\{0, 62\}$. The due time is discrete uniformly chosen between 9 and 15 time steps after the pick-up time. For all orders the size of the order is also discrete uniformly chosen between 1 and 5.

Benchmark Solution Methods

In this section we present the benchmark methods. First we introduce the method for achieving a lower bound (the B_x -models), then we present three simple online optimisation methods.

In the normal setting, orders become known 12 time steps in advance. In practice those orders include uncertainty. However, in order to determine a lower bound for the problem, we may disregard this uncertainty element, implying that orders are confirmed immediately after they become known: we call this the B_{12} model. Even better approximations can be found if the orders are announced at an earlier stage. The ultimate version of this is when all orders are known for the whole period, the B_{72} model. For these models the ILP is solved, based on the certain input data. For the B_{12} model this means that 51 ILPs for an SPTW have to be solved. For the B_{72} model only one, very big, ILP for one SPTW has to be solved. These are offline methods, as all input is considered known and certain. The difference between the score of these methods and the online methods can be seen as the price of uncertainty.

We consider three simple online optimisation methods. The *RC method* (Requested as Confirmed) is the most obvious method to solve the problem, in which the uncertainty is not taken into account. In other words, at each decision moment in the model the requested appointment times are assumed to be the confirmed ones. Given that assumption, each single-period time window is solved by solving only one ILP, whereafter the solution on the interval $[t_a, t_a + \delta]$ is actually saved. For inland orders, the RC method works fine. For the outland orders, however, the method has some disadvantages. If the confirmed appointment time turns out to be earlier than the requested one, and the model had decided (based on the requested appointment time) to send the order last minute to its destination, possibly unnecessary costs are incurred because the order has to be trucked. Especially orders of large size may cause problems. To ensure such problems will not occur, the *EC method* (Earliest as Confirmed) is proposed. The method does assume that the confirmation of each requested appointment time is the 'worst case'. In other words, the confirmed appointment time is assumed to be the earliest possible appointment time. Observe that the problem faced in the RC method does not relate to the inland orders, so it would probably not benefit to assume the earliest possible appointment time for both type of orders. Therefore, the requested appointment time belonging to an inland order is assumed to be the

average appointment time. Although the first possibility of infeasibility is avoided, the second possibility based on the container assignment at the origin could still occur. For both the RC and EC method, to deal with the uncertainty, the assumption is made to regard the appointment times of the requested ones in the beginning of the uncertainty interval to ensure that the transportation of outland orders by barge is possible (and no unnecessary costs are incurred). However, in most cases (to be precise five out of seven) the actual confirmed appointment time will be scheduled later than the requested one. Naturally, we do not charge any cost for being on time, but if an outland order does arrive at the container terminal way too early it is not beneficial. The containers corresponding to the order could stay on board of the barge the remaining time or the containers could be (partially) unloaded at the container terminal. The main drawback of the first option is the unnecessary use of the capacity, implying that some other orders cannot be loaded (fully) on the barge. Moreover, a pick-up appointment can only be visited if the delivery time does not collide with the delivery time of the outland order. The second option does avoid those drawbacks, but another disadvantage does appear. In the model, handling time is taken into account for both the unloading and loading processes at the container terminal, implying that the barge has to wait at least one time step after arrival at the container terminal. After the order has been unloaded, the order could be trucked to its destination, implying that cost has to be taken into account, or the order could be loaded on another (or the same) barge at a later moment, and transported to its destination without any cost, implying that this barge is forced to wait at least one time step at the container terminal as well. In other words, the model does not charge cost for being way too early, but the model does charge time, which could (again) lead to extra cost. Therefore, the *AC method* (Average as Confirmed) is added as a third model in order to investigate if shifting to the middle of the interval (i.e., the average) does benefit. Observe that the drawback of the RC method does emerge even more, and infeasibility could occur again in both ways.

Numerical Results

As can be seen in Table 13.2, the difference between the B_{12} model and the Simulation Approach is positive in terms of the average cost and the average number of trucks used, implying that the B_{12} model has overall better results. On the other hand, the Simulation Approach surpasses the performance of the simple methods. By surprise, the results of the AC method are much better than expected. It is even the solution method that used on average the least number of trucks for the long trips corresponding to outland orders, which is the part dealing the most with the uncertainty element. With the exception of one instance, the results for the Simulation Approach were at most within 20% of the B_{12} benchmark. As shown in Table 13.2, the gap was 12.5% on average, were some instances performed even better for the Simulation Approach than the B_{12} model (containing less uncertainty). The robustness is represented by the standard deviation of the average differences.

Table 13.2 Comparison of the results

	Mean	SD	Gap	Time MPTW	Time SPTW
B_{72}	-1181.8	780.4	-8.9%	22.52 h	22.52 h
B_{12}	0.0	0.0	0.0%	1.00 h	48.41 s
SIM	1647.7	1665.1	12.5%	11.74 h	0.21 h
AC	2134.1	1744.2	16.2%	0.81 h	48.31 s
RC	3102.3	2000.2	23.5%	0.79 h	46.94 s
EC	3815.9	1334.6	28.9%	0.76 h	45.26 s

Observe that the EC method not only is the most robust solution method but also has the worst performance in terms of average cost. Disregarding the EC method, the Simulation Approach surpasses the simple methods both in terms of average cost and robustness. Besides that, no penalties occurred during the Simulation Approach, whilst during the other methods penalties did occur. We may conclude that the Simulation Approach is more reliable.

The results also give an overview of the computational time of the solution methods. The models have been implemented in Python and were solved with the commercial solver CPLEX 12.7 through the Python API. The experiments were conducted using 16 cores of 2.4 GHz each, working with 16 GB of RAM. To put the running time of the Simulation Approaches into perspective, we should realise that the running time per multi-period time window (MPTW) refers to the transportation of containers distributed over the entire network. In actual applications, we are only interested in the running time per single-period time window (SPTW), where a decision need to be made for the upcoming hours. Concerning the SIM model, almost all SPTWs can be solved within 10 min, except for some outliers. In each instance, it might occur once or twice that the decision space is quite large, implying that the number of ILPs that need to be solved increases significantly. For example, during the experiments, it occurred that the decision space consisted of 20 potential decisions. It took 58 min to find the best decision, what is too time consuming. Although 15 decisions were discarded after the first phase (i.e., after the check that is performed after 11 simulations), the computational time of the first phase was 44 min. In this specific example, the 5 contenders had nonzero frequency already after three simulations. If the check was done right then, the computational time of the decision moment had shrunk to 34 min, where the first phase took only 12 min. Therefore, it might be a good idea to improve the first phase if the decision space is large. This could be done by performing the check at an earlier stage, by discarding decisions after one or two simulations if the objective value deviates too much from the others or by taking into account the symmetry of some decisions even more.

In addition, because of the tree structure of the algorithm, the Simulation Approach can be parallelised easily. Around 85% of the computational time is spend on solving ILPs, implying that the computational time on a PRAM computer with infinite many processors and zero communication cost is less than a minute [1].

Although this is practically unrealistic, it shows that the average running time per SPTW can be reduced.

As it can be seen the simple methods are advantageous in terms of running time. Solving a SPTW takes only 48 s, because only one normal ILP has to be solved. Finally, the heuristic ILP turns out to be almost four times faster in terms of running time. Since the extra amount of ILPs that need to be solved is increased only by a factor 1.36, the algorithm does benefit considerably. Without using the heuristic, running the SIM model would take around 28.85 h, respectively, implying that the algorithm has been speed up by a factor of approximately 2.25.

Conclusions and Further Research

In this chapter, an online optimisation approach is proposed, where the input data come in sequentially and decisions have to be taken while part of the relevant information is still uncertain or unknown. At each decision moment, the uncertainty element in the requested appointment times is converted to an offline optimisation problem by simulating various potential future scenarios. The approach is benchmarked against three simple online methods, in which the uncertainty is partially disregarded. The models assume that the requested appointment times will be confirmed at the requested, the earliest possible and the average appointment time, respectively. For this benchmark, experiments were carried out for 11 randomly generated instances. To say something about the quality and the practical relevance, the results were presented as the difference in results of the B_{12} model and the solution methods. Although the B_{12} model does not guarantee optimality, the outcome can still be used as a benchmark for the problem. With the exception of one instance, the results of the cost function for the simulation model were within 20% of the B_{12} model. The gap was 12.5% on average, where some instances performed even better for the simulation model than the B_{12} model (containing less uncertainty). Although not optimal, the simulation model provides a reliable vehicle routing and a container-to-mode assignment. Within the decision making process, the model finds the best decisions that are resistant to change. The practical relevance of the simulation model is restricted in the sense that the model is built on several assumptions. Every order is announced exactly 36 h in advance and confirmed exactly 12 h later. In practice, the announcement and confirmation times are more scattered. Furthermore, the assumption is made that the confirmed appointment can only be scheduled within an uncertainty interval of length seven, where the confirmation is based on the probability vector p that is uniformly distributed. Finally, due to the lack of real-world data, no adequate comparison can be made between the decisions made by the simulation algorithm and the decisions that barge planners would make in practice.

To conclude this research, we discuss some further research directions that may be developed and possibly lead to future success or usefulness. First of all, the network under consideration can be extended to a more practice oriented model.

That extended network should take into account the farther away deep-sea terminals located in the Maasvlakte I and II and waiting nodes on water located in between the two terminal clusters. Besides that, the model can be generalised in terms of the number of barges and orders or the size of the orders. Even more general, the simulation based model can be applied to any multi-commodity network design problem including uncertainty (based on probability distributions) using the idea of solving explicitly a vehicle routing problem and implicitly a container-to-mode assignment. Furthermore, a better comparison can be made if the probability vector p is inferred from real-world data. The most ideal would be when the probability distribution can be extracted from the real-world data. If not, a sensitivity analysis could be carried out for different kinds of probability distributions. Finally, because of the tree structure of the algorithm, the simulation model can be parallelised easily. By doing so, the algorithm should benefit in terms of computational time.

References

1. Chatterjee, S., & Prins, J. (2002). Parallel and distributed computing pram algorithms. *COMP*, 203, 13.
2. Chiscop, I. (2018). A robust optimization approach to synchronodal container transportation. Master's thesis, Delft University of Technology.
3. Crainic, T. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2), 272–288.
4. Crainic, T. G., & Laporte, G. (1997). Planning models for freight transportation. *Design and Operation of Civil and Environmental Engineering Systems*, 97(3), 343.
5. Foulds, L. (1981). A multi-commodity flow network design problem. *Transportation Research Part B: Methodological*, 15(4), 273–283.
6. Fragkos, I., Cordeau, J. F., & Jans, R. (2017). The multi-period multi-commodity network design problem. CIRRELT.
7. Gendron, B., & Crainic, T. G. (1994). Relaxations for multicommodity capacitated network design problems. Tech. rep., Centre de recherche sur les transports, Université de Montréal.
8. Han, J., Lee, C., & Park, S. (2014). A robust scenario approach for the vehicle routing problem with uncertain travel times. *Transportation Science*, 48(3), 373–390.
9. Hendriks, M. P. (2009). Multi-step optimization of logistics networks: Strategic, tactical, and operational decisions. Ph.D. thesis, Eindhoven University of Technology.
10. Johnson, D. S., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1978). The complexity of the network design problem. *Networks*, 8(4), 279–285.
11. Kooiman, K., Phillipson, F., & Sangers, A. (2016). Planning inland container shipping: A stochastic assignment problem. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications* (pp. 179–192). Springer.
12. Kooiman, K., Phillipson, F., & Sangers, A. (2020). A classification framework time stamp stochastic assignment problems. In *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems (ICORES), Valletta (Malta)*.
13. Pérez Rivera, A., & Mes, M. (2016). Service and transfer selection for freights in a synchronodal network. *Lecture Notes in Computer Science*, 9855, 227–242.
14. Phillipson, F. (2015). Planning nurses in maternity care: A stochastic assignment problem. In *Journal of Physics: Conference Series* (Vol. 616). IOP Publishing.

15. Rivera, A. E. P., & Mes, M. R. (2017). Anticipatory freight selection in intermodal long-haul round-trips. *Transportation Research Part E: Logistics and Transportation Review*, 105, 176–194.
16. Schmidt, G., & Wilhelm, W. E. (2000). Strategic, tactical and operational decisions in multinational logistics networks: a review and discussion of modelling issues. *International Journal of Production Research*, 38(7), 1501–1523.

Chapter 14

Optimising and Recognising 2-Stage Delivery Chains with Time Windows



F. Phillipson

Abstract In logistic delivery chains time windows are common. An arrival has to be in a certain time interval, at the expense of waiting time or penalties if the time limits are exceeded. This chapter looks at the optimal placement of those time intervals in a specific case of a barge visiting two ports in sequence. For the second port a possible delay or penalty should be incorporated. Next, recognising these penalty structures in data is analysed to if see certain patterns in public travel data indicate that a certain dependency exists.

Introduction

Delivery windows are a known phenomenon in time window constrained models for production scheduling and vehicle routing. In [5] an overview can be found of recent literature on the use in production logistics. In the context of a delivery performance model, a delivery window is defined as the difference between the earliest acceptable delivery date and the latest acceptable delivery date. In supply chain management and home delivery in e-commerce the problem of interest is the optimal positioning of the delivery time window to minimise the expected cost of untimely delivery, such as inventory costs and penalties or the estimation of accumulated delivery times with uncertainty [1, 5–10, 12, 13].

Delivery windows are also used in Vehicle Routing Problems (VRP). A VRP involves finding a set of routes, starting and ending at a depot, that together cover a set of customers. Each customer has a given demand, and no vehicle can service more customers than its capacity permits. The objective is to minimise the total distance travelled or the number of vehicles used, or a combination of these. A special case of the VRP is when the service at a customer's place must start within a given time window. There are two types of time windows. Time windows are called soft when they can be violated for a penalty cost. They are hard when they cannot

F. Phillipson (✉)
TNO, The Hague, The Netherlands

be violated, i.e., if a vehicle arrives too early at a customer, it must wait until the time window opens; and it is not allowed to arrive late. In all the cases these time windows are given in advance [2, 3, 11].

In this work a delivery chain is studied where a barge has to visit two ports. In each port a number of containers is handled. For the planning of the port, the planner of the barge should indicate a time slot in which the barge will arrive. If the barge is too early, it has to wait until the beginning of the slot. If the barge is too late, it has to wait some penalty time. If the barge arrives within the time slot, the handling starts immediately. This means that we introduce a penalty which occurrence is dependent on the arrival time, which duration is dependent on the arrival time in case of early arrival, in combination with a two-stage time window. Within this study, first the optimisation of the choice of the time slots is elaborated in Sect. “[Optimisation](#)”. The main question here is what the optimal time slots are to be communicated to minimise the total of the penalties. Secondly, in Sect. “[Recognising Time Windows in Data](#)” the way to recognise the existence of such time slots with penalties in travel data is studied. In practice often not all data and/or the precise process is known. There the question is if we only see the arrival and departure times of a barge (for example, from GPS or AIS data) can we predict the underlying process, to be able to predict the arrival time of the barges at some (final) stop.

Optimisation

The central case in this chapter is a delivery chain where a barge has to visit two ports. In each port a number of containers should be handled. For the planning of the port, the planner of the barge should indicate a time slot in which the barge will arrive. If the barge is too early, it has to wait until the beginning of the slot. If the barge is too late, it has to wait some penalty time. If the barge arrives within the time slot, the handling starts immediately. In this section the optimal choice of the time window is determined.

Problem Description

To formulate the problem, first some notation is defined:

I = Set of ports;

T_i = Transportation time to port $i \in I$, starting at the former location;

H_i = Handling time at port $i \in I$;

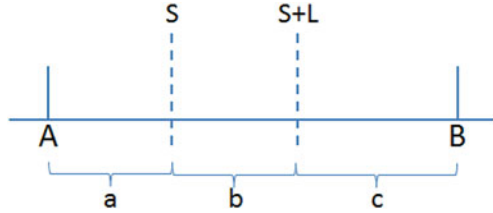
W_i = Waiting time at port $i \in I$;

S_i = Start time slot at port $i \in I$;

L = Length time slot;

K, k = Penalty wait time, fixed, stochastic or function depending on context.

Fig. 14.1 Process of the described problem



The question that arises is what would be the optimal start times S_1 and S_2 of both slots to minimise the sum of the waiting times ($W_1 + W_2$)? Different probability distribution functions are used for the transportation and handling times and, as a consequence, for the arrival time (X) at the port under consideration. The arrival, in each of the ports therefore we skip the indices here, will be in the interval (A, B) (see Fig. 14.1). We assume that, for each of the two stages, $S \geq A$ and $B \geq S + L$, while losing a part of the time slot will not be smart. Only if $L \geq (B - A)$ this will not hold, but then we have no problem. The arrival will be in one of the three intervals $a = [A, S]$, $b = [S, S + L]$ or $c = [S + L, B]$. For each realisation of the arrival time x we can calculate the waiting time:

$$\begin{array}{ll}
 A \leq x < S & W = S - x \\
 S \leq x \leq S + L & W = 0 \\
 S + L < x \leq B & W = K
 \end{array}$$

First Stage

Now the optimal choice for the starting time of the time slots can be derived, by minimising the expected waiting time as a function of S_1 . We assume three different options for the penalty: a fixed time, a function of the delay and a random value. At the first port the arrival time X is equal to the transportation time T_1 . For various probability distribution functions for T_1 we obtain the optimal value (S) for S_1 , the start of the first time slot.

Fixed Penalty

Given a fixed penalty K , the expected waiting time is given by:

$$\begin{aligned}
 \mathbb{E}[W] &= \mathbb{E}[W \mathbb{1}_{X < S}] + \mathbb{E}[W \mathbb{1}_{S \leq X < S+L}] + \mathbb{E}[W \mathbb{1}_{S+L \leq X}] \\
 &= \mathbb{E}[(S - X) \mathbb{1}_{X < S}] + \mathbb{E}[0 \mathbb{1}_{S \leq X < S+L}] + \mathbb{E}[K \mathbb{1}_{S+L \leq X}]
 \end{aligned}$$

$$\begin{aligned}
 &= S\mathbb{E}[\mathbb{1}_{X < S}] - \mathbb{E}[X\mathbb{1}_{X < S}] + 0 + K\mathbb{E}[\mathbb{1}_{S+L \leq X}] \\
 &= SF(S) - \int_{-\infty}^{\infty} x\mathbb{1}_{x < S}f(x)dx + K(1 - F(S + L)) \\
 &= SF(S) - \int_{-\infty}^S xf(x)dx + K(1 - F(S + L)).
 \end{aligned}$$

The expected waiting time is minimised by:

$$\frac{d}{dS}\mathbb{E}[W] = 0$$

resulting in

$$\begin{aligned}
 \frac{d}{dS}\mathbb{E}[W] &= \frac{d}{dS}SF(S) - \frac{d}{dS}\int_{-\infty}^S xf(x)dx + \frac{d}{dS}K(1 - F(S + L)) \\
 &= Sf(S) + F(S) - Sf(S) - Kf(S + L) = F(S) - Kf(S + L).
 \end{aligned}$$

So:

$$\frac{d}{dS}\mathbb{E}[W] = 0 \iff F(S) = K \cdot f(S + L).$$

Now any distribution for X can be used. For three examples this will be elaborated.

Uniform Distribution If the transportation time and consequently the arrival time X is uniform (A,B): $F(S) = \frac{S-A}{B-A}$ and $f(S) = \frac{1}{B-A}$, so we obtain:

$$\frac{d}{dS}\mathbb{E}[W] = 0 \iff \frac{S - A}{B - A} = K \cdot \frac{1}{B - A} \iff S - A = K \iff S = A + K.$$

Recall that S has a maximum value of $B - L$, thus $S = \min(A + K, B - L)$.

Exponential Distribution If the transportation time and consequently the arrival time X is exponential distributed (λ) the expected waiting time equals: Exponential: $F(S) = 1 - \exp^{-\lambda S}$ and $f(S) = \lambda \exp^{-\lambda S}$. So we obtain:

$$\begin{aligned}
 \frac{d}{dS}\mathbb{E}[W] = 0 &\iff 1 - \exp^{-\lambda S} = K\lambda \exp^{-\lambda(S+L)} \\
 &\iff 1 = (K\lambda \exp^{-\lambda L} + 1) \exp^{-\lambda S} \\
 &\iff -\lambda S = \log\left(\frac{1}{K\lambda \exp^{-\lambda L} + 1}\right) \\
 &\iff S = \frac{1}{\lambda} \log(K\lambda \exp^{-\lambda L} + 1).
 \end{aligned}$$

Normal Distribution If the arrival time is normal (μ, σ) distributed, where $\phi(\cdot)$ denotes the normal probability density function and $\Phi(\cdot)$ the cumulative probability density, the waiting time is minimised by solving for S in:

$$\Phi(S) = K\phi(S + L),$$

which has to be solved numerically.

Penalty as Function of Delay

Now assume the penalty depends on how late the barge is. Again, $\mathbb{E}[W]$ is calculated, since the only term that changes compared to the situation above is $\mathbb{E}[W\mathbb{1}_{S+L \leq X}]$. The penalty equals $k(X - S - L)$ for some function $k : [0, \infty) \rightarrow [0, \infty)$.

$$\mathbb{E}[W\mathbb{1}_{S+L \leq X}] = \mathbb{E}[k(X - S - L)\mathbb{1}_{S+L \leq X}] = \int_{S+L}^{\infty} k(x - S - L)f(x)dx.$$

The derivative follows from:¹

$$\begin{aligned} \frac{d}{dS}\mathbb{E}[W\mathbb{1}_{S+L \leq X}] &= \frac{d}{dS} \int_{S+L}^{\infty} k(x - S - L)f(x)dx \\ &= \int_{S+L}^{\infty} -k'(x - S - L)f(x)dx - k(x - S - L)f(x)|_{x=S+L} \\ &= - \int_{S+L}^{\infty} k'(x - S - L)f(x)dx - k(0)f(S + L) \\ &= - \int_0^{\infty} k'(x)f(x + S + L)dx - k(0)f(S + L). \end{aligned}$$

Combining with the steps above results in:

$$\frac{d}{dS}\mathbb{E}(W) = F(S) - \int_0^{\infty} k'(x)f(x + S + L)dx - k(0)f(S + L).$$

¹ Under some regularity assumptions, for instance, k must be differentiable on $(0, \infty)$ and continuous on $[0, \infty)$.

So

$$\frac{d}{dS}\mathbb{E}(W) = 0 \iff F(S) = \int_0^\infty k'(x)f(x+S+L)dx + k(0)f(S+L).$$

Note that if k is a constant, this expression reduces to what was found earlier.

Penalty is Random Variable, Independent of X

The third option concerns a random penalty K , independent of X . Then the last term becomes:

$$\mathbb{E}[W\mathbb{1}_{S+L\leq X}] = \mathbb{E}[K\mathbb{1}_{S+L\leq X}].$$

Since K and X are independent, so are K and $\mathbb{1}_{S+L\leq X}$, the expectations can be multiplied to obtain:

$$\mathbb{E}[W\mathbb{1}_{S+L\leq X}] = \mathbb{E}[K]\mathbb{E}[\mathbb{1}_{S+L\leq X}] = \mathbb{E}[K](1 - F(S+L))$$

$$\frac{d}{dS}\mathbb{E}(W) = 0 \iff F(S) = \mathbb{E}[K]f(S+L).$$

In the case that K is constant this expression reduces to the first case again.

Second Stage

The first stage resulted in a general formulation that can be used for the second stage, given that the probability distribution of the arrival time at the second port is known. However, the probability distribution function of the arrival time (X) is more complicated, namely the sum of two transportation times, a handling time and possibly a penalty. First the penalty is neglected, later, the propagation of the penalty is studied.

Second Time Slot Without Penalty in the First Time Slot

For the second time slot without penalty, the same approach can be taken as in the first stage. First note that here it is assumed that there is no penalty in the first time slot, but obviously there is one in the second (since otherwise nothing would have to be optimised). Now again for the three probability distributions (of each of the

stochastic variables, adding up to the arrival time at the second stage) the solution can be derived.

Uniform Distribution If the two transportation times and the handling time all follow a uniform distribution, the arrival time has an Irwin-Hall distribution [4]. This distribution converges quickly to the normal distribution. From our experience, even in the case of only three underlying uniform distributions, a normal approximation is usable in practice. If $T_1 \sim \text{uniform}(U_1, U_2)$, $T_2 \sim \text{uniform}(U_3, U_4)$ and $H_1 \sim \text{uniform}(U_5, U_6)$, then by approximation $X \sim \text{Normal}(\mu, \sigma)$ where

$$\mu = \frac{1}{2}(U_2 - U_1) + \frac{1}{2}(U_4 - U_3) + \frac{1}{2}(U_6 - U_5),$$

$$\sigma = \sqrt{\frac{(U_2 - U_1)^2}{12} + \frac{(U_4 - U_3)^2}{12} + \frac{(U_6 - U_5)^2}{12}}.$$

Now the method for the normal distribution of the previous stage can be used.

Exponential Distribution In the case of exponential handling and transporting times (and assuming independence) the second arrival time has an Erlang(3, λ) distribution. This means:

$$F(x) = 1 - \sum_{n=0}^2 \frac{1}{n!} \exp^{-\lambda x} (\lambda x)^n$$

$$f(x) = \frac{1}{2} \lambda^3 x^2 \exp^{-\lambda x}.$$

The formula above reduces the problem to finding S such that:

$$1 - \exp^{-\lambda S} - \lambda S \exp^{-\lambda S} - \frac{1}{2} \lambda^2 S^2 \exp^{-\lambda S} = \frac{1}{2} K \lambda^3 S^2 \exp^{-\lambda(S+L)}$$

$$\iff \exp^{\lambda S} - 1 - \lambda S - \frac{1}{2} \lambda^2 S^2 = \frac{1}{2} K \lambda^3 S^2 \exp^{-\lambda L}$$

$$\iff \exp^{\lambda S} = \frac{1}{2} (K \lambda^3 \exp^{-\lambda L} + \lambda^2) S^2 + \lambda S + 1.$$

The latter expression can be solved for S numerically.

Normal Distribution If the two transportation times and the handling time are all normally distributed and independent, the arrival time has again a normal distribution. If $T_1 \sim \text{Normal}(\mu_1, \sigma_1)$, $T_2 \sim \text{Normal}(\mu_2, \sigma_2)$ and $H_1 \sim \text{Normal}(\mu_3, \sigma_3)$

then $X \sim Normal(\mu, \sigma)$ where

$$\mu = \mu_1 + \mu_2 + \mu_3,$$

$$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}.$$

Now the method for the normal distribution of the previous section can be used.

Propagation of Penalty: Second Time Slot with Penalty

The challenge now is to derive an expression for the arrival time at the second port, including the fact that there may have been a penalty at the first port. Then the formula presented earlier can be applied to find the expression that has to be solved.

We assume T_1 , T_2 and H_1 to be independent. The time that is added to this due to not arriving within the time frame, is the penalty P . So P is not only due to arriving late. We see then:

$$P = \begin{cases} S - T_1 & \text{if } T_1 \leq S \\ 0 & \text{if } S < T_1 \leq S + L \\ k & \text{if } T_1 > S + L. \end{cases}$$

Now we are interested in the second arrival time $X = T_1 + P + H + T_2$. Since P and T_1 are dependent of each other and the rest is independent, we will call $X_1 = T_1 + P$ and $X_2 = H + T_2$. The interesting part here is X_1 :

$$X_1 = \begin{cases} T_1 + S - T_1 = S & \text{if } T_1 \leq S \\ T_1 + 0 = T_1 & \text{if } S < T_1 \leq S + L \\ T_1 + k & \text{if } T_1 > S + L. \end{cases}$$

Now the cumulative distribution function of X_1 equals:

$$F_{X_1}(x) = \begin{cases} 0 & \text{if } x < S \\ F_{T_1}(x) & \text{if } S \leq x \leq S + L \\ F_{T_1}(S + L) & \text{if } S + L < x \leq S + L + k \\ F_{T_1}(x - k) & \text{if } x > S + L + k. \end{cases}$$

This is visualised in Fig. 14.2. The jump in the point S means that the random variable is not absolutely continuous. This is what we expect, since the probability of starting the handling at point S equals $\mathbb{P}(T_1 \leq S) = F_{T_1}(S)$, which is strictly

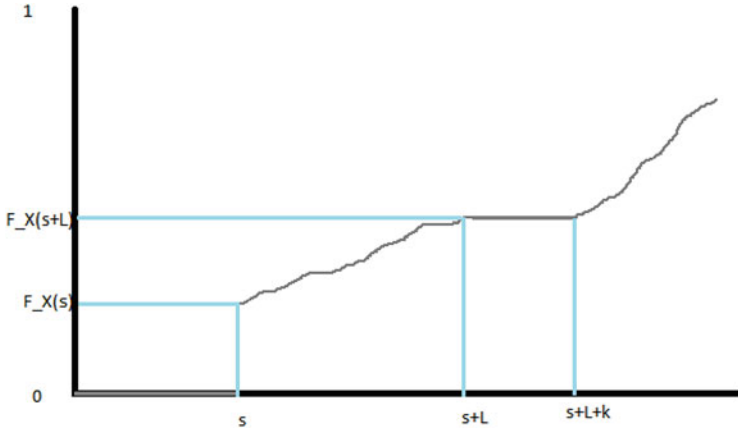


Fig. 14.2 CDF of time that handling begins

positive. We can describe the ‘density’ in this way:

$$f_{X_1}(x) = \begin{cases} 0 & \text{if } x < S \\ \text{mass } F_{T_1}(S) & \text{if } x = S \\ f_{T_1}(x) & \text{if } S \leq x \leq S + L \\ 0 & \text{if } S + L < x \leq S + L + k \\ f_{T_1}(x - k) & \text{if } x > S + L + k. \end{cases}$$

Now we would like to obtain the cumulative density function and the density of the sum of X_1 and X_2 .²

$$\begin{aligned} F_X(x) &= \mathbb{P}(X_1 + X_2 \leq x) = \int_{b=-\infty}^{\infty} \int_{a=-\infty}^{x-b} f_{X_1, X_2}(b, a) da db \\ &= \int_{b=-\infty}^{\infty} \int_{a=-\infty}^{x-b} f_{X_1}(b) f_{X_2}(a) da db = \int_{b=-\infty}^{\infty} \int_{a=-\infty}^{x-b} f_{X_2}(a) da f_{X_1}(b) db \\ &= \int_{-\infty}^{\infty} F_{X_2}(x - b) f_{X_1}(b) db. \end{aligned}$$

² Note that the following computations are strictly speaking ill-defined, since f is not a continuous function. However, it is correct and this way a more intuitive derivation is given. To be precise, one would have to use the Lebesgue–Stieltjes integral to avoid speaking of f . Also note that we use independence of X_1 and X_2 when their joint probability distribution function is written as the product of the marginals.

Now, using the description that we found of f_{X_1} , we obtain:

$$\begin{aligned} F_X(x) &= \int_{-\infty}^{\infty} F_{X_2} A(x-b) f_{X_1}(b) db \\ &= F_{T_1}(S) F_{X_2}(x-S) + \int_S^{S+L} F_{X_2}(x-b) f_{T_1}(b) db \\ &\quad + \int_{S+L+k}^{\infty} F_{X_2}(x-b) f_{T_1}(b-k) db. \end{aligned}$$

Differentiating this with respect to x yields (under some regularity conditions):

$$\begin{aligned} f_X(x) &= F_{T_1}(S) f_{X_2}(x-S) + \int_S^{S+L} f_{X_2}(x-b) f_{T_1}(b) db \\ &\quad + \int_{S+L+k}^{\infty} f_{X_2}(x-b) f_{T_1}(b-k) db. \end{aligned}$$

Note that $X_2 \geq 0$, so $f_{X_2}(x-b)$ will be 0 for $b > x$. So in practice, a part of the integral will drop out.

To find the optimal time, we need to use the formula of the first stage optimisation again: $F_X(S_2) = k_2 f_X(S_2 + L_2)$. We obtain as the equation that has to be solved for S_2 :

$$\begin{aligned} &F_{T_1}(S) F_{X_2}(S_2 - S) + \int_S^{S+L} F_{X_2}(S_2 - b) f_{T_1}(b) db \\ &+ \int_{S+L+k}^{\infty} F_{X_2}(S_2 - b) f_{T_1}(b-k) db = k_2 F_{T_1}(S) f_{X_2}(S_2 + L_2 - S) \\ &+ k_2 \int_S^{S+L} f_{X_2}(S_2 + L_2 - b) f_{T_1}(b) db + k_2 \int_{S+L+k}^{\infty} f_{X_2}(S_2 + L_2 - b) f_{T_1}(b-k) db. \end{aligned}$$

By rearranging a bit, we get:

$$\begin{aligned} &(k_2 f_{X_2}(S_2 + L_2 - S) - F_{X_2}(S_2 - S)) F_{T_1}(S) \\ &= \int_S^{S+L} (F_{X_2}(S_2 - b) - k_2 f_{X_2}(S_2 + L_2 - b)) f_{T_1}(b) db \\ &\quad + \int_{S+L+k}^{\infty} (F_{X_2}(S_2 - b) - k_2 f_{X_2}(S_2 + L_2 - b)) f_{T_1}(b-k) db. \end{aligned}$$

Note that in any situation with a sum of random variables, the convolution integral appears. This usually cannot be simplified, except for nice situations such as some known sums of random variables. This is the reason for the integrals with two densities in them. The penalty P is of different nature in different cases, this

accounts for the multiple integrals. This suggests that there is not much hope of finding nicer expressions.

Case

As example we look at the following case. As input data we use:

$$T_1 = U(180; 234)$$

$$H_1 = U(50; 150)$$

$$T_2 = U(120; 156)$$

$$H_2 = U(50; 150)$$

$$L = 30 \text{ minutes}$$

$$K = 45 \text{ minutes}$$

Now the optimal value of S_1 can be calculated by $S_1^* = \min(A + P, B - L) = \min(180 + 45; 234 - 30) = 204$ resulting in $E(W_1) = 5.33$. The same holds for S_2 . First for the case neglecting the penalty at the first port. Minimum value for S_2 can be derived easily $S_2 = 180 + 50 + 120 = 350$, and also the maximum value $S_2 = 234 + 150 + 156 = 540$. The arrival time on port 2 is a sum of three uniform distributed variables. If we assume that the sum of three uniform variables has a normal distribution, then the arrival time on port 2 is normal distributed with $\mu = 350 + 0.5 * (190) = 445$ and $\sigma = \sqrt{\frac{(54)^2}{12} + \frac{(100)^2}{12} + \frac{(36)^2}{12}} = 34.4$. Solving the formula of the first stage for a normal distribution gives $S_2^* = 438$.

These results can be checked by a numerical simulation of 70,000 realisations of trips with these parameters. Figure 14.3 shows that the minimum delay is reached (indeed) around 204. Furthermore, it can be seen how sensitive the outcome is for a choice of S_1 : 10 min off, gives 5 min extra delay.

Simulating the second stage without penalty results in the outcome as shown in Fig. 14.4. The optimal value of 438 is confirmed; however, the graph is rather flat around the optimum, and the sensitivity of the delay on the window is low.

From the simulation of the second stage with penalty at the first stage also comes that taking the penalty into account, the optimal S_2^* becomes 441, as depicted in Fig. 14.5.

Recognising Time Windows in Data

In practice often not all data and/or the precise process is known. For example, only GPS-data is available and this information is used in planning. Then it would be nice to understand where the interactions and (possible) correlations in data comes from. In this section we look at the data in the case were we only see the arrival and departure times of a barge (for example, from GPS or AIS data) and want

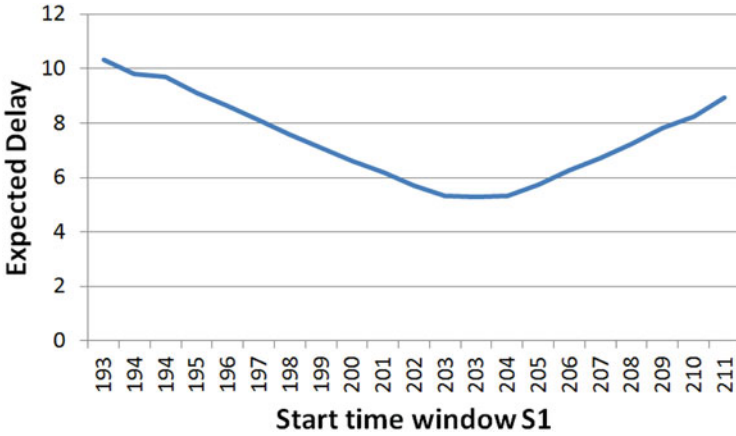


Fig. 14.3 Simulated optimum S_1

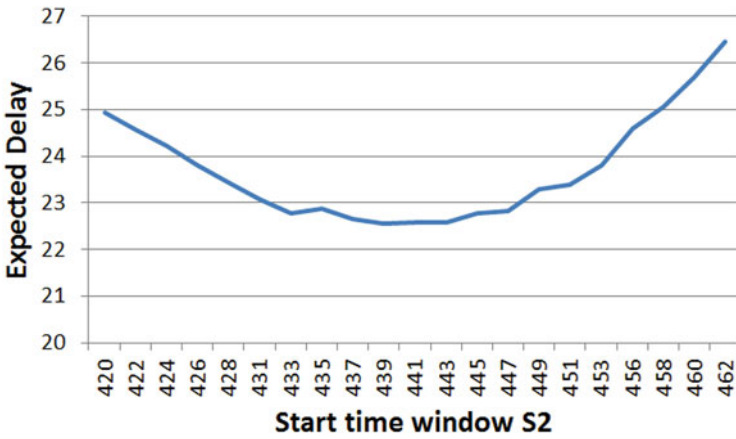


Fig. 14.4 Simulated optimum S_2 without penalty

to understand the underlying process better by analysing this data. We want, for example, to be able to predict the arrival time of the barges at some (final) stop. For this we can try to predict the separate steps in the chain, here, for example, the transportation times and the handling times. But what if there are dependencies, for example, caused by waiting times that are depending on whether some time slot is met by arrival, as explained in the previous section.

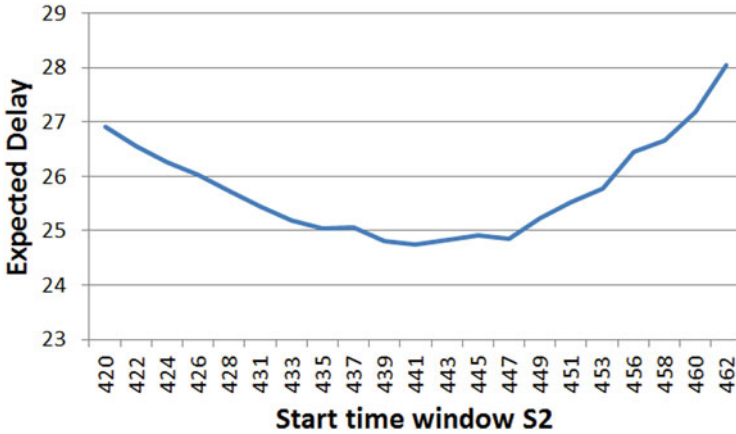


Fig. 14.5 Simulated optimum S_2 with penalty

Analysis

To get some idea on this, we simulated the process as defined in the previous section for four cases:

1. No time slot; a barge is handled on arrival at each port.
2. Optimal time slots are chosen; as defined in the previous section.
3. Time slots are chosen around the expected arrival time; the planner puts the time slot symmetrically around the expected arrival time.
4. No optimisation; the planner places the start of the time slot at the earliest arrival time.

As numerical input we take (in minutes):

$$T_1 = U(180; 234)$$

$$H_1 = U(50; 150)$$

$$T_2 = U(120; 156)$$

$$H_2 = U(50; 150)$$

$$L = 30$$

$$K = 30$$

This gives a minimal lead time of 400 min and a maximum lead time of 690 (plus 60 min of penalties) minutes. For each of the four cases we simulated 5000 realisations, were only the arrival and departure times were reported. From these times the two transportation and two handling times were calculated, as depicted in Fig. 14.6. Again for each of the four cases, the correlation between the four arrival/departure times and the average total lead time were derived. For each correlation value also the p-value was calculated to test whether the correlation is significantly different from zero. The results are presented in Tables 14.1, 14.2, 14.3, 14.4, and 14.5.

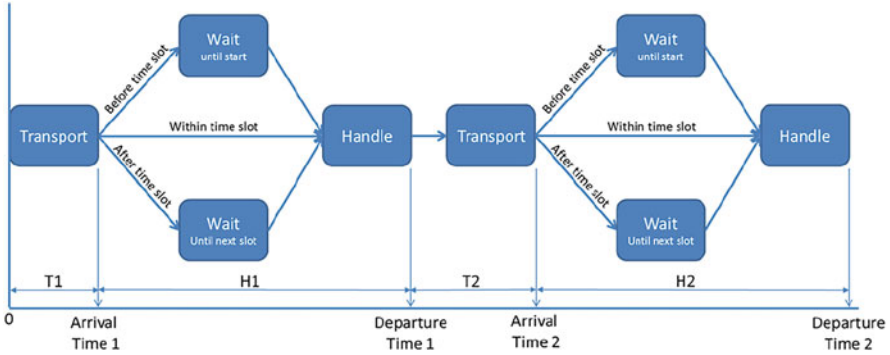


Fig. 14.6 Process

Table 14.1 Correlation in the case ‘No time slot’; p-value between brackets

	T_1	H_1	T_2
H_1	0.028 (0.052)		
T_2	-0.018 (0.201)	0.001 (0.961)	
H_2	0.022 (0.122)	-0.005 (0.750)	-0.001 (0.952)

Table 14.2 Correlation in the case ‘Optimal time slot’; p-value between brackets

	T_1	H_1	T_2
H_1	-0.210 (0.000)		
T_2	0.004 (0.787)	0.003 (0.850)	
H_2	-0.008 (0.559)	-0.056 (0.000)	-0.005 (0.741)

Table 14.3 Correlation in the case ‘Time slots around the expected arrival time’; p-value between brackets

	T_1	H_1	T_2
H_1	0.216 (0.000)		
T_2	0.008 (0.562)	-0.012 (0.406)	
H_2	0.047 (0.000)	0.063 (0.000)	-0.009 (0.518)

Table 14.4 Correlation in the case ‘Not optimised’; p-value between brackets

	T_1	H_1	T_2
H_1	0.382 (0.000)		
T_2	-0.007 (0.635)	-0.011 (0.431)	
H_2	0.171 (0.000)	0.232 (0.000)	0.031 (0.0278)

There are some observations we can make:

1. In the case ‘No time slots’ there is no correlation between the transportation and handling times.

Table 14.5 Total process time of the four cases

Case	Total time
1	545
2	567
3	572
4	581

2. In the case ‘No optimisation’ there exist: positive correlation between (T_1, H_1) , positive correlation between (T_1, H_2) and positive correlation between (H_1, H_2) , all by the penalty. There also is a (small but significant) correlation between (T_2, H_2) .
3. In the case ‘Time slots are chosen around the expected arrival time’, the correlations become lower; the effect of the penalty is less than in the not optimised case.
4. In the case ‘Optimal time slots chosen’, the correlation between (T_1, H_2) disappear (no delay propagation anymore), the two other relations that had a positive correlation (T_1, H_1) and (H_1, H_2) become negative. This means that longer delays do not cause the big penalty anymore, but being early (lower arrival time) leads to small waiting times.

Limitations

Up to here we assumed that the planning and realisations are independent. However, in practice people are going to react on realisations. For example:

- If a barge is early, the captain can decide to slow down and save fuel. This could lead to a shift in the transportation time distribution and from the optimised case to the ‘time slots around expected arrival time’ case.
- If a barge had delay in the first part (transportation, penalty and/or handling) the captain could decide to go faster. This again leads to a shift in the transportation time distribution and potentially a decrease in the correlation between (T_1, T_2) and (H_1, T_2) .

Conclusions

This chapter investigated a delivery chain in logistics, where a barge has to visit two ports and was faced by delivery time slots in which the barge has to arrive. We looked at two issues: first, how can the time slot be chosen optimally and secondly, how can time slots with penalty for untimely arrival be recognised in travel data. For the former an optimisation framework was given to derive the optimal time slots at the first and second stage with various options for penalty

functions in case of a not timely arrival. For certain distribution functions of the handling and transportation times an explicit expression was derived. Also for the most complicated case, the second stage with propagation of the penalty of the first stage, an expression was derived. For the latter some insight was given to recognise these time slot constructions from correlation values of travel and handling times. Four cases were distinguished where each case showed specific characteristics in the correlation values. The characteristics could, in practice, be compensated by the interaction of humans.

References

1. Agatz, N., Campbell, A., Fleischmann, M., & Savelsbergh, M. (2011). Time slot management in attended home delivery. *Transportation Science*, 45(3), 435–449.
2. Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., & Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers and Operations Research*, 40(3), 856–866.
3. de Armas, J., Melián-Batista, B., Moreno-Pérez, J. A., Brito, J. (2015). GVNS for a real-world rich vehicle routing problem with time windows. *Engineering Applications of Artificial Intelligence*, 42, 45–56.
4. Batsyn, M., & Kalyagin, V. (2013). An analytical expression for the distribution of the sum of random variables with a mixed uniform density and mass function. In *Models, Algorithms, and Technologies for Network Analysis* (pp. 51–63). Springer
5. Bushuev, M. A., & Guiffrida, A. L. (2012). Optimal position of supply chain delivery window: Concepts and general conditions. *International Journal of Production Economics*, 137(2), 226–234.
6. Garg, D., Narahari, Y., & Viswanadham, N. (2006). Achieving sharp deliveries in supply chains through variance pool allocation. *European Journal of Operational Research*, 171(1), 227–254.
7. Guiffrida, A. L., & Nagi, R. (2006). Cost characterizations of supply chain delivery performance. *International Journal of Production Economics*, 102(1), 22–36.
8. Hernandez, F., Gendreau, M., & Potvin, J. Y. (2017). Heuristics for tactical time slot management: a periodic vehicle routing problem view. *International Transactions in Operational Research*, 24(6), 1233–1252.
9. Safaei, M., Issa, S., Seifert, M., Thoben, K. D., & Lang, W. (2013). A method to estimate the accumulated delivery time uncertainty in supply networks. In *Dynamics in Logistics* (pp. 337–347). Springer.
10. Safaei, M., Mehraei, A., & Thoben, K. D. (2014). A computational method in analyzing of delivery time uncertainty for highly complex supply networks. *Measurement*, 55, 549–563.
11. Salani, M., Battarra, M., & Gambardella, L. M. (2016). Exact algorithms for the vehicle routing problem with soft time windows. In *Operations Research Proceedings 2014* (pp. 481–486). Springer.
12. Tanai, Y., & Guiffrida, A. L. (2015). Reducing the cost of untimely supply chain delivery performance for asymmetric Laplace distributed delivery. *Applied Mathematical Modelling*, 39(13), 3758–3770.
13. Vanany, I., Zailani, S., & Pujawan, N. (2009). Supply chain risk management: literature review and future research. *IGI Global*, 2(1), 16–33.

Chapter 15

Two-Step Approach for the Multi-Objective Container Assignment Problem with Barge Scheduling



F. Phillipson

Abstract In this chapter we look at the problem of an LSP controlling the means of transport and responsible for the assignment of orders to these means. Here the demand, i.e., the arrival of orders, is uncertain until the moment of unveiling. The problem we consider falls in the fourth quadrant of Fig. 1.2. The combination of uncertainty or stochasticity and the huge number of controllable items makes the problem difficult to solve. That is why we propose to split the problem in two parts, first creating a general schedule and thereafter assigning containers to trains and barges following the schedule or assign them to a truck. We do not benchmark the solution of the method. However, we show that the approach of the assignment gives an improvement in comparison with an ‘Ad hoc’ method, which is considered a proxy for a human planner’s approach.

Introduction

We consider a network of terminals, where a Logistic Service Provider (LSP) has to ship containers between those terminals, using various modalities. A number of those terminals are owned by the LSP, who can use these terminals to store the containers temporarily or as transshipment location. The LSP has his own trucks, barges and trains and can control them at any time. The arrival of a container at a terminal (from outside the system) is only known a number of days before the shipment has to be realised. For this we define the following times:

- Release time: moment the information about a container arrives at the LSP.
- EPU: Earliest Pick-up time: from this moment the container is available for pick-up at the terminal.
- LPU: Latest Pick-up time: the container has to be picked-up before this moment.
- EDT: Earliest Delivery Time: the earliest moment the container can be delivered at the terminal of destination.

F. Phillipson (✉)
TNO, The Hague, The Netherlands

- LDT: Latest Delivery Time: the latest moment the container can be delivered at the terminal of destination.

The LSP wants to optimise a combination of multiple objectives: costs, CO₂ and lateness and has to decide on the planning of the means of transport and the assignment of the containers. Lateness here is the number of containers delivered or collected outside of the pick-up and delivery intervals. Following the notation of [5] we have the following problem:

$$D2R, RO, RD, RC, RDT | DRD | \cdot | \cdot | social | 1$$

There are two important aspects in logistic optimisation problems: the presence of uncertainty and the scope of the optimisation. The uncertainty can be in many parts of the logistic system, as shown in [13]. Uncertainty can be in demand, supply or arrival of goods at the client, availability of resources and within the transportation process, think of travel times, failures in equipment etcetera.

For the scope of the problem both assignment of goods to modalities can be considered, as the operations of the vehicles (routes, departure times, etc.). The latter part is often known as service network design [4], which has a part that is not flexible at all, think of the location of (rail) roads and water ways, and a more flexible part like time tables and routing. The latter part can also be taken into account at the (tactical) service network design phase, but especially in synchromodal logistic problems, this is often taken into account during the (more) operational planning phase.

In Fig. 1.2 these elements are combined into 4 regions of problems. In the first region the events or orders to be assigned are not uncertain (fixed) and the infrastructure (vehicles) has fixed schedules. These are common assignment or planning problems, examples can be found in [12, 14, 15, 20]. In problem 2, uncertainty or stochasticity is added, making it a, more complicated, problem of assignment or planning under uncertainty, as shown in [10, 21]. In the third problem both the orders as the infrastructure needs to be planned, which degrees of freedom gives a larger problem to be solved. Examples of this approach can be found in [2, 16, 19]. The fourth problem brings uncertainty to the third problem. This problem is discussed in [17, 22].

The problem we consider falls in the fourth quadrant. The combination of uncertainty or stochasticity and the huge number of controllable items makes the problem difficult to solve. That is why we propose to split the problem in two parts:

1. Create a general schedule for trains and barges.
2. Assign containers to trains and barges following the schedule or assign them to a truck.

For the first problem we propose here a new approach. The problem in the first part is to construct a schedule to ship the containers between the terminals, where we can use some terminals as temporary storage or transshipment facility. For a specific

day and a specific demand matrix, where this matrix indicates how many containers need to be shipped from a certain terminal to another terminal, this problem can be seen as a ‘Vehicle Routing Problem with Pick-ups, Deliveries and Transshipments’ as defined in [3]. However, we build the schedule based on an average demand matrix per day, where we can combine the demand for a number of days, as long as the introduced waiting time at the terminals can be handled within the maximum delivery time, with a specific probability. This results in a ‘Multi-Day Recurring Vehicle Routing Problem with Transshipment and Probabilistic Constraints’. Note that transshipment and VRP with transshipment is also, very recently, mentioned in [6, 11]. We think that both of those two papers have a different definition of transshipment facility. In [11] transshipment means that if a store has a specific item not in stock, and the item can also not be delivered by the depot, then another store, that does have the item in stock, can ship the item to the first store. In [6] a transshipment facility is a local pick-up point where multiple customers can collect their goods, instead of delivering the goods to the customers’ houses. We mean with transshipment the same as [3], being a transfer point, where goods can be for a short amount of time until the next vehicle will collect them for further transport. Both papers [6, 11] propose (large) neighbourhood search methods. Also the review of [9] show that most papers on VRP with simultaneous pick-up and delivery use search based meta-heuristics, like tabu search (29%), local search (26%), neighbourhood search (23%) and ant colony systems (19%). We will propose a construction method, which are often faster than global search strategies.

For the second problem we propose an online optimisation method, based on intermediate optimisation routines. Incoming orders will be assigned in a greedy way at their release time, when the information of the incoming order is arriving in the system. Then, regularly, e.g., every day, all assignments that still can be changed are taken into an optimisation problem, executed through a simulated annealing approach.

In Sect. “[Schedule Construction](#)” the first problem is defined and the methodology is presented. Also an example use case is presented that will serve as running example in this work. In Sect. “[Container Assignment](#)” the second problem is defined and again the proposed methodology is presented and applied to the running example.

Schedule Construction

Methodology

We first try to create a recurring schedule that fulfils the demand of container shipments between the terminals. The schedule has to be such that an arbitrary container arriving at a terminal will reach its destination on time with a certain probability. We named this problem the ‘Multi-Day Recurring Vehicle Routing Problem with

Transshipment and Probabilistic Constraints'. We assume $v = 1, \dots, V$ types of vehicles, each with fixed costs P_v , variable costs p_v per kilometre and capacity c_v . Costs can be a combination of various (monetised) cost components, like OpEx (Operating Expenditure), CO₂, etc. We have $t = 1, \dots, T$ terminals, where the distance between terminal i and j equals $d_{i,j}$ and average number of containers to ship from terminal i to terminal j equals $D_{i,j}$. The set S is a subset of $\{1, \dots, T\}$ and is the set of transshipment terminals, owned by the LSP. All active connections are in set C . We now propose the following approach:

1. Assign a connection from all terminals to their nearest transshipment location: select for each $t \in \{1, \dots, T\} \setminus S : l_t = \arg \min_{s \in S} (d_{s,t})$.
2. Assign connections between transshipment creating a minimum spanning tree, using Prim's algorithm [7].
3. Assign a vehicle to each connection, such that costs are minimised and maximum delivery times are met. Calculate the maximum number of containers on the (return) trip for all connection $c \in C$, resulting in $N(c)$ and the length of this trip $l(c)$. The resulting vehicle type for this connection follows now from:

$$\arg \min_{v=1, \dots, V} \left(\frac{P_v + p_v l(c)}{\lfloor \frac{c_v}{N(c)} \rfloor} \right). \quad (15.1)$$

4. Repeat:

- Calculate costs of all combination of two barge movements to one.
- Check the feasibility of the combination.
- Select best (minimising costs, meeting delivery items) set of combinations such that all terminals are covered; the same procedure as in step 3 can be used to select the best combinations.
- Effectuate this best set.

5. Until no new cost saving can be found.

In step 4 a new question arises: when is a route (combination) feasible? In Eq. (15.1) the frequency of a connection is used: $\lfloor \frac{c_v}{N(c)} \rfloor$ indicates how often the barge of type v visits the terminal, given the total demand of the connection. We assume that a container that arrives has to wait an amount of time that follows a uniform distribution $[0, \lfloor \frac{c_v}{N(c)} \rfloor]$. Every time there is a change in connection for a container, this adds to the waiting time, combining this to an Irwin-Hall distribution [1] as also used in Chap. 14. We now assume a combination is feasible when the 90% percentile of the sum of all transportation and waiting times for all container origin-destination paths are such that the container will reach its destination on time.

Illustrative Example

Here an example use case is introduced that will be used throughout this work. First, in this example we use a number of terminals in the port of Rotterdam, and one in the hinterland of the Netherlands. We use the clustering of terminals into 5 regions (from [18]), as shown in Fig. 15.1. Region 6 is the hinterland location. We assume to have two types of barges, type 1 having a capacity (c_v) of 75 TEU and type 2 having a capacity of 150 TEU. The travel times (in time steps) between the regions are depicted in Table 15.1 and the average shipment in TEU per time step in Table 15.2. Terminal 3 and 6 are transshipment terminals, owned by the LSP. We assume $P_v = \{1000, 3000\}$ and $p_v = \{3, 5\}$.

Steps 1 and 2 now result in the following structure: there is a connection between region 3 and all other regions. In step 3 the analysis is done as shown in Table 15.3. For each connection the frequency is calculated: what is the minimal number of time steps the barge should visit the terminals, such that, in expectation, the number of container is not exceeding the capacity. Then the cost for both types of barges can be calculated and the best option is shown in bold in Table 15.3. This results in the intermediate solution where connections 1, 2, and 3 each have a type 1 barge and make a tour every 5, 2 and 1 time steps, respectively. Connection 4 uses a type 2 barge every 3 time steps and connection 5 makes a tour every time step with a



Fig. 15.1 Clustering of terminals within the Port of Rotterdam into 5 regions, as proposed in [18]

Table 15.1 Travel times in time steps between regions 1 to 6

From	To					
	1	2	3	4	5	6
1	0	2	4	3	4	24
2	2	0	2	4	6	22
3	4	2	0	2	4	20
4	6	4	2	0	2	22
5	8	6	4	2	0	22
6	24	22	20	22	24	0

Table 15.2 Average shipment $D_{i,j}$ in TEU per time steps between regions 1 to 6

From	To					
	1	2	3	4	5	6
1	0	0	1	2	3	8
2	1	0	10	0	0	10
3	1	11	0	6	5	5
4	3	0	6	0	0	37
5	0	0	5	0	0	37
6	3	13	10	39	33	0

Table 15.3 Outcome of step 3

c	Trajectory	$\lfloor \frac{c_1}{N(c)} \rfloor$	$\lfloor \frac{c_2}{N(c)} \rfloor$	Distance	Costs ($v = 1$)	Costs ($v = 2$)
1	1-3	5	10	80	248	340 (NF)
2	2-3	2	5	40	560	640 (NF)
3	3-4	1	2	40	1120	1600
4	3-5	1	3	80	1240	1133
5	3-6	0	1	400	-	5000

Table 15.4 Combination of first iteration of step 4

Combination $c_1 - c_2$	Costs 1 ($v = 1$)	Costs ($v = 2$)	Cost original	Saving
1-2	620	850	808	-188
1-3	1360	1800	1368	-8
1-4	1480	1900	1381	99
2-3	-	3400	1680	1720
2-4	1360	1800	1693	-333
3-4	-	3400	2253	1147

Table 15.5 Combinations of route combination in first iteration of step 4

Combination	Total cost solution
1-2 3-4	7873
1-3 2-4	7720
1-4 2-3	8061

type 2 barge. Not the multiple barges can be used per connection if the travel times are longer than the frequency of the connection. Not that the solutions with type 2 barges for connections 1 and 2 are not feasible, due to the probabilistic travel time restrictions.

In the first iteration of step 4, we now try to combine routes of the assigned barges. In Table 15.4 the costs of the combinations per barge types are depicted and the costs of the current situation. We see here that the combination of connections 1 and 2 leads to a saving of 188 when a type 1 barge is used. A combination using a type 2 leads to an increase of costs. To prevent us from actions that would cause local optimality, we look at combinations of these route combinations. We see in Table 15.5 that the second combination yields the best solution, and that

Table 15.6 Outcome of first iteration of step 4

<i>c</i>	Trajectory
1	1-3
2	3-4
3	3-6
4	5-2-3

Table 15.7 Outcome of second iteration of step 4

<i>c</i>	Trajectory
1	3-6
2	5-2-3
3	4-1-3

combination 2 – 4 is the combination within that solution that yields the highest cost saving. Thus, this step results in the solution depicted in Table 15.6.

In the second iteration only the combinations 1–2 and 1–4 give feasible solutions. Combinations of those two combinations are not feasible. This means we can just choose the best combination, resulting in the solution as depicted in Table 15.7.

Now, no new feasible combinations can be realised. Here the algorithm stops. Note that the proposed solution, combine regions 2 and 5 in one tour and regions 1 and 4 in the other tour is not the solution that one expects at first sight. A full tour, visiting all regions in the Port of Rotterdam area, or combining regions 4 and 5, on the one hand, and regions 1 and 2, on the other, would be more obvious, looking through the eyelashes to the problem.

Container Assignment

Given a schedule that fits the expected demand best, as defined in the previous section, we now look at the assignment of containers to barges and trucks. For this an online optimisation method is constructed. Online optimisation is a field of optimisation theory that deals with optimisation problems having no or incomplete knowledge of the future. These kind of problems are denoted as online problems and are seen as opposed to the classical optimisation problems where complete information is assumed, the so-called offline optimisation. In general three approaches can be distinguished:

1. Rule based: fixed rules are used to update the planning when a new container or in general new information arrives.
2. Ad hoc: the best allocation is done when a new container or in general new information arrives. The plan is not changed otherwise.
3. Decision window optimisation: an optimisation procedure is run over that part of the planning that is still allowed to change.

As indicated in Sect. “[Introduction](#)”, the LSP wants to optimise a combination of multiple objectives, e.g., costs, CO₂ and lateness. Lateness is defined as the number of containers delivered outside the pick-up and delivery intervals. This defines a multi-objective optimisation problem in the form of:

$$\max \sum_{i,j} w_{ij} KPI_{ij}, \quad (15.2)$$

where w_{ij} is the weight customer i gives to KPI (Key Performance Indicator) j and KPI_{ij} is the score of all orders of client i on KPI j .

Here we propose an online algorithm that combines an ‘Ad hoc’ approach with ‘Decision window optimisation’: orders (one or more containers) are assigned to barges and truck in an ad hoc manner, such that it minimises the incremental costs, as defined in Eq. (15.2). This assignment is done at the release time (see Sect. “[Introduction](#)”) of the order. Then, regularly, e.g., every day, a ‘Decision window optimisation’ is performed on all orders that are still adjustable, meaning the current time is between their release time and the moment that they are shipped. This results in the following approach:

1. Build paths using a breadth first search, assuming an ordered list of transportation movements.
2. Add truck option.
3. For all paths from origin to destination: add costs.
4. Determine best assignment under capacity constraints.
5. Repeat step 1–4 for all containers per time step.
6. Perform ‘Simulated Annealing’ [8] for all adjustable assignments.

In Step 1 an ordered list of transportation movements is expected. This is the schedule from the previous section, divided into the individual stages between terminals (or regions). For a specific order, we now select all movements that start after the EPU of the order and bring the container one step further (in all possible directions). In the next iterations, each of these movements can be extended by a next step, creating a path, if the departure time and origin match the first step. In this way a tree grows organically building paths from the origin location to all other locations. Each path ends when the destination of the order is reached or when a specific number of movements is exceeded.

We will explain this with the help of an example. Say the order under consideration has to be transported from terminal 4 to terminal 6 and is available for shipping (EPU) at $t = 2$, having the list of transportation movements available as shown in Table 15.8.

After the first iteration we now have the transportation movements with ID 1. In iteration 2, a new movements is added to obtain the path 1 – 3, those two moves fit together. This path says that the order is shipped at $t = 3$ from group 4 to group 1 and arrives there at $t = 9$. Now at $t = 10$ it departs from group 3 to group 1 and arrives there at $t = 14$. In iteration 3, moves 5, 6 and 7 are added to the move

Table 15.8 Example list of transportation movements

ID	Origin	Destination	t_{start}	$t_{arrival}$
1	4	1	3	9
2	5	2	4	10
3	1	3	10	14
4	2	3	11	13
5	3	5	14	18
6	3	4	15	17
7	3	6	18	38
8	6	3	39	59

Table 15.9 Example full tree of paths, depicting combinations of IDs of transportation movements from Table 15.8

Paths		
1		
1	3	
1	3	5
1	3	6
1	3	7

from iteration 2, each fitting after move 3. This results in the total list as shown in Table 15.9.

Here only 1-3-7 fits the requirement of the order. In the case multiple moves are available, all are scored on the KPIs (step 3) and a truck option is added (step 2), also scored on the KPIs (step 3). Now the best option is selected (step 4) and the assignment is registered. At the end of the time-unit, a simulated annealing approach is used to optimise over all assignments that still can be changed. As input, all the options are provided to the simulated annealing approach.

One issue in using online methods, and especially in ‘Ad hoc’ methods, is the deployment of a new item of transportation. In most cases, an less a less good allocation is preferred over allocation to, e.g., a new barge. However, the increased costs by this deployment do not have to be assigned totally to this one assignment. To overcome this, in our implementation we used average costs for a specific modality in the ‘Ad hoc’ assignment. Here, however, there is a danger that too many means of transport will be used, each only partly used. In the ‘Decision window optimisation’ we used the real cost structure.

For the running example, the efficiency of the ‘Decision window optimisation’ step is obvious. We ran the approach with and without step 6. This means that in one run only the ‘Ad hoc’ method was used, which should be a proxy for a human planner. In another run, also the ‘Decision window optimisation’ step was used. We see in Fig. 15.2 that this optimisation step is able to remove almost all truck movements from the planning, also lowering the CO₂ levels. Even the number of used barges is lower. However, this comes at a cost: the lateness is worse in the optimised case. The overall, weighted cost combination is in the optimised run lower.

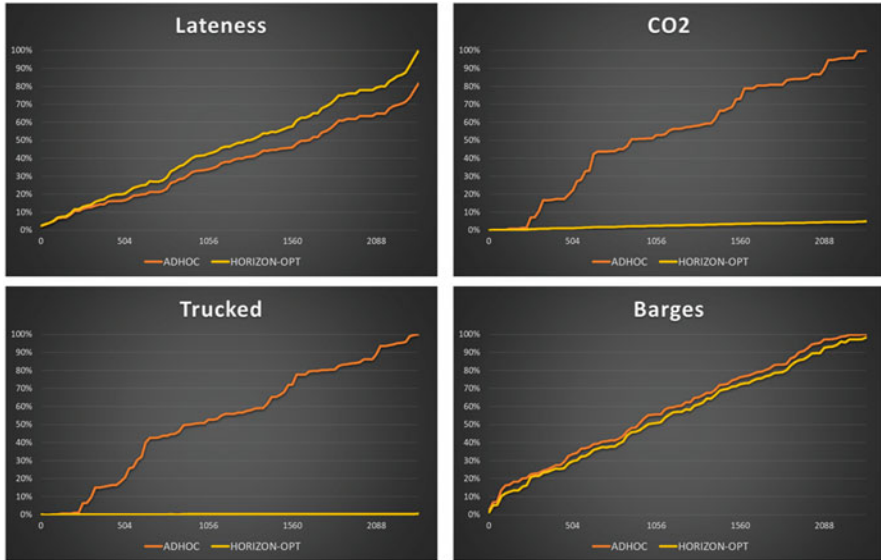


Fig. 15.2 Comparison of two approaches, the ‘Ad hoc’ method and the full approach including the ‘Decision window optimisation’

If we change the weights, for example, to a (financial) costs only optimisation, the algorithm makes other choices. The number of trucks is even further decreased (and the CO₂ levels as a result of this), again at the expense of the lateness as depicted in Fig. 15.3.

Conclusion

In this chapter we looked at the problem of an LSP controlling the means of transport and responsible for the assignment of orders to these means. Here the demand, i.e., the arrival of orders, is uncertain until the moment of unveiling. The problem we consider falls in the fourth quadrant. The combination of uncertainty or stochasticity and the huge number of controllable items makes the problem difficult to solve. That is why we propose to split the problem in two parts, first creating a general schedule and thereafter assigning containers to trains and barges following the schedule or assign them to a truck. We did not benchmark the solution of the method. However, we showed that the approach of the assignment gives an improvement in comparison with an ‘Ad hoc’ method, which is considered a proxy for a human planner’s approach.

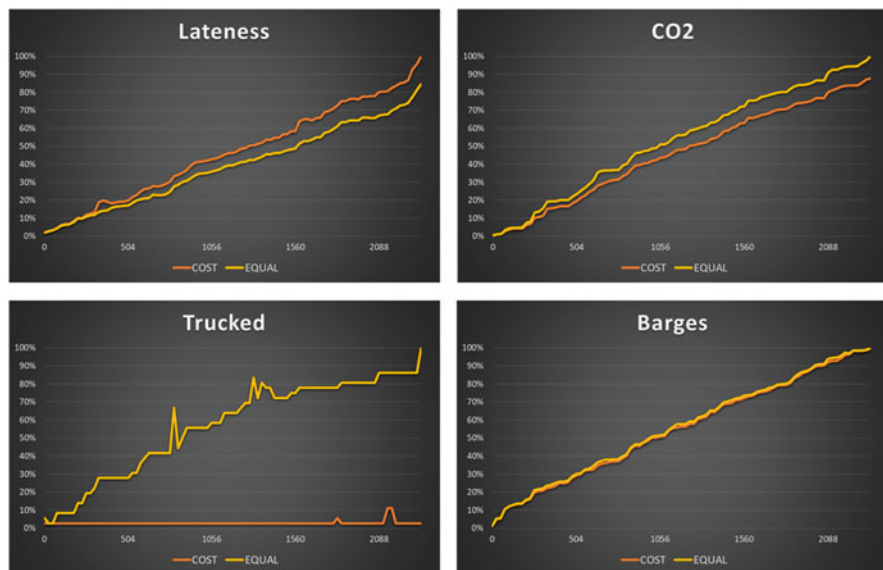


Fig. 15.3 Comparison of two different ways of weighting the objectives: equal weights (EQUAL) and (financial) costs only (COST)

References

1. Batsyn, M., & Kalyagin, V. (2013). An analytical expression for the distribution of the sum of random variables with a mixed uniform density and mass function. In *Models, Algorithms, and Technologies for Network Analysis* (pp. 51–63). Springer.
2. Behdani, B., Fan, Y., Wiegman, B., & Zuidwijk, R. (2016). Multimodal schedule design for synchromodal freight transport systems. *European Journal of Transport and Infrastructure Research*, 16(3), 424–444.
3. Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1), 1–31.
4. Crainic, T. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2), 272–288.
5. De Juncker, M. A., Huizing, D., del Vecchio, M. O., Phillipson, F., & Sangers, A. (2017). Framework of synchromodal transportation problems. In *International Conference on Computational Logistics* pp. 383–403. Springer.
6. Friedrich, C., & Elbert, R. (2022). Adaptive large neighborhood search for vehicle routing problems with transshipment facilities arising in city logistics. *Computers and Operations Research*, 137, 105491.
7. Greenberg, H. J. (1998). *Greedy algorithms for minimum spanning tree*. Denver: University of Colorado.
8. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680.
9. Koç, Ç., Laporte, G., & Tükenmez, İ. (2020). A review of vehicle routing with simultaneous pickup and delivery. *Computers and Operations Research*, 122, 104987.

10. Kooiman, K., Phillipson, F., & Sangers, A. (2016). Planning inland container shipping: A stochastic assignment problem. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications* (pp. 179–192). Springer.
11. Leelertkij, T., Parthanadee, P., & Buddhakulsomsiri, J. (2021). Vehicle routing problem with transshipment: mathematical model and algorithm. *Journal of Advanced Transportation*, 2021, 1–15.
12. Li, L., Negenborn, R. R., & De Schutter, B. (2016). Distributed model predictive control for cooperative synchromodal freight transport. *Transportation Research Part E: Logistics and Transportation Review*, 105, 240–260. <https://doi.org/10.1016/j.tre.2016.08.006>. <http://www.sciencedirect.com/science/article/pii/S1366554515303069>.
13. Li, L., & Schulze, L. (2011). Uncertainty in logistics network design: a review. In *Proceedings of the International MultiConference of Engineers and Computer Scientists* (Vol. 2).
14. Lin, X., Negenborn, R. R., & Lodewijks, G. (2016). Towards quality-aware control of perishable goods in synchromodal transport networks. *IFAC-PapersOnLine*, 49(16), 132–137.
15. Mes, M., & Iacob, M. (2016). Synchromodal transport planning at a logistics service provider. In *Logistics and Supply Chain Innovation* (pp. 23–36). Springer.
16. Nabais, J. L., Negenborn, R. R., Benitez, R. B. C., & Botto, M. A. (2013). A constrained MPC heuristic to achieve a desired transport modal split at intermodal hubs. In *2013 16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)* (pp. 714–719). IEEE.
17. Pérez Rivera, A., & Mes, M. (2016). Service and transfer selection for freights in a synchromodal network. *Lecture Notes in Computer Science*, 9855, 227–242.
18. Pruijn, S. (2018). *An algorithm for scheduling containers on barges*. B.S. thesis, University of Twente.
19. Riessen, B. V., Negenborn, R. R., Dekker, R., & Lodewijks, G. (2015). Service network design for an intermodal container network with flexible due dates/times and the possibility of using subcontracted transport. *International Journal of Shipping and Transport Logistics*, 7(4), 457–478.
20. Theys, C., Dullaert, W., & Notteboom, T. (2008). Analyzing cooperative networks in intermodal transportation: a game-theoretic approach. In *Nectar Logistics and Freight Cluster Meeting, Delft, The Netherlands* (pp. 1–37).
21. Xu, Y., Cao, C., Jia, B., & Zang, G. (2015). Model and algorithm for container allocation problem with random freight demands in synchromodal transportation. *Mathematical Problems in Engineering*, 2015. <https://doi.org/10.1155/2015/986152>.
22. Zhang, M., & Pel, A. (2016). Synchromodal hinterland freight transport: model study for the port of Rotterdam. *Journal of Transport Geography*, 52, 1–10.

Chapter 16

A Robust Optimisation Approach to Sychromodal Container Transportation



I. Chiscop

Abstract This chapter addresses sychromodal planning at operational level from the perspective of a logistics service provider and studies an optimisation problem with simultaneous vehicle routing and container-to-mode assignment, having uncertain data. It is assumed that the release times of the containers belong to an uncertainty interval, and no further statistical information is available. This problem belongs to the fourth quadrant of Fig. 1.4. The container routing problem is modelled as a mixed integer program with explicit time variables and lateness penalties. A robust formulation is then proposed to eliminate the uncertain parameters from the objective function and constraints. By solving the new model exactly, with the aid of an optimisation solver, robust solutions are obtained corresponding to transportation plans which remain feasible for any realisation of the release times within the pre-specified uncertainty interval. In order to introduce some flexibility in the transportation plan, the continuous time variables are modelled as affine functions of the uncertain parameters. The resulting two-stage decision model is tested for a small-sized instance in both situations, with high and low lateness penalties.

Introduction

Sychromodal transportation can be studied from multiple perspectives. There are several agents acting in the transportation network, each with their own modes and terminals/warehouses but sharing the existing infrastructure. Although sychromodality entails collaboration between all these parties, this is not always the case. Therefore, it is necessary to understand how much information is actually available and shared, and what kind of optimisation objectives are desired. The information within the network is available globally or locally. If the information

I. Chiscop (✉)
TNO, The Hague, The Netherlands
e-mail: irina.chiscop@tno.nl

is locally available, it means that only the agents themselves know, for example, where they are or what their status is at a certain time. If the information is global, this information is also known to the network operator, to all other agents or both. Furthermore, if all agents need to be individually optimised, the optimisation objective is local. If the optimisation objective is global, the best option for the entire network is the desired outcome.

The logistics service provider (LSP) whose activity is serving as a case study in this chapter is interested in reducing its own overall costs but has certain knowledge of the other agents in the network. This corresponds to a selfish approach to synchronomodality as described in Chap. 1 and illustrated in Fig. 1.2. Given these facts, the following question arises: how can the logistics service provider optimally plan his transportation activities in order to minimise the associated costs? By investigating the characteristics of the problem further, we can develop this question into a proper research inquiry. In the following subsection we give a description of the practical setting behind the activity of the LSP and identify the optimisation problem in their planning process based on the information that was made available for us.

Use Case

Practical Setting

A logistics service provider is a company that uses its resources to offer and perform transportation services of goods from origin to destination. The company usually manages the goods being transported along the entire way and is responsible for storage and handling. In our case, the LSP has a few inland terminals and one warehouse. Moreover, it has a fleet of trucks and several kinds of chartered barges for transporting containers between the deep-sea or inland terminals and different customer warehouses in the hinterland. These barges may have different capacities. For instance, the larger ones may transport up to 156 TEU in three layers.

The LSP receives transportation requests from customers on a daily basis. These requests consist of one or more standardised containers to be picked up at a terminal and then transported either to another terminal or to the customer's warehouse. The transport between terminals is usually carried out by barge and, when this is not possible, by truck. The way in which these orders are handled within the LSP administration can be visualised in Fig. 16.1.

When a transportation order is received by the LSP the amount of information accompanying it may vary. In general, the destination and due date, namely the latest time at which the containers should arrive at their destination, are always specified. Moreover, the terminal from where these containers should be picked up, the time at which they are available for pickup and the shipping company may be indicated. However, this is not always the case. If the pickup location is known,

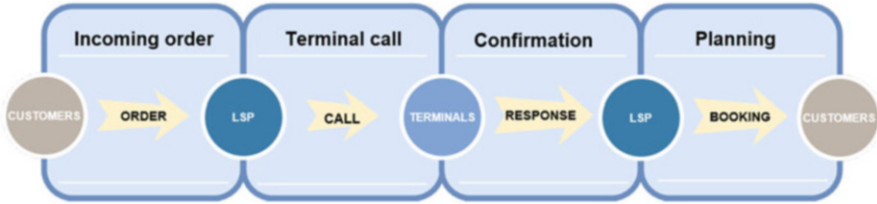


Fig. 16.1 Administration of a transportation request at the LSP

then the planners of the LSP will make a call towards that particular terminal in order to request a date and time-slot for the pickup. Depending on the working volume and the number of vessels to handle, the terminal may either confirm the proposed appointment, confirm the appointment on a different date, or not confirm an appointment at all. It is worth mentioning that the last two scenarios occur quite often in practice. Depending on the particular terminal, the time difference between the requested time and the confirmed time, otherwise known as the planning delay, can reach up to 10 days. After a response has been received from the terminal, then the LSP planners need to evaluate the current positions and loads of the available barges and decide which one will execute the pickup and when, and inform the customer about this. This process is difficult and the resulting plan is often subjected to change due to the uncertain elements in the network. The planner aims to schedule the available barges in such a way that all containers are picked up on time, then timely delivered to their destinations with a minimum amount of costs. These costs emerge from the usage of transportation modes, stationing at the terminals before the actual handling of containers and the eventual failure of meeting the due dates at the customers.

Our goal is to make use of all the practical information available in order to formulate an optimisation problem. Therefore, we need to further elaborate on what kind of elements are influencing the planning and what information is available to a planner at the moment that a decision must be made. To achieve this, we employ the framework for synchronodal problems developed in Chap. 2, where is distinguished between *resources* and *demand elements*. Intuitively, the resource elements refer to the available transportation modes, namely barges or trucks, whilst the demand elements consist of freight containers.¹ The features of these elements may be:

- *controllable*: Since we are discussing a decision problem, at least one element of the system must be in control. This can be, for instance, the allocation of demands to resources.
- *fixed*: A fixed element does not change within the scope of the problem.

¹ In this chapter the demand elements will always correspond to one container.

- *dynamic*: A dynamic element might change over time or due to a change in the state of the system (e.g., the amount of containers changes the travel time of a barge), but this change is known or computable beforehand.
- *stochastic*: A stochastic element is not necessarily known beforehand. For instance, it is not known when transportation orders will arrive, but the arrivals occur according to a Poisson process.
- *irrelevant*: It might occur that for certain problems not all elements are taken into consideration to model the system. Then these elements are irrelevant.

We will closely follow the classification in of Chap. 2 to describe all the elements occurring in the planning process of the LSP. However, not all elements encountered in the practical setting are encompassed by this framework.

Resource Elements

- *Resource type*: In this study, the LSP owns a fleet of barges of different known capacities and a uniform fleet of trucks. One may distinguish here between owned and subcontracted resources.
- *Resource features*: The resource capacities are fixed. The schedules of the barges and trucks are not fixed. Therefore the resource origin and resource destination are controllable elements. However, the resource departure time, resource travel time and resource arrival time are not controllable. This is a consequence of the delays which may occur either when receiving a confirmation from the terminal, or at the terminal itself, when the handling time takes longer than expected (this can happen due to a crane malfunction, for instance). We will classify these elements outside the framework as *uncertain*, since there is no information available concerning their distribution. Finally, we also have a resource price. Here we can distinguish between the price for employing a certain resource which is a fixed amount (per day, for instance) and the price for handling services provided at the terminals. The latter depends on the load to be handled, which is an uncertain element at the beginning of the planning period.
- *Terminal Handling time*: This refers to time required to handle different types of modes at the terminal. It includes both the waiting time and the time allocated for loading/unloading containers. This is also an *uncertain* element since there exist incoming orders which do not specify the pickup time or locations. For instance, it may be the case that a barge is waiting at a terminal to pick up some containers which have not arrived there yet.

Demand Elements

- *Demand type*: The LSP under study can transport containers of different sizes, of either 1 TEU or 2 TEU in load. Therefore, this element is fixed.
- *Demand-to-Resource allocation*: The assignment of containers to barges is essentially a decision that a planners have to make. Therefore, it is a controlled element.
- *Demand features*: The destination of a container, as well as its volume (in TEU) and due date at the customer to whom it belongs, are fixed elements. The demand

origin (pickup terminal of a container) and its release date (moment in time at which it can be loaded on a barge) are uncertain elements. This uncertainty emerges from the missing data in the transportation order, as customers simply do not specify it.

- *Demand Penalty*: This term refers to costs that are incurred when the due date at the destination for a container is not met. Since these costs are in general customer-dependent, we can classify this element as dynamic.

The resource and demand elements described are the main input for creating a schedule for the barges and trucks. However, the planning process does not only rely on the information that is available but also rely on the moment at which this information becomes available. At the beginning of the planning period, the planner knows the exact locations of all the barges and trucks in the fleet, their capacity and has a list of orders with specified destinations and due dates to be picked up sometime in the next 9 days. Moreover, at every moment in time, a planner has an estimation of the maximum and average delay of the deep-sea terminals (based on historic data in the last 30 days). This is the initial amount of knowledge. As time progresses, more information becomes available. That is, pickup locations along with release times of containers are revealed, and terminals send confirmation for appointment times. Moreover, new transportation orders may come in, which are also required to be executed within the next 9 days. This information can become available at any time so the planner must create a schedule that can handle real-time switches.

Given this practical setting, one may formulate the decision making of the LSP planners as an optimisation problem in which a routing of transportation modes and an assignment of the containers to modes must be provided under uncertain data in such a way that the total delay and costs are minimised.

Base Instance

In order to be able to develop a mathematical model and later on explore solution methods, we consider the following simplified instance obtained by reducing the size of the real-life problem and introducing some assumptions. The network comprises the following elements:

- 2 customers denoted C_1, C_2 : Their physical location is known and it is accessible only by truck.
- 2 deep-sea terminals denoted T_1, T_2 : Deep-sea vessel arrives here and unloads the containers that belong to the two customers.
- 1 container terminal operated by the LSP denoted T : Barges leave from here and go to the deep-sea terminals to pick up containers.
- 1 hinterland terminal operated by the LSP denoted D : It is the central terminal of the LSP, closest in distance to any customer (Fig. 16.2).



Fig. 16.2 Geographical display of the network

We notice here that there is one main difference between the container terminal and the hinterland terminal of the LSP. The container terminal is located in the port, nearby deep-sea terminals. On the other hand, the hinterland terminal is situated further away on the continent, in the proximity of customers. This is illustrated in Fig. 16.2.

The LSP has the following resources:

- *3 barges*: All with capacity of 20 units. Two of the barges at the terminal T whilst the other one is situated at the central terminal D . There is a fixed cost per kilometre² travelled by a barge.
- *unlimited trucks*: All with capacity 1. There is a fixed cost per kilometre travelled by a truck.

Suppose we are given two transportation orders with the following specifications:

1. Customer C_1 asked the LSP to pick up 30 containers from T_1 . The terminal has confirmed a time window for the pickup: $[10, 11]$.³ These containers have an uncertain release time. They will be simultaneously released sometime in the interval $[10, 11]$. This order needs to arrive at the customer by time unit 20.
2. Customer C_2 has 10 containers to be picked up from terminal T_2 . This terminal has also confirmed a time window for the pickup: $[15, 16]$. All 10 containers are already available. This order needs to arrive at the customer warehouse by time unit 20.

When developing this base model we have made several assumptions. We discuss them and their relation with the real practical setting below.

- The planning period starts at midnight or otherwise interpreted, at time step 0 and covers one full day, until time step 24, respectively.

² We will elaborate on transportation costs of barges and trucks later in the chapter.

³ We will take a time unit as being 1 h. Therefore, regard this interval as the time between 10:00 and 11:00.

- We assume fixed time windows at the deep-sea terminals. In practice we saw that a terminal can either answer an appointment call or not. In this scenario, we assume that we have confirmed appointment calls at the beginning of the planning period.
- If a barge arrives either too early or too late at a deep-sea terminal, it can be handled right away. So we assume that there is no waiting time involved.
- We assume that there is no handling time.
- Once it has been loaded, a barge may leave the deep-sea terminal right away.
- At any point in time, there are trucks available at every terminal, which can transport the released containers to other locations.
- There is a waterway connecting the terminals. The customers' warehouses can only be reached by truck.
- The travel times in between any two locations of the barges and trucks are known.

Given this simple instance, we are interested in minimising the overall costs and the total delay at the customers. In order to maintain a uniform objective, we can associate costs with the delay in such a way that the final objective will represent the costs overall. This simple instance will serve as a starting point in developing a mathematical model that determines an assignment of containers to transport modes, and also a specific routing of the containers. Whilst this base model is not of any practical relevance, it will serve as a basic tool to understand, and later on, to incorporate more complex features of the transport network.

After analysing the base instance, we understand that our choice for modelling approaches is somewhat restricted by the lack of probabilistic knowledge. In this case, we will study the container routing problem from a robust perspective. In other words, since we cannot employ stochastic models, we will look at robust optimisation techniques.

Deterministic Problem Formulation

In this chapter we present a mathematical model for the freight routing problem described as our base instance. We will use Sharypova's model [9] as the basis of our research and further develop it to incorporate all aspects which are of interest in the context of uncertain parameters in the transportation network. We describe all the modifications brought to the original mixed integer linear program and elaborate on a further extension of the model that can be used to incorporate multiple trips of a vehicle to a certain location.

Deterministic Model

Sharypova's model [9] serves as a starting point in our problem formulation. This model provides a transportation schedule of minimum cost which meets the strict delivery deadlines at the destinations of commodities. Since our goal is to investigate the impact of uncertainty on the transportation plan, it is reasonable to allow for more flexibility in the network, namely replace the strict deadlines of commodities by the so-called *soft due times*. This implies that a commodity may arrive later than its due date at its destination, in which case a penalty cost is incurred. To model this aspect, we introduce *lateness* decision variables L_i^m , describing how late vehicle m is at location i . Clearly, these variables are only defined when location i is a destination node for some commodity. Moreover, in our base instance we assumed that a barge arriving at a deep-sea terminal must wait a certain amount of time before it starts being loaded or unloaded. Thus, we will incorporate a terminal specific parameter w_i , representing the amount of time that a barge has to wait at terminal i . Finally, the components of the objective function must be addressed. For optimisation, the focus will be on time-related components. Generally speaking, we are interested in minimising the utilisation of trucks. However, in practice it is often the case that trucks and trailers are rented by the hour and motivated by this, we will aim for minimising the trucking hours. A trucking hour is 1 h in which a truck has been utilised for transportation purposes. It is important to remark here that the amount of trucks used overall or the time travelled by a truck without being loaded are not important quantities in this setting. Moreover, we complete our multi-objective function by incorporating the total lateness recorded in the planning. This term quantifies by how much time container arrivals differ from their specified due date. Weights are associated with each of the two objectives for scaling purposes. These weights allow us to prioritise one objective over the other one, as in practice one might often find that arriving an hour late at a location might be preferable to renting a truck for another hour. A thorough discussion on these weights will follow later. We will define the following here:

Sets:

V = set of locations

A = set of travelling arcs between locations

M = set of vehicles

K = set of commodities

V_m = set of locations that can be accessed with vehicle m

Decision variables:

$x_{i,j}^{k,m}$ = number of containers of commodity $k \in K$ transported from location $i \in V$
to location $j \in V$ by vehicle $m \in M$

$$y_{i,j}^m = \begin{cases} 1 & \text{if vehicle } m \in M \text{ travels from location } i \in V \text{ to location } j \in V \\ 0 & \text{otherwise} \end{cases}$$

$$z^m = \begin{cases} 1 & \text{if vehicle } m \in M \text{ is used in the transportation plan} \\ 0 & \text{otherwise} \end{cases}$$

$$\theta_i^{m,l} = \begin{cases} 1 & \text{if a transshipment occurs between vehicle } m \in M \text{ and vehicle } l \in M \text{ at location} \\ & i \in V_m \cap V_l \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_i^{k,m} = \begin{cases} 1 & \text{if any container of commodity } k \in K \text{ is loaded on vehicle } m \in M \text{ at node} \\ & i \in \{o(k), d(k)\} \\ 0 & \text{otherwise} \end{cases}$$

A_i^m = arrival time of vehicle $m \in M$ at location $i \in V$

D_i^m = departure time of vehicle $m \in M$ from location $i \in V$

L_i^m = lateness/arrival delay of vehicle $m \in M$ at location $i \in \{d(k) | k \in K\}$

$q_i^{k,m,l}$ = amount of containers of commodity $k \in K$ moved from vehicle $m \in M$
to vehicle $l \in M$ at location $i \in V_m \cap V_l$

Parameters:

s_i = service time at node $i \in V$

d_i^k = demand of commodity $k \in K$ at node $i \in V$

w_i = waiting time at terminal location $i \in V$

r_k = release time of commodity $k \in K$ at its origin

due_k = due time of commodity $k \in K$ at its final destination

$t_{i,j}^m$ = travelling time of vehicle $m \in M$ from node $i \in V$ to node $j \in V$

c^m = maximum capacity of vehicle $m \in M$

$(o(k), d(k))$ = origin-destination node pair of commodity $k \in K$

$\omega_{1,2}$ = weights of the objective functions

Given the previous decision variables and parameters, the model for container assignment and vehicle routing is ⁴:

$$\min \omega_1 \sum_{k \in K} \sum_{m \in M} \sum_{(i,j) \in A} t_{i,j}^m x_{i,j}^{k,m} + \omega_2 \sum_{m \in M} \sum_{i \in V} L_i^m$$

$$s.t. \sum_{(j) \in V^+(i)} y_{i,j}^m - \sum_{j \in V^+(i)} y_{j,i}^m = 0 \quad \forall m, \forall i \in V_m \quad (16.1)$$

$$\sum_{m \in M} \sum_{j \in V^+(i)} x_{i,j}^{k,m} - \sum_{m \in M} \sum_{j \in V^-(i)} x_{j,i}^{k,m} = d_i^k \quad \forall i \in V, \forall k \quad (16.2)$$

$$\sum_{k \in K} x_{i,j}^{k,m} \leq c^m y_{i,j}^m \quad \forall m, \forall (i, j) \in A \quad (16.3)$$

$$\sum_{k \in K} q_i^{k,m,l} > 0 \iff \theta_i^{m,l} = 1 \quad \forall m, l, \forall i \in V_m \cap V_l \quad (16.4)$$

$$\sum_{\mathfrak{x} \in V^-(i)} x_{j,i}^{k,m} = \sum_{l \in K} q_i^{k,m,l} \quad \forall m, k, \forall i \in V_m \setminus \{d(k)\} \quad (16.5)$$

$$\sum_{\mathfrak{x} \in V^+(i)} x_{i,j}^{k,m} = \sum_{l \in K} q_i^{k,l,m} \quad \forall m, k, \forall i \in V_m \setminus \{o(k)\} \quad (16.6)$$

$$\sum_{l \in K} q_i^{k,m,l} = 0 \quad \forall m, k, \forall i \in \{o(k), d(k)\} \quad (16.7)$$

$$\sum_{j \in V^+(i)} x_{i,j}^{k,m} > 0 \iff \tau_i^{k,m} = 1 \quad \forall m, k \forall i \in \{o(k)\} \quad (16.8)$$

$$\sum_{j \in V^-(i)} x_{j,i}^{k,m} > 0 \iff \tau_i^{k,m} = 1 \quad \forall m, k, \forall i \in \{d(k)\} \quad (16.9)$$

$$\theta_i^{m,l} = 1 \Rightarrow D_i^l - A_i^m - s_i \geq 0 \quad \forall m, l, \forall i \in V_m \cap V_l \quad (16.10)$$

$$y_{i,j}^m = 1 \Rightarrow D_i^m + t_{i,j}^m - A_j^m \leq 0 \quad \forall m, \forall (i, j) \in A \quad (16.11)$$

$$D_i^m \geq A_i^m + s_i \quad \forall m, \forall i \in V_m \quad (16.12)$$

$$D_i^m \geq r_k \tau_i^{k,m} \quad \forall m, k, \forall i \in \{o(k)\} \quad (16.13)$$

$$\tau_i^{k,m} = 1 \Rightarrow L_i^m \geq A_i^m - due_k \quad \forall m, k, \forall i \in \{d(k)\} \quad (16.14)$$

$$\sum_{j \in V^+(i)} y_{i,j}^m \leq z^m \quad \forall m, \forall i \in V_m \quad (16.15)$$

$$x_{i,j}^{k,m} \in \mathbb{N}_0 \quad \forall m, k, \forall (i, j) \in A \quad (16.16)$$

$$q_i^{k,m,l} \in \mathbb{N}_0 \quad \forall m, k, l, \forall i \in V_m \cap V_l \quad (16.17)$$

$$A_i^m, D_i^m, L_i^m \geq 0 \quad \forall m, \forall i \in V_m \quad (16.18)$$

$$y_{i,j}^m \in \{0, 1\} \quad \forall m, \forall (i, j) \in A \quad (16.19)$$

$$\theta_i^{m,l} \in \{0, 1\} \quad \forall m, l, \forall i \in V_m \cap V_l \quad (16.20)$$

⁴ In all cases where we say $\forall m, \forall l$ or $\forall k$ we mean $\forall m \in M, \forall l \in M$ and $\forall k \in K$.

$$\tau_i^{k,m} \in \{0, 1\} \quad \forall m, k, \forall i \in \{o(k), d(k)\} \quad (16.21)$$

$$z^m \in \{0, 1\} \quad \forall m \quad (16.22)$$

We recall that the objective is the weighted sum of trucking hours and total lateness. Constraints (16.1) ensure flow conservation at a location, while constraints (16.2) account for the demand requirement at the origin and destination of every commodity. Constraints 16.3 impose the capacity restriction of each vehicle. Constraints (16.4)–(16.7) regulate the occurrence of transshipment of containers from one vehicle to another depending on their current location. The inequalities in (16.8)–(16.9) assure that every commodity leaves its origin and arrives at its destination by means of some vehicle. The following four sets of inequalities (16.10)–(16.13) validate the time-related variables. Constraints (16.14) are of particular importance, as they establish the definition of lateness variables. Inequalities (16.15) provide the relation between used travel routes and the number of vehicles. Finally, the remaining constraints define the range of each decision variable.

The mixed integer program presented above describes a transportation problem which can be viewed as a complex extension of the capacitated vehicle routing problem with time windows (abbreviated as CVRP-TW). Since VRP is known to be NP-hard, we understand that there is no polynomial-time algorithm to solve the freight routing problem. Therefore, we expect that solving this problem even for small data instances with state-of-the-art optimisation solvers might require a considerable computational effort.

Additional Remarks

The mixed integer program described in the previous section has many binary and integer variables which makes it difficult to solve. Therefore, it is important to ensure that the solution space is as restricted as possible. In order to do so, we include the following *strong forcing* constraints:

$$x_{i,j}^{k,m} \leq \min\{D_k, C^m\} y_{i,j}^m \quad \forall (i, j) \in A, \forall k \in K, \forall m \in M,$$

where D^k is the demand of containers of commodity k , to be transported from their origin location to their destination. These constraints can be derived as flow cover inequalities and have been shown to be effective in improving the LP-relaxation of multi-commodity network design problems [6]. Therefore, they are added to the mixed integer program presented in the previous section.

A final remark concerns the modelling of trucks. Since a truck in general only has the capacity to transport one or two containers, it was preferable to not model them individually, as the size of the instance would have been too large. Instead, a number of trucks with very large capacity (set to 3000 in our instances) was enabled at every location. This number was set equal to the number of commodities to ensure that there is enough transport capacity for timely deliveries.

Robust Problem Formulation

In Chap. 4 we have presented a deterministic model for the freight routing problem which gives optimal solutions if the input data is assumed to be fully correct. However, in practice, this is almost never the case, as perturbations in data occur due to estimation, prediction or implementation errors. This sort of uncertainty may drastically affect the quality of the solution and it is not considered in deterministic optimisation. Nevertheless, it can be handled by stochastic optimisation (SO) and robust optimisation (RO). Stochastic programming is a commonly used method which optimises the problem by making use of the parameters' expected value. This approach generates a number of scenarios that represent the possible realisations of the stochastic parameters, assigns a probability to each of these scenarios and finally, creates a model optimising over all scenarios. Stochastic programming cannot be used when detailed statistical information is missing or when the number of scenarios becomes too large, making the problem intractable. In Chap. 4, it was shown that already for a small instance the freight routing problem with fixed vehicle routes and stochastic travel times, the scenario tree becomes prohibitively large. Although we will not investigate stochastic programming further, we refer the interested reader to [5]. The robust optimisation framework, on the other hand, is based on obtaining solutions which remain feasible for any realisation of the parameters within a predefined uncertainty set. For this reason, we will explore in this chapter how can the adjustable robust optimisation be used in order to deal with the uncertainty in the release times of the containers. We present the robust optimisation paradigm and explain how to formulate and solve the robust counterpart. The robust mathematical formulation of the freight routing problem is given at the end of this section.

Robust Optimisation Paradigm

Robust optimisation is an increasingly popular methodology to model mathematical optimisation programs with uncertain data. Instead of assuming a known probability distribution, the uncertain data is presumed to reside in a user-specified set of realisations, called the uncertainty set. We consider a general formulation of an uncertain linear optimisation problem:

$$\min_x \{c^T x : Ax \leq b\}, \quad (\text{P}_0)$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Suppose that the matrix A is uncertain and it belongs to a bounded uncertainty set $\mathcal{U}_A \subset \mathbb{R}^{m \times n}$. In a similar fashion we assume that right hand side vector b belongs to uncertainty set $\mathcal{U}_b \subset \mathbb{R}^m$, whilst the objective coefficients c reside in the uncertainty set $\mathcal{U}_c \subset \mathbb{R}^n$. The sets \mathcal{U}_A , \mathcal{U}_b and \mathcal{U}_c specify all possible realisations of the uncertain data and are collectively

referred to as the uncertainty set \mathcal{U} . The robust optimisation paradigm as described by Ben-Tal et al. [2] relies on the following assumptions:

- A.1** All decision variables $x \in \mathbb{R}^n$ represent here-and-now decisions: they should be assigned specific numerical values as a result of solving the problem before the actual data ‘reveals itself’.
- A.2** The decision maker is fully responsible for consequences of the decisions to be made when, and only when, the actual data is within the pre-specified uncertainty set \mathcal{U} .
- A.3** All the constraints of the uncertain problem in case are ‘hard’: we cannot tolerate violations of constraints when the data is in the uncertainty set \mathcal{U} .

These assumptions indicate what are the relevant feasible solutions of the linear uncertain problem P_0 . The first assumption **A.1** asserts that the solution vector should have fixed values or otherwise said, it should not contain any components to which there has not been assigned a numerical value. By assumptions **A.2** and **A.3** this solution vector should satisfy all the constraints, regardless of the realisation of the data in the uncertainty set \mathcal{U} . Such a solution is called *robust feasible* [2]. Thus we understand that robust optimisation is concerned with finding robust feasible solutions for problems with a predefined uncertainty set.

The Robust Counterpart

We observe that the linear uncertain problem P_0 exhibits uncertainty in all parameters. In fact, one can show that this problem can be re-formulated in such a way that only the matrix A will contain uncertain entries. Firstly, the uncertainty in the objective function can be removed by introducing an additional continuous decision variable $t \in \mathbb{R}$. Problem P_0 is then equivalent to:

$$\min_{x,t} \{t : c^T x - t \leq 0 \quad \forall c \in \mathcal{U}_c, \quad Ax \leq b \quad \forall A \in \mathcal{U}_A, \forall b \in \mathcal{U}_b\}.$$

Secondly, the uncertain components of vector b can be transferred to the matrix A in the following way: vector b is added as a column of A and value $x_{n+1} = -1$ is added as an extra component to the vector x . Then the problem P_0 can be written as:

$$\min_{x,t} \{t : c^T x - t \leq 0 \quad \forall c \in \mathcal{U}_c, \quad Ax \leq 0 \quad \forall A \in \mathcal{U}_A \cup \mathcal{U}_b\}.$$

Given these two reformulations, we conclude that it is always safe to assume that uncertain quantities occur only in the matrix of coefficients. This being said, we can finally give a most general form of the uncertain linear problem as:

$$\min_x \{c^T x : Ax \leq b \quad \forall A \in \mathcal{U}\}. \tag{P}$$

The robust reformulation of problem P is referred to as the *robust counterpart* (RC) problem [2] and we will present it as given in [10]. We assume that the coefficient matrix $A(\zeta)$ is an affine⁵ function of the uncertain parameter ζ :

$$\min_x \{c^T x : A(\zeta)x \leq b \quad \forall \zeta \in \mathcal{Z}\}, \quad (\text{RC})$$

where $\mathcal{Z} \subset \mathbb{R}^p$ denotes the user defined uncertainty set. Recall that a solution x is robust feasible if the constraints $A(\zeta)x \leq b$ are satisfied for every value of $\zeta \in \mathcal{Z}$. As discussed in [2], the robust counterpart of an uncertain linear optimisation problem with a certain objective is a ‘constraint-wise’ construction. In other words, the original i th row constraint $(Ax)_i \leq b_i \Leftrightarrow a_i^T x \leq b_i$, (with a_i being the i th row in A) from the nominal problem is replaced by $a_i^T x \leq b_i \quad \forall [a_i; b_i] \in \mathcal{U}_i$, where \mathcal{U}_i is the projection of \mathcal{U} on the space of data of i th constraint: $\mathcal{U}_i = \{[a_i; b_i] : [A, b] \in \mathcal{U}\}$. Therefore, we can address the uncertainty by a single constraint. For instance, we extract one constraint from the robust counterpart problem RC modelled as an affine expression in terms of ζ :

$$(a_i + P\zeta)^T x \leq b_i \quad \forall \zeta \in \mathcal{Z}, \quad (16.23)$$

where $a_i \in \mathbb{R}^n$ is interpreted as the nominal value of the data, $P \in \mathbb{R}^{n \times p}$ and $b_i \in \mathbb{R}$. The idea behind this process is to reformulate the robust counterpart constraint-wise in such a way that it becomes computationally tractable. The expression in (16.23) has infinitely many constraints due to the *for all* (\forall) quantifier and it is thus intractable in general. In [10] the authors provide a compact overview of the steps to be followed in order to remove this quantifier. We will closely follow their approach. Consider a polyhedral uncertainty set defined as:

$$\mathcal{Z} = \{\zeta : D\zeta + q \geq 0\}, \quad (16.24)$$

where $D \in \mathbb{R}^{m \times p}$, $\zeta \in \mathbb{R}^p$ and $q \in \mathbb{R}^m$.

In a worst-case reformulation, when the realisation of the uncertain data yields the largest objective value, one can re-write the nominal problem P as:

$$a_i^T x + \max_{\zeta} \{(P^T x)^T \zeta : D\zeta + q \geq 0\} \leq b_i. \quad (16.25)$$

By strong duality, the inner maximisation problem in the expression above can be replaced by its dual. Therefore, expression (16.25) is equivalent to:

$$a_i^T x + \min_w \{q^T w : D^T w = -P^T x, \quad w \geq 0\} \leq b_i. \quad (16.26)$$

⁵ A function $f : A \rightarrow B$ is affine if and only if the mapping $x \mapsto f(x) - f(0)$ is linear.

We see that in order to satisfy inequality (16.26), it suffices to find at least one w . Hence, the final formulation of the RC is given by:

$$\exists w : a_i^T x + q^T x \leq b_i, \quad D^T w = -P^T x, \quad w \geq 0, \quad (16.27)$$

which is an LP feasibility problem.

From everything that we have done so far, we conclude that solving the robust counterpart of a general linear optimisation problem with continuous variables and a polyhedral uncertainty set reduces to finding a feasible solution to the linear problem described in Eq. (16.27). Therefore, the robust counterpart of an uncertain linear program (LP) with a polyhedral uncertainty set is in fact a computationally tractable LP. Moreover, this property also holds for the so-called box uncertainty set of the form: $\mathcal{Z} = \{\zeta : \|\zeta\|_\infty \leq 1\}$, since the robust counterpart in this case is simply given by $a_i^T x + \|P^T x\|_1 \leq b_i$ [10]. For a thorough mathematical discussion on tractability properties of the robust counterpart for various uncertainty sets, the reader is referred to the book of Ben-Tal et al. [2].

Adjustable Robust Optimisation

The robust optimisation formulation given earlier is static in the sense that the numerical values of all decision variables must be determined before the uncertain quantities reveal their true value. For this reason, the solutions obtained by solving the robust counterpart are indeed robust feasible but sometimes very conservative: they are only optimal for the worst-case realisations of the uncertain data. With this static approach it may often be the case that the objective function of the solution becomes unnecessarily high given the actual data realisations attained in practice. This concept is also known as the *price of robustness*, described by Bertsimas and Sim [4] as the trade-off between the optimal solution and robustness. In order to achieve a reasonable price of robustness, the adjustable robust optimisation framework has been proposed [3]. In this framework, assumption **A.1** from the robust optimisation paradigm is relaxed, meaning that we allow for some *wait-and-see* decision variables. In other words, some decision variables can be adjusted at a later point in time according to the realisation of the data. Most commonly, these adjustable decisions are modelled as functions of the uncertain data. In view of this, the adjustable robust counterpart (ARC) can be formulated as:

$$\min_{x, y(\cdot)} \{c^T x : A(\zeta)x + By(\zeta) \leq b\} \quad \forall \zeta \in \mathcal{Z}, \quad (\text{ARC})$$

where $x \in \mathbb{R}^n$ represents the first stage here-and-now decision vector that is made before $\zeta \in \mathbb{R}^p$ is realised, $y \in \mathbb{R}^k$ denotes the second-stage wait-and-see decision vector that can be computed according to the realisation of ζ , and $B \in \mathbb{R}^{n \times k}$ is a given coefficient matrix. For the scope of this chapter it is sufficient to assume that the matrix B does not contain any uncertain elements. In general, it is difficult to

optimise over functions, so a commonly used approach is to express the adjustable decision variables as affine functions of the uncertain data, namely:

$$y(\zeta) = y_0 + Q\zeta. \quad (16.28)$$

In the expression above, $y_0 \in \mathbb{R}^k$ and $Q \in \mathbb{R}^{k \times p}$ are here-and-now decisions to be optimised by the model in the first stage. Substituting the expression for y given in Eq. (16.28) into the ARC we obtain the affinely adjustable robust counterpart (AARC):

$$\min_{x, y_0, Q} \{c^T x : A(\zeta)x + B y_0 + B Q \zeta \leq b \quad \forall \zeta \in \mathcal{Z}\}. \quad (\text{AARC})$$

Since the AARC is linear in both the decision variables and the uncertain parameter, it can be solved by following the same reformulation steps as in the previous section. Therefore, the AARC has the same tractability as the original robust counterpart, regardless of the uncertainty set chosen. Two important remarks are required here. First of all, the AARC might contain many more decision variables than the RC due to the size of matrix Q . Secondly, although the AARC will likely require more computational effort, the solution thus obtained will be at least as good as the one given by solving the RC.

Up to this point, we have presented both the static and the affinely adjustable robust counterpart problems and showed that in the case of linear programs with polyhedral or box uncertainty, both formulations are tractable. The fact that we can provide adjustable robust feasible solutions, makes the robust optimisation approach extremely appealing for further applying it to our freight routing problem. Nevertheless, the model developed includes many binary and integer variables for which the mathematical treatment is not directly applicable. Therefore, we will further discuss how robust optimisation techniques can be used in the context of mixed integer programs.

Robust Optimisation for Mixed Integer Programs

A mixed integer program is a mathematical program which contains both real valued decision variables and variables restricted to take integer values. It is well known that determining whether a feasible solution of a given mixed integer program with rational coefficients exists is in the class of NP-complete problems [7]. As such, we expect that a robust counterpart of a mixed integer linear program is also intractable. Consider the general form of a mixed integer program:

$$\min_{x, y} \{c^T x + d^T y : Ax + Gy \leq p, \quad x \in \mathbb{Z}^n, y \in \mathbb{R}^k\}, \quad (\text{MIP})$$

where $c \in \mathbb{Q}^n$ and $d \in \mathbb{Q}^k$ are given cost vectors, $A \in \mathbb{Q}^{l \times n}$ and $G \in \mathbb{Q}^{l \times k}$ are coefficients matrices and $p \in \mathbb{Q}^l$. We assume that the matrix A is the only element

affected by uncertainty. This assumption is motivated by our deterministic model in Chap. 4, in which the uncertain release time is multiplied with a binary variable in constraints (16.14). Thus we consider a model of the form:

$$\min_{x \in \mathbb{Z}^n, y \in \mathbb{R}^k} \{c^T x + d^T y : A(\zeta)x + Gy \leq p, \quad \forall \zeta \in \mathcal{Z}\}. \quad (\text{RC-MIP})$$

Since uncertainty was showed to appear constraint-wise in a general linear program, we can once again model uncertainty affected constraints by an affine transformation of the uncertainty term $\zeta \in \mathcal{Z}$, namely every element of A can be written as a summation between a linear combination of the components of vector ζ and a constant:

$$A(\zeta)^T = [a_1(\zeta) \ a_2(\zeta) \ \dots \ a_n(\zeta)] = [a_1 \ a_2 \ \dots \ a_n] + [P_1\zeta \ P_2\zeta \ \dots \ P_n\zeta]. \quad (16.29)$$

The robust counterpart then contains constraints of the form:

$$\min_{x \in \mathbb{Z}^n, y \in \mathbb{R}^k} \{c^T x + d^T y : (a_i + P\zeta)^T x + g_i^T y \leq p_i, \quad \forall i \in 1, \dots, l \quad \forall \zeta \in \mathcal{Z}\}, \quad (16.30)$$

where $a_i \in \mathbb{Q}^n$ is the nominal value, $P \in \mathbb{R}^{n \times p}$ is the matrix with vectors $P_1, P_2, \dots, P_n \in \mathbb{R}^p$ as columns, g_i^T is a vector corresponding to the i th row of matrix G and p_i is the i th entry of vector p . Just as in the case of a general linear program, we now wish to bring the RC-MIP problem into a reasonable form, removing the ‘for all’ (\forall) operator. The uncertainty set to be considered is the simple box uncertainty:

$$\mathcal{Z} = \{\|\zeta\|_\infty \leq 1\}. \quad (16.31)$$

This kind of uncertainty set is the most intuitive for the freight routing problem, since the release time of a container is assumed to belong to a certain bounded interval of time. Using the worst-case values of the uncertain parameter ζ , the robust counterpart RC-MIP is re-formulated as:

$$\min_{x \in \mathbb{Z}^n, y \in \mathbb{R}^k} \{c^T x + d^T y : a_i^T x + \|P^T x\|_1 + g_i^T y \leq p_i \quad \forall i \in 1, \dots, l\}. \quad (16.32)$$

We observe that expression (16.31) is a convex optimisation problem that can be re-written as a linear mixed integer problem by introducing auxiliary decision variables. The next step from here is to adjust the continuous variables which in our freight routing model correspond to time-related decisions. We assume that they can

be written as affine functions of the uncertainty as in expression (16.28). By doing so, we obtain the following adjustable robust counterpart:

$$\min_{x \in \mathbb{Z}^n, y_0 \in \mathbb{R}^k, Q \in \mathbb{R}^{k \times p}} \{c^T x + d^T (y_0 + Q\zeta) : A(\zeta)x + G(y_0 + Q\zeta) \leq p, \forall \zeta \in \mathcal{Z}\}. \quad (16.33)$$

In the case of box uncertainty, this can be formulated as a convex problem very similar to (16.31):

$$\min_{x \in \mathbb{Z}^n, y_0 \in \mathbb{R}^k, Q \in \mathbb{R}^{k \times p}} \{c^T x + d^T (y_0 + Q\zeta) : a_i^T x + \|P^T x + Q^T g_i\|_1 + g_i^T y_0 \leq p_i\}. \quad (16.34)$$

The final form of this mixed integer problem without uncertainty removed from the objective and constraints is:

$$\begin{aligned} \min_{x \in \mathbb{Z}^n, y_0 \in \mathbb{R}^k, Q \in \mathbb{R}^{k \times p}, t \in \mathbb{R}} \{ & t : c^T x + d^T y_0 + \|Q^T d\|_1 - t \leq 0, \\ & a_i^T x + \|P^T x + Q^T g_i\|_1 + g_i^T y_0 \leq p_i \quad \forall i \in \{1, \dots, l\}. \end{aligned} \quad (\text{ARC-MIP})$$

Finding a solution to the ARC-MIP reduces to solving a mixed integer program bigger in size than the original MIP. Nevertheless, it provides a suitable modelling framework for the freight routing problem and a way to find static and adjustable robust feasible solutions. Since the robust optimisation approach has been discussed for both a general linear program and the mixed integer case, we are now ready to present a robust model for the freight routing problem.

Robust Model

In the robust model the release times of the commodities are uncertain. We recall that every container has a predefined earliest and latest pickup time from its terminal of origin, and the moment at which it is actually released from the terminal and available for loading on the vehicle is contained in this time window. In mathematical terms we have:

$$r_k \in [e_k, l_k] \quad \forall k \in K,$$

where e_k and l_k mark the earliest and the latest pickup time, respectively. In practice, these two quantities are made available in advance by the terminal where the pickup

should occur. The release is known to take place sometime between these two moments. This can be modelled as follows:

$$r_k = \frac{1}{2}e_k(1 - \zeta_k) + \frac{1}{2}l_k(1 + \zeta_k) \quad \forall k \in K,$$

where $\zeta_k \in [-1, 1]$ is the actual uncertain parameter based on which the release r_k can be computed. Therefore, just as in Sect. “[Robust Optimisation for Mixed Integer Programs](#)”, the uncertainty set is the simple boxed uncertainty given by:

$$\mathcal{Z} = \{\zeta \in \mathbb{R}^k : \zeta_k \in [-1, 1]\}.$$

Based on this uncertainty set, we introduce an adjustable robust model which contains two stages of decisions: the first stage variables that must be determined before the value of the uncertain parameter becomes known, and second-stage decision variables which can change their value according to the realisation of the parameters. In our robust model, the first stage variables $x_{i,j}^{k,m}$, $y_{i,j}^m$, z^m , $\theta_i^{m,l}$, $\tau_i^{k,m}$ and $q_i^{k,m,l}$ concern the routing, the sequence of terminal visits, the assignment and transshipment of containers. The second-stage decisions are the continuous variable D_i^m , A_i^m and L_i^m which account for the explicit departure and arrival times and are modelled as adjustable variables. The idea of adjusting time variables to the uncertain parameters originates from Agra et al. [1], who give a robust formulation for a maritime inventory routing problem with uncertain vessel sailing times. Therefore, we define $D_i^m(\zeta)$, $A_i^m(\zeta)$ and $L_i^m(\zeta)$ as the arrival time, departure time and lateness, respectively, when scenario ζ (a vector containing release times of all commodities) has been revealed.

The first stage solution must ensure that, for each possible realisation of the release times in the uncertainty set, the containers are transported from their origin to their destination without missing any of their planned transshipment on the way. In other words, these decisions should result in a robust plan that can be carried out regardless of delayed releases of containers. In the original deterministic model, all time-related constraints (16.10)–(16.14) become:

$$\theta_i^{m,l} = 1 \Rightarrow D_i^l(\zeta) - A_i^m(\zeta) - s_i \geq 0 \quad \forall m, l \in M, \forall i \in V_m \cap V_l, \forall \zeta \in \mathcal{Z} \quad (16.35)$$

$$y_{i,j}^m = 1 \Rightarrow D_i^m(\zeta) + t_{i,j}^m - A_j^m(\zeta) \leq 0 \quad \forall m \in M, \forall (i, j) \in A, \forall \zeta \in \mathcal{Z} \quad (16.36)$$

$$D_i^m(\zeta) \geq A_i^m(\zeta) + s_i \quad \forall m \in M, \forall i \in V_m, \forall \zeta \in \mathcal{Z} \quad (16.37)$$

$$D_i^m(\zeta) \geq r_k \tau_i^{k,m} \quad \forall k \in K, \forall i \in \{o(k)\}, \forall m \in M, \forall \zeta \in \mathcal{Z} \quad (16.38)$$

$$\tau_i^{k,m} = 1 \Rightarrow L_i^m(\zeta) \geq A_i^m(\zeta) - due_k \quad \forall k \in K, \forall i \in \{d(k)\}, \forall m \in M, \forall \zeta \in \mathcal{Z}. \quad (16.39)$$

As we have already discussed in the previous section, a common approach to handle adjustable variables is to use affine decision rules. In this case, we can write the arrival and departure times as affine functions of the uncertain release times:

$$D_i^m(\zeta) = D_{i,0}^m + \sum_{k \in K} D_{i,k}^m \zeta_k \quad \forall i \in V, \forall m \in M \quad (16.40)$$

$$A_i^m(\zeta) = A_{i,0}^m + \sum_{k \in K} A_{i,k}^m \zeta_k \quad \forall i \in V, \forall m \in M \quad (16.41)$$

$$L_i^m(\zeta) = L_{i,0}^m + \sum_{k \in K} L_{i,k}^m \zeta_k \quad \forall i \in \{d(k) : k \in K\}, \forall m \in M. \quad (16.42)$$

The newly introduced variables $D_{i,0}^m \geq 0$, $D_{i,k}^m \in \mathbb{R}$ and so on must be determined in the first stage, together with the routing, assignment and transshipment decisions. We are interested in robust feasible solutions that satisfy constraints (16.35)–(16.39) for any realisation of the release time vector $\zeta \in \mathcal{Z}$. Such a solution must also satisfy the following re-formulated constraints ($m, l \in M, k \in K, \zeta \in \mathcal{Z}$):

$$\theta_i^{m,l} = 1 \Rightarrow D_{i,0}^l + \sum_{k \in K} D_{i,k}^l \zeta_k \geq s_i + A_{i,0}^m + \sum_{k \in K} A_{i,k}^m \zeta_k \quad \forall m, l, \zeta, \forall i \in V_m \cap V_l \quad (16.43)$$

$$y_{i,j}^m = 1 \Rightarrow D_{i,0}^m + \sum_{k \in K} D_{i,k}^m \zeta_k + t_{i,j}^m \leq A_{j,0}^m + \sum_{k \in K} A_{j,k}^m \zeta_k \quad \forall m, \zeta, \forall (i, j) \in A \quad (16.44)$$

$$D_{i,0}^m + \sum_{k \in K} D_{i,k}^m \zeta_k \geq A_{i,0}^m + \sum_{k \in K} A_{i,k}^m \zeta_k + s_i \quad \forall m, \zeta, \forall i \in V_m \quad (16.45)$$

$$D_{i,0}^m + \sum_{k \in K} D_{i,k}^m \zeta_k \geq \left(\frac{1}{2}e_k(1 - \zeta_k) + \frac{1}{2}l_k(1 + \zeta_k)\right)\tau_i^{k,m} \quad \forall k, m, \zeta, \forall i \in \{o(k)\} \quad (16.46)$$

$$\tau_i^{k,m} = 1 \Rightarrow L_{i,0}^m + \sum_{k \in K} L_{i,k}^m \zeta_k \geq A_{i,0}^m + \sum_{k \in K} A_{i,k}^m \zeta_k - due_k \quad \forall k, m, \zeta, \forall i \in \{d(k)\}. \quad (16.47)$$

As shown earlier, the uncertainty ζ_k can be removed from the constraints by assuming a worst-case realisation of the data. For example, constraints (16.43) can be written as follows:

$$\theta_i^{m,l} = 1 \Rightarrow D_{i,0}^l - s_i - A_{i,0}^m \geq \sum_{k \in K} (A_{i,k}^m - D_{i,k}^l)\zeta_k \quad \forall m, l \in M, \forall i \in V_m \cap V_l, \forall \zeta \in \mathcal{Z}.$$

Since this inequality should hold for any realisation of ζ_k , we impose the following constraint:

$$\theta_i^{m,l} = 1 \Rightarrow D_{i,0}^l - s_i - A_{i,0}^m \geq \sum_{k \in K} |A_{i,k}^m - D_{i,k}^l| \quad \forall m, l \in M, \forall i \in V_m \cap V_l.$$

Moreover, we note that in the constraints above there is no uncertain parameter anymore and all the decision variables are to be determined in the first stage. Moreover, the absolute value can be removed from the expression by introducing an additional decision variable $\alpha_{m,l}^{i,k}$:

$$\begin{aligned} \theta_i^{m,l} = 1 &\Rightarrow D_{i,0}^l - s_i - A_{i,0}^m \geq \sum_{k \in K} \alpha_{m,l}^{i,k} && \forall m, l \in M, \forall i \in V_m \cap V_l \\ -\alpha_{m,l}^{i,k} &\leq A_{i,k}^m - D_{i,k}^l \leq \alpha_{m,l}^{i,k} && \forall k \in K, \forall m, l \in M, \forall i \in V_m \cup V_l. \end{aligned}$$

One can reformulate constraints (16.44)–(16.47) in a similar fashion and obtain the following inequalities:

$$y_{i,j}^m = 1 \Rightarrow D_{i,0}^m + t_{i,j}^m - A_{j,0}^m + \sum_{k \in K} \beta_{k,m}^{i,j} \leq 0 \quad \forall m, \forall (i, j) \in A \quad (16.48)$$

$$-\beta_{k,m}^{i,j} \leq D_{i,k}^m - A_{j,k}^m \leq \beta_{k,m}^{i,j} \quad \forall (i, j) \in A, \forall k, m$$

$$D_{i,0}^m - A_{i,0}^m - s_i \geq \sum_{k \in K} \gamma_{k,m}^i \quad \forall m, \forall i \in V_m \quad (16.49)$$

$$-\gamma_{k,m}^i \leq A_{i,k}^m - D_{i,k}^m \leq \gamma_{k,m}^i \quad \forall i \in V_m, \forall k, m$$

$$D_{i,0}^m \geq \delta_{k_0,m} + \sum_{k \neq k_0} \epsilon_{i,k}^m \quad \forall k_0 \in K, \forall i \in \{o(k)\}, \forall m \quad (16.50)$$

$$-\delta_{k_0,m}^i \leq \frac{1}{2}(l_{k_0} - e_{k_0})\tau_i^{k,m} - D_{i,k_0}^m \leq \delta_{k_0,m}^i \quad \forall k_0 \in K, \forall i \in \{o(k)\}, \forall m$$

$$-\epsilon_{i,k}^m \leq D_{i,k}^m \leq \epsilon_{i,k}^m \quad \forall k, m, \forall i \in \{o(k)\}$$

$$\tau_i^{k,m} = 1 \Rightarrow L_{i,0}^m + due_k - A_{i,0}^m \geq \sum_{k \in K} \eta_{k,m}^i \quad \forall k, m, \forall i \in \{d(k)\} \quad (16.51)$$

$$-\eta_{k,m}^i \leq A_{i,k}^m - L_{i,k}^m \leq \eta_{k,m}^i \quad \forall k, m, \forall i \in \{d(k)\}.$$

We observe that in the constraints above there is no uncertain parameter ζ_k present anymore and all the decision variables are to be determined in the first stage.

The adjustable robust counterpart of the deterministic model presented in Chap. 4 is thus composed from constraints (16.1)–(16.9), (16.15)–(16.22) and (16.48)–(16.51). The objective of the mixed integer program is modelled according to the method described in the beginning of Sect. “The Robust Counterpart”, meaning that the following expression is added to finalise the model:

$$\begin{aligned} \max \quad t \quad \text{where} \quad & \omega_1 \sum_{k \in K} \sum_{m \in M} \sum_{(i,j) \in A} t_{i,j}^m x_{i,j}^{k,m} + \omega_2 \sum_{m \in M} \sum_{i \in V} (L_{i,0}^m + \sum_{k \in K} \mu_{i,k}^m) \leq t \\ & - \mu_{i,k}^m \leq L_{i,k}^m \leq \mu_{i,k}^m, \quad \forall k \in K, \forall i \in \{d(k)\}, \forall m \in M. \end{aligned} \quad (16.52)$$

Solving the robust model will determine all the routing, assignment and transshipment variables. The value obtained for the objective value corresponds to the worst-case realisation of the data. Nevertheless, using the adjustable time variables in the second stage, when the data is revealed, we can improve the value of the objective without re-solving the model. That is due to the fact that in the second stage the lateness term in the objective can still be adjusted and reduced when the realisation of the data is favourable.

Computational Results

In this section, we report on the solutions found for the deterministic and robust formulations of the freight routing problem and compare them to past approaches. We explain how the test instances were generated and show the results for the deterministic model and the robust approach.

Instance Generation

In order to test the models that were given before, we generated multiple problem instances. These were inspired from the work of Kishan Kalicharan [8], who has designed a transport network of eight terminal locations based on Google Maps data. Since some of these locations represented clustered terminals, the original instances were modified to include only nodes which correspond to actual physical locations in real life. For comparison purposes, the number of locations was kept the same. The barge travel times on waterways were assumed to be fixed and their values were approximated using online tools which compute sea distances based on the speed of the vessel. In our transport infrastructure, we assume that some of the locations are terminals, where containers can be transhipped, and some of them are customers, serving as end-locations for the containers. There is also direct connection between every pair of locations in our model. Furthermore, we assume that the service time

is the same at every location. There is a set of commodities (bookings with one or more containers) that need to be transported from the terminals to the customer locations. As in [8], the demand value of each commodity is randomly chosen in the interval $[0, 125]$.

Barges and trucks are available for container transport. The capacity of barges is assumed to be of 100 containers. These barges always start at a particular terminal which in real-life interpretation is a hub-location. We assume that there is an infinite amount of trucks of large capacity available at every location. To ensure that all containers can be transported, the total capacity of all vehicles is always larger than the total demand of all commodities [9]. Finally, the due dates and release times of the containers are chosen in such a way that the difference between them is strictly larger than the time required by barge to travel on the direct connection arc from the origin of the containers to their destination.

To assess the computational difficulty of our models we create instances with 8 nodes, 6 and 12 barges, and 5, 10, 20 and 30 commodities. In total, we generate 10 instances which are tested for three different objective functions by varying the values of the weights ω_1 and ω_2 . We denote a problem instance by km , where k the number of commodities and m the number of available barges. Both the deterministic and the robust model were implemented in AIMMS Developer version 4.53, a mathematical optimisation modelling tool, and solved with CPLEX optimisation solver (Version 12.8, 32-bit). Numerical experiments were carried out on a DELL Latitude E7240 laptop with an Intel(R) Core(TM) i5-4310U CPU 2.00 GHz 2.60 GHz processor and 8 GB RAM memory. This laptop is operational on a 64-bit operating system.

Results of Deterministic Model

In the freight routing problem we are interested in providing an assignment of containers to vehicles whilst minimising the total number of trucking hours and overall lateness. In order to get an idea of how the allowed lateness affects the solution time of the freight routing problem, the deterministic model in Chap. 4 was tested for three different objective functions. These were obtained by varying the weights w_1 and w_2 . Since we have no knowledge of the real costs of trucking activities in practice, we shall gradually increase the weight ω_2 of lateness and keep the first weight $\omega_1 = 1$. The following three objective functions are considered:

- Objective 1: $\sum_{k \in K} \sum_{m \in M} \sum_{(i,j) \in A} t_{i,j}^m x_{i,j}^{k,m} + \sum_{m \in M} \sum_{i \in V} L_i^m$
- Objective 2: $\sum_{k \in K} \sum_{m \in M} \sum_{(i,j) \in A} t_{i,j}^m x_{i,j}^{k,m} + 0.1 \cdot \sum_{m \in M} \sum_{i \in V} L_i^m$
- Objective 3: $\sum_{k \in K} \sum_{m \in M} \sum_{(i,j) \in A} t_{i,j}^m x_{i,j}^{k,m} + 1000 \cdot \sum_{m \in M} \sum_{i \in V} L_i^m$

These values give us a reasonable way to assess which objective yields a solution fast enough. Since these weights are chosen in a way that highly penalises lateness, the results of our model should be comparable to those obtained when completely

Table 16.1 Objective value, solution time (CPU seconds) and gaps between the lower and upper bounds for the freight routing model with lateness allowed. * an upper bound of 3600s was set on the running time of the solver

Data instance	Obj. 1	Gap (%)	Runtime (s)	Obj. 2	Gap (%)	Runtime (s)	Obj. 3	Gap (%)	Runtime (s)
k5m6	100	0	1.94	100	0	1.94	100	0	1.8
k10m6	149.5	0	22.56	149.5	0	22.92	149.5	0	72.53
k20m6	235.5	0	263	235.5	0	245.36	235.5	0	565.52
k30m6	na	na	3600*	na	na	3600*	na	na	3600*
k5m12	100	0	2.06	100	0	3.52	100	0	2.22
k10m12	149.5	0	19.23	149.5	0	24.02	149.5	0	18.9
k20m12	235.5	0	303.47	235.5	0	380.89	235.5	0	178.36
k30m12	na	na	3600*	na	na	3600*	na	na	3600*

Table 16.2 Objective value, solution time (CPU seconds) and gaps between the lower and upper bounds for the freight routing model with no lateness allowed. * an upper bound of 3600s was set on the running time of the solver

Data instance	Objective	Gap (%)	Runtime (s)
k5m6	100	0	4.84
k10m6	149.5	0	114.08
k20m6	na	na	3600*
k30m6	na	na	3600*
k5m12	100	0	30.38
k10m12	149.5	0	21.66
k20m12	235.5	0	319.8
k30m12	na	na	3600*

removing lateness variables. Table 16.1 shows the solution and computation time for each instance and each of the three objective functions that were chosen. The results obtained for the original model with no lateness allowed are given in Table 16.2.

What immediately stands out from the results above is the computational difficulty of the deterministic models for freight routing as for instances with thirty commodities k30m6 and k30m12 the solver could not find a feasible solution within 1 h for any of the models considered. However, we see that the model incorporating lateness performs better than the original version in terms of the computational time required and the solution found. For example, instance k20m6 can be solved to optimality for all the three objective functions considered in the case of allowed lateness but not for the original model. This might be due to the fact that the model including lateness is always feasible, and therefore, it is easier for the solver to find an initial feasible solution than in the situation of hard due dates for the commodities.

In general, we note that the computational time significantly increases for all cases considered when the number of commodities increases. On the other hand, the number of vehicles does not seem to drastically influence the computational time of the instances that we have tested since there are no compelling differences between instances with six or twelve barges. In particular, for the instance with twenty commodities the solver found an optimal solution three times faster when the number of available barges was doubled. This result confirms our expectation,

as a larger fleet of barges offers more routing possibilities and requires less transshipments of containers to trucks.

Regarding the two model formulations, with and without allowing for lateness, we see that the results obtained are the same. This suggests that despite the penalties, the overall lateness obtained if all containers were transported by barge on the main leg of the trip is still much larger than the cost resulting from trucking everything. This is fully due to the choice of values for the parameter ω_2 . One could indeed assign lower numerical values to this weight to obtain solutions with late arrivals of commodities. However, since lateness is used mostly for computational reasons here, we will not look into those situations. In terms of the objective function used to generate the results in Table 16.1, optimising the problem for Objective 3 is the most computationally expensive at least in the case of the first three instances. However, when the number of available barges is increased to twelve, the running time of the solver for Objective 3 is much lower than for Objectives 1 and 2. One possible explanation for this is the fact that the weight $\omega_2 = 1000$ adds a large contribution to the cost solution and thus the solver begins by finding a very expensive feasible solution and then reduces it by re-assigning the commodities over the available barges. If more barges are enabled, then more capacity is available for re-assigning and transporting containers by water instead.

Overall, the results in this section provide an important insight into the computational difficulty required by the deterministic model and help us set an expectation on the numerical effort for the robust model. The largest instances that we could solve, namely k20m6 and k20m12 that have been used are comparable to the transportation activity of a real logistics service provider.

Results of Robust Model

In this section we focus on solving the robust model explained in Sect. “**Robust Model**”, in which the release times of the commodities belong to a predefined uncertainty set. As we already know, the robust mixed integer linear program will determine the routing of vehicles and assignment and transshipment of containers in the first stage, leaving the time variables to be determined in the second stage, when the uncertain release times have been revealed. It is expected that the robust solution is more conservative and thus, of higher cost, than the original deterministic solution. Our goal is to investigate the difference between these solutions and assess whether the ‘price of robustness’ is acceptable given the size of the instance, the level of uncertainty and the practical implications. Moreover, we would like to know what is the influence of the adjustable variables on the solution and objective function when lateness has a high and low penalty.

High Lateness Penalties

For our numerical test we will only consider an instance of manageable size, namely instance k5m12 with Objective 3. This objective is chosen because it has the highest lateness penalty and it has recorded the fastest computational time for the deterministic case, a fact which can be noticed in the last column of Table 16.1. Larger instances have not been considered due to two main reasons. Firstly, the solver would require a very large amount of time to solve them. Secondly, the simple k5m12 instance is already sufficiently diverse to allow us to study different features of the solution. In view of comparison purposes, we assume that all containers have an uncertain release date in an interval of fixed length. We consider six possible interval lengths of 2, 4, 6, 8, 10, and 12 h, which encompass scenarios ranging from small to extremely large delays. For clarification, an uncertainty interval of 2 h, for instance, suggests that the release of a container can occur 1 h before or after its nominal release value.

The solutions obtained by solving the robust model with different sizes for the interval uncertainty are given in Table 16.3. Some remarks are in place concerning the last two columns of this table. When using robust optimisation, it is important to assess what is the ‘price of robustness’, namely what is the additional cost to be paid when immunising the solution with respect to the uncertain parameters. In order to do that, we have also considered the situation when the release times are already available at the beginning of the planning and solved the deterministic problem for two different realisations: the best case, in which every container is released at the earliest opportunity ($\zeta = -1$) and an ‘average’ case ($\zeta = 0$), when the release times occurs at the midpoint of the uncertainty interval. Then we calculated by how much the robust cost increases from the deterministic solution for both cases, and displayed those values in Table 16.3. As we expected, the cost of the robust solution increases as the size of the interval of the release time is enlarged. Moreover, we observe that there is a certain amount of delay that the planning can handle. Namely, for instances with a release delay within 3 h, the solution attained is identical to the one obtained by solving deterministic model with nominal release time values.

Table 16.3 Objective value of the robust model, gap between the current solution and the best lower bound found so far, computation time (CPU seconds), percentage increase from the deterministic objective for $\zeta = -1$, and percentage increase from the deterministic objective for $\zeta = 0$. * an upper bound of 36,000s was set to the execution time of the solver

Data instance	Robust solution	Gap(%)	Runtime(s)	Increase best case	Increase ‘average’ case
k5m12					
2 h interval	100	0	198.63	0 %	0%
4 h interval	100	0	1854.22	0%	0%
6 h interval	100	0	379.25	0%	0%
8 h interval	1415	8.13	36,000*	1315%	8.84%
10 h interval	1465	11.26	36,000*	1365%	12.69%
12 h interval	1595.5	18.52	36,000*	1495.5%	22.73%

Table 16.4 Transportation characteristics of the robust planning

Data instance k5m12	Number of commodities fully trucked	Number of barges used
2 h interval	0	6
4 h interval	0	5
6 h interval	0	5
8 h interval	1	2
10 h interval	4	1
12 h interval	3	2

Table 16.5 Statistics concerning the number of time variables that are adjusted by the affine rules. A random realisation $\zeta \neq 0$ was used for checking whether the time variables have adjusted or not

Data instance k5m12	Number of adjusted time variables	Number of adjusted arrivals	Number of adjusted departures
2 h interval	7	3	4
4 h interval	0	0	0
6 h interval	3	1	2
8 h interval	0	0	0
10 h interval	0	0	0
12 h interval	10	5	5

Regarding the ‘price of robustness’, we see that for a data realisation at the midpoint of the uncertainty set (not a favourable situation), the difference between the robust solution and the best deterministic solution is at most 22.73% (corresponding to a ± 6 h margin for delay). The objective value for the robust solution alone is not sufficiently insightful to assess how the transportation changes when the uncertainty interval increases. To give a measure of this, we include the number of commodities that are transported only by truck and the utilisation of barges in Table 16.4. It is apparent from this table that when commodities are released with significant delay (larger than 3h), it becomes impossible to transport them by barge. However, the uncertainty intervals that we considered were still not sufficiently large to enforce a transportation plan with no barge being utilised.

The numbers of adjusted time variables for every uncertainty interval are highlighted in Table 16.5. From here, we can immediately notice that the affine rules that were proposed indeed induce the adjusting of arrival and departure times. However, it is striking that for uncertainty intervals of 4 h and 8 h, no adjustment occurs. Another important observation from Table 16.5 is that, in the case when adjustment occurs, it does not affect all time variables which were assigned a numerical value in the solution.

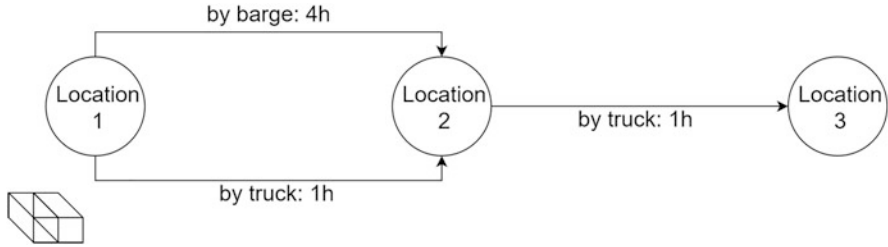


Fig. 16.3 Example of a transportation network with three nodes and two commodities

Low Lateness Penalties

When the lateness penalties are assigned sufficiently small numerical values, the adjustable variables can directly influence the value of the objective function. In order to illustrate this idea we have repeated the test from the previous section using Objective 1 (having a lateness penalty of 0.1). However, due to the high computational effort required by the solver, we were not able to record results in a reasonable amount of time as to include them in this report. Nevertheless, we will demonstrate the benefits of adjusting time variables using a simple example which still incorporates all the transportation elements that were shown by the other generated instances. Consider the transportation network in Fig. 16.3. We assume that there are two containers to be released at Location 1 in the interval [8,10]. One needs to arrive at Location 2 by time unit 14 whilst the other has the same due time but it is destined for Location 3. It is assumed that there are sufficient barges and trucks to carry out these transport requests.

For this particular instance, the worst-case solution given by the robust model in Chap. 5 with Objective 1 has cost 1.2. This corresponds to the situation when these containers are released at time unit 10. Both containers will be taken by barge to Location 2, and from there one will be trucked to its final destination, where it records a lateness of 2 h. However, if the release occurs at time unit 8, then the adjusted cost will be of only 1 cost unit, since there is no lateness recorded at Location 3 anymore.

This result also shows that unlike in the results shown in the previous section, in this case it is possible to adjust the time variables along the routes which include a transshipment. This is due to the fact that we have included in the model constraints of the form

$$A_{i,k}^m \geq \lambda, \quad \forall i \in V, k \in K, m \in M, \tag{16.53}$$

where λ is some small scalar (chosen to be 1 in this example). These ensure in this case that the affine coefficients of the adjustable arrival times will be non-zero. We see that by including these constraints in the model, we can guarantee the adjustment of all time variables.

This example demonstrates that the affinely adjustable robust optimisation framework can be used to obtain improved solutions for the freight routing problem with a high tolerance for lateness. However, the choice for the lateness parameter as well as for the scalar λ is instance-specific. Therefore, at this stage of the research, it is difficult to make assertions about how the robust model can be used for any general instance.

Discussion

In this section, we discuss the most important findings from the numerical experiments and where necessary, provide more insight into the results obtained by closely inspecting the solutions. The deterministic formulation of the freight routing can be successfully solved exactly with the branch-and-bound method for instances as large in size as those including twelve vehicles and twenty commodities. These instances are comparable to what is encountered in practice. However, the relatively large computational time required by the solver is likely due to two main factors, namely the large amount of (binary) decision variables in the model as well the many symmetries of the problem. Given the results in Table 16.1, we see that allowing for lateness with high penalties yields a model which can be solved faster than the original version with hard deadlines.

First of all, the computational results of the simple instance showed that we can obtain robust feasible solutions for the container freight routing problem by solving exactly the robust counterpart. These solutions correspond to transportation plans that can be carried whenever the release time of a container falls within a pre-specified interval. An increase in the size of this uncertainty interval induces higher solution costs, since containers which have a short delivery span will not be transported by barge. This results is fully confirmed by the data in Table 16.2. However, we see that the price to be paid for the robustness of solutions is quite high. As an example we consider the k5m12 case with an 8 h interval. When we assume that the release times of the containers can deviate from their nominal value by 4 h, and they in fact are released on the earliest time possible, the transportation plan obtained is 1315% more expensive than the plan that could have been achieved if all data was known beforehand. If we assume a less favourable realisation, in which half of the commodities are released at their nominal value, and the other half at the latest time possible, the increase is only of 8.84%. Given the lack of information on the real-life situation, therefore difficult to assess if the price of robustness is acceptable when modelling highly uncertain releases for the containers. However, we can state that in a practical instance in which one can infer from historic data that parameters often attain ‘bad realisations’, the robust transport plan can be employed in exchange for a reasonable cost increase.

The price of robustness can also be regarded from a slightly different perspective. We consider a situation in which given some uncertainty intervals, one makes a deterministic plan assuming a certain nominal value for the releases. If the actual

realisation of the parameters is worse than the nominal values, then the deterministic solution is likely to be infeasible. This enforces re-planning of the current vehicles and container assignment. Although there are many ways in which one can re-route, the newly obtained transportation solution might have a higher cost than the robust solution that could have prevailed over the delays.

We have applied the affinely adjustable robust optimisation framework in order to allow the arrival and departure times of the vehicle to change according to the realisation of the container release and induce some degree of flexibility in the planning. Whilst the objective value of a solution remained unchanged, due to very high lateness penalties, some vehicles might be able to arrive or depart earlier at certain locations. Concerning the actual adjustment of variables, it was at first surprising to notice the low proportion of time variables are affected by the changes in data realisation. However, at a closer inspection of the container routes given by the solutions we were able to find a possible explanation for this. We have found that adjustment is only effective for arrival and departure times on a particular kind of route. In other words, for direct routes, on which a commodity is shipped from its origin location to their destination by means of a single vehicle, adjustment takes place. Otherwise, if a switch of vehicle, transshipment or additional commodity pickup occurs on the way, then only arrivals and departures that can be adjusted are those at the beginning location of the route. When inspecting the solution, we have found that the instances k5m12 with 4 h, 8 h and 10 h intervals, in which no adjustment took place, indeed included no direct routes. Moreover, adjustment seemed to be particularly successful for the case with the highest uncertainty interval. Essentially, since all commodities are directly trucked from origin and destination in this case, all the arrivals and departures are adjusted.

It is difficult to further explain why adjustment only affects direct routes, but it might be related to the fact that the vehicle synchronisation and transshipment constraints in the robust model force the affine coefficients of the adjustable variables to take the value zero. This suggests that on non-direct routes it is difficult to ensure adjustment with respect to all the decisions made on that route: transshipment, vehicle switch, loading or unloading of commodities. Another factor which may influence adjustment is the symmetry of solutions and the fact that the same objective can be achieved by many different routes. For instance, there are solutions in which a barge is assigned for every commodity, resulting in significantly less transshipments, which have the same objective as a solution with a smaller barge utilisation. Nevertheless, we were able to produce an example in which the robust model with an additional sets of constraints gives a fully adjusted solution, which is cheaper than the worst-case scenario, if the data assumes a favourable realisation. Therefore, we have shown that the adjustment of variables can result in a direct improvement of the objective function. For a generalisation of this result, a sensitivity analysis of the instance parameters on the lateness term in the objective is required.

References

1. Agra, A., Christiansen, M., Hvattum, L. M., & Rodrigues, F. (2018). Robust optimization for a maritime inventory routing problem. *Transportation Science*, 52(3), 509–525. <https://doi.org/10.1287/trsc.2017.0814>.
2. Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. (2009). *Robust Optimization (Princeton Series in Applied Mathematics)*. Princeton University Press. <https://www.amazon.com/Robust-Optimization-Princeton-Applied-Mathematics/dp/0691143684?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0691143684>.
3. Ben-Tal, A., Goryashko, A., Guslitzer, E., & Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2), 351–376. <https://doi.org/10.1007/s10107-003-0454-y>.
4. Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research*, 52(1), 35–53. <https://doi.org/10.1287/opre.1030.0065>.
5. Birge, J. R., & Louveaux, F. (2011). *Introduction to Stochastic Programming (Springer Series in Operations Research and Financial Engineering)*. Springer
6. Chouman, M., Crainic, T. G., & Gendron, B. (2017). Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, 51(2), 650–667.
7. Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer programming, volume 271 of graduate texts in mathematics*.
8. Kalicharan, K. (2018). *Intermodal transport: Routing vehicles and scheduling containers*. Master's thesis, TU Delft.
9. Sharypova, K. (2014). Optimization of hinterland intermodal container transportation. Ph.D. thesis, Eindhoven University of Technology.
10. Yanıkoğlu, İ., Gorissen, B. L., den Hertog, D. (2019). A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3), 799–813.

Index

A

Adjustable robust optimisation, 277
Agent-centric network, 171
Agility, 120
Alternative performance indicator, 119
Arc residual capacity cut, 159
Authority, 22

C

Complexity, 4, 10
Connected components, 203
Cooperative, 8
Customer satisfaction, 129
Cutset cut, 163
Cutting planes, 157

D

Decision space, 226
Delay, 239
Demand elements, 25
Demifuture, 64

E

Equilibrium, 187
Event-driven simulation, 175
Expected future iteration, 78

F

Fair, 201
Fair user equilibrium, 203
Flexibility, 120, 125

Framework, 17

Full information model, 178

G

General cuts, 158
Global, 23
Graph reduction, 43, 48, 90

I

Identifier, 22

L

Lexicographic method, 135
Limited, 7
Local, 23
Logistic Service Provider (LSP), 8

M

Markov decision processes, 21, 76
MCMCF problem, 42, 119, 120, 133, 134, 143
Measure of robustness, 122
Multi-commodity network design, 144
Multi objective approach, 135
Multi-objective optimisation, 133, 258
Multistage stochastic programming, 67, 73, 74

O

Objective, 23
Omnifuture, 63
Order tolls, 204

P

Pareto optimal solution, 135
Partially pessimistic future iteration, 81
Path tolls, 193, 210
Public information models, 176

R

Resilience, 120
Resource elements, 24
Robustness, 120
Robust optimisation problem, 274

S

Selfish, 8
Self-organisation, 4, 10
Service network design, 20
Shared information, 171
Simulation, 175, 219
Single future iteration heuristics, 78
Social, 8
Solution method mapping, 19, 31, 35
Space-time network, 42
Stochastic optimisation problem, 59, 251, 274

Strong cut, 165

Symmetry breaking cut, 158
Synchronodal, 7
System optimum, 188

T

Time windows, 245
Tolls, 193, 204, 210
Transit idea, 61
Transit instance, 61
Transshipments, 253
2-stage delivery chains, 235
Two-stage stochastic programming, 66

U

Uncertainty, 12, 219, 235, 251
User equilibrium, 187, 190, 201

V

Variable reduction, 143, 147
Vehicle routing problem (VRP), 19, 32, 33,
235, 253, 273