Dario Pacino
Stefan Voß
Rune Møller Jensen (Eds.)

LNCS 8197

# Computational Logistics

4th International Conference, ICCL 2013
Copenhagen, Denmark, September 2013
Proceedings

Springer

Lecture Notes in Computer Science 8197

*Commenced Publication in 1973*
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Dario Pacino  Stefan Voß
Rune Møller Jensen (Eds.)

# Computational Logistics

4th International Conference, ICCL 2013
Copenhagen, Denmark, September 25-27, 2013
Proceedings

Springer

Volume Editors

Dario Pacino
Technical University of Denmark
DTU TRANSPORT, Dept. of Transport
Bygningstorvet 116B, 2800 Kgs. Lyngby, Denmark0
E-mail: darpa@transport.dtu.dk

Stefan Voß
University of Hamburg
Institute of Information Systems (IWI)
Von-Melle-Park 5, 20146 Hamburg, Germany
E-mail: stefan.voss@uni-hamburg.de

Rune Møller Jensen
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
E-mail: rmj@itu.dk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-41018-5 e-ISBN 978-3-642-41019-2
DOI 10.1007/978-3-642-41019-2
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013947933

CR Subject Classification (1998): F.2.2, H.4.1-2, D.2.2, I.2.8, I.6, J.1, J.7, C.2, F.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues



*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

Computational logistics refers to the planning and implementation of logistics tasks using computations and advanced decision support. It covers significant work regarding theory and application of systems and methodologies for decision analytics, advancing planning, and operations in logistics. It is applied in various areas including the flow and storage of goods or services as well as related information from their source to their destination. Typically, optimization models and algorithms are developed, verified, and applied for planning and executing complex logistics tasks, e.g., for finding the most efficient scheduling/plan for the transport of passengers or goods. These models and algorithms are integrated with advanced information and communication technology (IT) to obtain satisfactory results in appropriate time even for large scale problem instances and providing interactivity, visualization etc. for a better understanding, problem solution, and decision support. Furthermore, computational logistics involves the use of information systems and modern IT tools for the design, planning, and control of logistics networks as well as the complex tasks within them.

The *International Conference on Computational Logistics* (ICCL) provides an opportunity for researchers and practitioners in the field of computational logistics to present their latest results and findings in a fruitful and open-minded environment. This volume of the *Lecture Notes in Computer Science* series consists of selected papers presented at the 4th International Conference on Computational Logistics, held at the Technical University of Denmark (DTU), Kgs. Lyngby, Denmark, during September 25th to 27th, 2013.

The ICCL 2013 was the fourth of its kind, the first was held in 2010 in Shanghai (see a split issue of selected papers in *Netnomics*, Volume 12, Issue 3 and Volume 13, Issue 2), the second one in Hamburg, Germany (see Volume 6971 of the *Lecture Notes in Computer Science*), and the third one again in Shanghai, (see Volume 7555 of the *Lecture Notes in Computer Science*). The idea of inviting participants to Copenhagen came by the city's ambition of becoming the first $CO_2$-neutral city by 2025. Computational logistics plays a very important role in order to achieve this goal, and both public and private funds have opened the road for innovative research. Among others, a special focus is paid on analyzing logistic systems using alternative energy resources. A large effort is also directed towards the maritime sector where a great effort is made to reduce the $CO_2$ footprint of the shipping industry. The participation in the ICCL 2013 of some of the key Danish researchers and practitioners, and the academic submissions included in this special issue, brought the participants great flavor of the state-of-the-art of computational logistics and its scientific outputs and implementations as well as applications.

The contributions presented at the conference as well as the papers in these proceedings show that computational logistics is gaining more and more

importance in various areas. Academics as well as practitioners are well involved in the development of the field. Computational logistics is going from strength to strength. This is well reflected in the advances seen in the contributions presented at the conference as well as the selected papers in these proceedings. Following the focus of those papers accepted we grouped the contributions into five parts as follows:

- Part I: Maritime Shipping
- Part II: Road Transport
- Part III: Vehicle Routing Problems
- Part IV: Aviation Applications
- Part V: Logistics and Supply Chain Management

While we believe that these proceedings provide insights into the state-of-the-art of the field, we also hope and know that the story is never-ending. That is, new advances on different levels are expected, taking into consideration innovations in all areas in computational logistics, building upon what we have developed.

Organizing a conference and publishing the proceedings is an effort relying on the help and support of many people in various activities. Many thanks go to all the authors and presenters for their contributions. In addition, we greatly appreciate the valuable help and cooperation from the members of the international Program Committee and the referees. While preparing the conference and compiling the proceedings we also received enthusiastic support from Julia Bachale (IWI Hamburg) as well as the team from the local organizers in Copenhagen and Kgs. Lyngby.

September 2013

Dario Pacino
Stefan Voß
Rune Møller Jensen

# Organization

## Organization Chair

| | |
|---|---|
| Dario Pacino | Technical University of Denmark, Denmark |
| Rune Møller Jensen | IT-University of Copenhagen, Denmark |
| Stefan Voß | University of Hamburg, Germany |

## Organization Committee

| | |
|---|---|
| Dario Pacino | Technical University of Denmark, Denmark |
| Rune Møller Jensen | IT-University of Copenhagen, Denmark |
| Line Blander Reinhardt | Technical University of Denmark, Denmark |

## Program Committee and Referees

| | |
|---|---|
| Panagiotis Angeloudis | Imperial College London, UK |
| Torben C. Barth | Technical University of Denmark, Denmark |
| Jürgen W. Böse | Hamburg University of Technology, Germany |
| Buyang Cao | Tongji University Shanghai, China |
| Marco Caserta | IE Business School Madrid, Spain |
| José Ceroni | Pontificia Universidad Catolica de Valparaíso, Chile |
| Marielle Christiansen | Norwegian University of Science and Technology, Norway |
| Theo Crainic | Université du Québec à Montréal, Canada |
| Joachim R. Daduna | Berlin School of Economics and Law, Germany |
| Rommert Dekker | Erasmus University, The Netherlands |
| Wolfgang Domschke | Technical University of Darmstadt, Germany |
| Kjetil Fagerholt | Norwegian University of Science and Technology, Norway |
| Yingjie Fan | University of Hamburg, Germany |
| Martin Josef Geiger | Helmut Schmidt University, Germany |
| Monica Gentili | Università di Salerno, Italy |
| Ricardo Giesen | Pontificia Universidad Católica de Chile, Chile |
| Hans-Dietrich Haasis | ISL Bremen, Germany |
| Richard F. Hartl | University of Vienna, Austria |
| Geir Hasle | SINTEF Applied Mathematics Department of Optimisation, Norway |
| Sin C. Ho | Aarhus University, Denmark |
| Hao Hu | Shanghai Jiao Tong University, China |
| Herbert Kopfer | University of Bremen, Germany |

| | |
|---|---|
| Gilbert Laporte | HEC Montréal, Canada |
| Rune Larsen | Technical University of Denmark, Denmark |
| Janny Leung | Chinese University of Hong Kong, China |
| Arne Løkketangen | Molde College, Norway (*in memoriam*) |
| Judith Mulder | Erasmus University, The Netherlands |
| Rudy Negenborn | Delft University of Technology, The Netherlands |
| Ana Paias | Cidade Universitária, Portugal |
| Giovanni Pantuso | Norwegian University of Science and Technology, Norway |
| David Pisinger | Technical University of Denmark, Denmark |
| Helena Ramalhinho Lourenco | Universitat Pompeu Fabra, Spain |
| Line Blander Reinhardt | Technical University of Denmark, Denmark |
| Stefan Røpke | Technical University of Denmark, Denmark |
| Juan José Salazar González | University of La Laguna, Spain |
| Hans-Jürgen Sebastian | RWTH Aachen, Germany |
| Xiaoning Shi | Shanghai Jiao Tong University, China, and University of Hamburg, Germany |
| Patrick Siarry | Paris-Est University, France |
| Maria Grazia Speranza | Brescia University, Italy |
| Robert Stahlbock | University of Hamburg, Germany |
| Kevin Tierney | IT-University of Copenhagen, Denmark |
| Theodore Tsekeris | Centre for Planning and Economic Research (KEPE), Greece |
| Min Wen | Technical University of Denmark, Denmark |
| David Woodruff | University of California at Davis, USA |
| Farouk Yalaoui | University of Technology of Troyes, France |
| Tsz Leung Yip | Hong Kong Polytechnic University, China |

# Table of Contents

## Maritime Shipping



## Road Transport



## Vehicle Routing Problems

## Aviation Applications



## Logistics and Supply Chain Management

# Hierarchical Control of Equipment in Automated Container Terminals

Jianbin Xin, Rudy R. Negenborn, and Gabriel Lodewijks

Delft University of Technology, Marine and Transport Technology
Transport Engineering and Logistics,
Delft, The Netherlands
{j.xin,r.r.negenborn,g.lodewijks}@tudelft.nl

**Abstract.** The performance of container terminals needs to be improved to adapt the growth of handled containers and maintain port sustainability. This paper provides a methodology to improve the throughput of the container terminal in an energy-efficient way. The behaviors of equipment are considered as consisting of a higher level and a lower level representing discrete-event dynamics and continuous-time dynamics, respectively. These dynamics need to be controlled. For controlling the higher level dynamics, a minimal makespan problem is solved. For this, the minimal time required for carrying out a task at the lower level is needed. The minimal time for carrying out a task at the lower level is obtained by Pontryagin's Minimum Principle. The actual operation time, allowed by the higher level for completing a task by one piece of equipment at the lower level, is determined by the scheduling algorithm at the higher level. The lower level dynamics are controlled using optimal control to achieve the minimal energy consumption when the operation time allowed is given. A simulation illustrates how energy-efficient management of equipment for the minimal makespan could be obtained by the proposed methodology.

**Keywords:** Optimal control, hierarchical control, container terminals, energy consumption.

## 1 Introduction

Over the last decades, there has been a significant growth of global freight transport due to the enormous commercial trade. Over 60% of worldwide deep-sea cargo is transported by containers [12]. The management of freight transport needs to accommodate this increasing demand of containers. Intermodal transport [5] is hereby considered frequently since it provides flexibility and scalability as different transport modalities can cover different areas with respect to transport distance. As an intermodal transport hub, a container terminal represents the interface among the modalities of vessel, barge, train and truck. Therefore, container terminals play a crucial role in freight transport.

The increasing amount of containers that arrive and depart with container ships provides much pressure for terminal operators. The throughput, i.e., the

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 1–17, 2013.

number of containers handled per hour, should be increased. Meanwhile, energy consumption needs to be reduced to adapt sustainability. In contrast to building new terminal infrastructures, terminal management can be improved in order to maximize the performance of the existing infrastructure, possibly in an economical way.

Despite of the accumulation of literature on container terminal control (see, e.g., [12]), there is a remarkable absence of research that provides energy-efficient management of equipment at the operational level, considering the dynamics and constraints of equipment. Object-oriented approaches [2, 9], agent-oriented programming [10, 15] and Mathematics-based approaches [1, 4] emphasize the throughput without consideration for energy consumption, which is driven by continuous-time dynamics. Little attention has been paid to sustainability of container terminals. Emissions and energy consumption in container terminals are emphasized in [13, 14]. Nevertheless, there, this issue was addressed at the strategic level, instead of the operational level. Consequently, it is still not clear how energy consumption can be reduced when it comes to operational container terminal management.

The aim of this paper is to investigate how to improve the performance at the operational level when combining throughput and energy consumption objectives. A container terminal is modeled as the combination of discrete-event dynamics and continuous-time dynamics. Previous work [16] considered a hybrid model predictive control (MPC) approach. The MPC approach proposed integrates throughput and energy consumption at the operational level, taking into account dynamics and constraints of the system. However, the controller involves solving a mixed-integer quadratic programming problem, which can be very time-consuming. Therefore, in this paper a hierarchical controller is proposed, based on a decomposition of the control problem into a control problem for higher level discrete-event dynamics and one for lower level continuous-time dynamics. This approach aims at achieving energy efficient management of equipment, while minimizing the makespan at the operational level. The actions for each piece of equipment are determined to achieve the desired performance.

This paper is organized as follows. Section 2 describes the modeling of three different pieces of equipment. Section 3 proposes a hierarchical control architecture of the equipment. Section 4 illustrates the behavior of the proposed approach in a simulation study. Section 5 concludes this paper and provides discussions for future research.

## 2 Modeling of Equipment

In general, in a container terminal there are multiple types of equipment used to handle containers. A container terminal is referred to as an automated container terminal when the equipment can be controlled fully automatically without any human intervention. There are multiple quay cranes (QCs), multiple automated guided vehicles (AGVs) and multiple automated stacking cranes (ASCs) for transporting containers from a vessel to the stacking area and vice versa. In this paper,

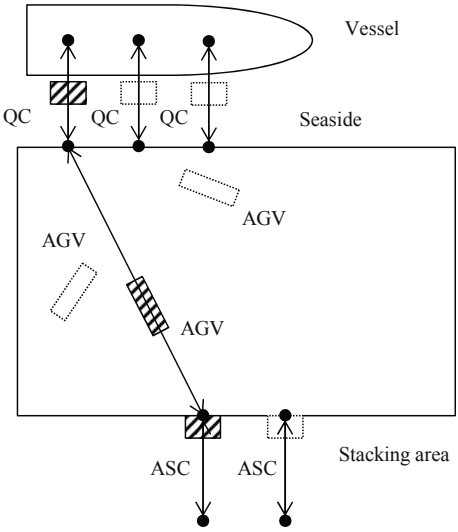



**Fig. 1.** Schematic layout of equipment in an automated container terminal. The dashed pieces of equipment are considered in this paper.

as a starting point for illustrating the hierarchical control structure proposed only one QC, one AGV, and one ASC are considered for transporting containers. In our system, the layout of the equipment is as shown in Fig. 1. A QC picks up a container from the vessel and then unloads it to an AGV. The AGV moves with a container from the quayside and unloads it to an ASC in the stacking area. The ASC then transports the container to the position in the storage area. Accelerations and decelerations of the pieces of equipment have to be determined in an optimal way, as well as the moment of which containers are transported from one piece of equipment to the next.

The dynamics of the pieces of equipment considered are driven by discrete events when a container is transferred from one piece of equipment to another one. Meanwhile, the continuous dynamics, i.e., the position and speed of one piece of equipment, evolve between these discrete changes. The dynamics of transporting containers can therefore be represented by the combination of discrete-event dynamics and continuous-time dynamics. The vessel and the stacking area are considered as components that have no internal dynamics.

### 2.1 Hierarchical Decomposition

In general, a large class of systems can be described by the architecture of Fig. 2 [8]. At the lower layer, the system model is usually described by means of differential-algebraic equations. At the higher level the system description is more abstract. Typically the controller designed for the top level is a discrete event supervisory controller (see, e.g., [11]). The higher level and the lower level

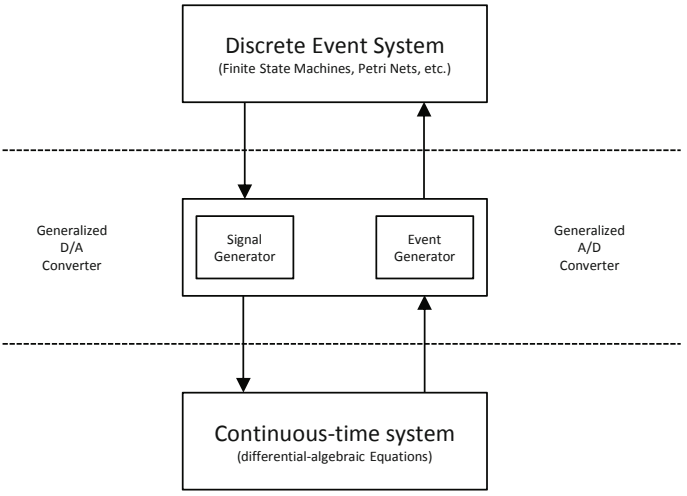



**Fig. 2.** System decomposed into a discrete-event and a continuous-time part [8]

communicate by means of an interface that translates between signals and symbols. For most of these systems the control design approach has been "divide and conquer", i.e., the higher-level and lower-level controllers are designed independently and then linked by an interface that is designed for the specific problem. The dynamics of transporting containers can be decomposed into two levels. At the higher level the discrete-event dynamics occur while at the lower level the continuous-time dynamics evolve. The discrete-event can be the exchanging of a container by two types of equipment while the continuous-time dynamics can be the transport of a container by one piece of equipment. As will be below detailed, these two levels can be linked by the operation time allowed for each piece of equipment for doing a certain task, since the higher level and the lower level both are involved. Correspondingly, the modeling of transporting containers can be decomposed into a discrete-event scheduling of the interactions of equipment and a continuous-time model used for control of each piece of equipment. In the following part, the discrete-event model for interaction of equipment and the continuous-time model for each piece of equipment will be discussed.

### 2.2 Higher-level Discrete-event Dynamics

The transport cycle of one container can be simplified into a sequence of six tasks involving discrete-event dynamics. The sequence of transporting containers by one QC, one AGV and one ASC as a sequence of six types of tasks is shown in Fig. 3. We consider the case in which container $i$ is transported from the seaside area to the stacking area. Let $\tau_{i,j}$ denote a type of task involving container $i$. We consider six types of tasks:

- $\tau_{i,1}$: move from position $P_2$ to $P_1$ to pick up container $i$;
- $\tau_{i,2}$: transport container $i$ from position $P_1$ to $P_2$;
- $\tau_{i,3}$: transport container $i$ from position $P_2$ to $P_3$;
- $\tau_{i,4}$: transport container $i$ from position $P_3$ to $P_4$;
- $\tau_{i,5}$: move from position $P_4$ to $P_3$ in preparation for picking up container $i+1$;
- $\tau_{i,6}$: move from position $P_3$ to $P_2$ in preparation for picking up container $i+1$;

Each task is carried out using a specific piece of equipment. Task $\tau_{i,1}$ and $\tau_{i,2}$ are carried out by the QC. Task $\tau_{i,3}$ and $\tau_{i,6}$ are carried out by the AGV. Task $\tau_{i,4}$ and $\tau_{i,5}$ are carried out by the ASC. An event refers to either the beginning or the completion of a task. For the six types of tasks defined above, whether or not task $\tau_{i,1}$ and task $\tau_{i,3}$ can begin depends on both the completion of task $\tau_{i,2}$ and task $\tau_{i,6}$; the beginning of task $\tau_{i,4}$ and task $\tau_{i,6}$ depends on both the completion of task $\tau_{i,3}$ and task $\tau_{i,5}$. Let $x_{i,j}$ denote the time at which task $\tau_{i,j}$ ends. The time at which both two various types of equipment are available for the shift of one container is called the meeting time. The time at which the different tasks end can be computed as follows:

$$x_{i,1} = \max(x_{i-1,2}, x_{i-1,6}) + s_{i,1} \tag{1}$$
$$x_{i,2} = x_{i,1} + s_{i,2} \tag{2}$$
$$x_{i,3} = \max(x_{i-1,6}, x_{i,2}) + s_{i,3} \tag{3}$$
$$x_{i,4} = \max(x_{i-1,5}, x_{i,3}) + s_{i,4} \tag{4}$$
$$x_{i,5} = x_{i,4} + s_{i,5} \tag{5}$$
$$x_{i,6} = \max(x_{i-1,5}, x_{i,3}) + s_{i,6}, \tag{6}$$

where $s_{i,j}$ describes the time required for doing task $\tau_{i,j}$.

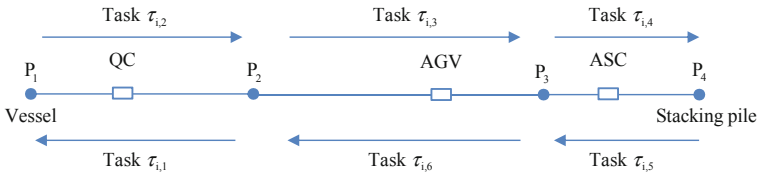



**Fig. 3.** The sequence of transporting containers by three pieces of equipment

### 2.3 Lower-level Continuous-time Dynamics

At the lower level the continuous-time dynamics of individual pieces of equipment are considered. In this paper, we assume that the dynamics of the pieces of equipment are identical. Let the continuous-time dynamics of one piece of equipment be described as follows:

$$\dot{\mathbf{z}}(t) = g(\mathbf{z}(t), u(t)), \tag{7}$$

where $\mathbf{z}(t) = [z_1(t), z_2(t)]^{\mathrm{T}}$ is the continuous state and $u(t)$ is the control variable of the equipment. More specifically, let the dynamics of one piece of equipment be given by:

$$\dot{z}_1(t) = z_2(t), \quad z_1(t) \in [0, S] \tag{8}$$

$$\dot{z}_2(t) = u(t), \quad z_2(t) \in [-v_{\max}, v_{\max}], \quad u(t) \in [-u_{\max}, u_{\max}], \tag{9}$$

where $z_1(t)$ (m) and $z_2(t)$ (m/s) describe the position and the velocity of equipment, respectively, $u(t)$ (m/s$^2$) represents the acceleration, $S$ is the traveling distance of equipment, $[-v_{\max}, v_{\max}]$ is the constraints on $z_2(t)$, and $u_{\max}$ is the maximum on the acceleration.

## 3 Hierarchical Controller

Based on the decomposed dynamics, the container terminal controller can be described in terms of two levels in a hierarchical structure (see Fig. 4):

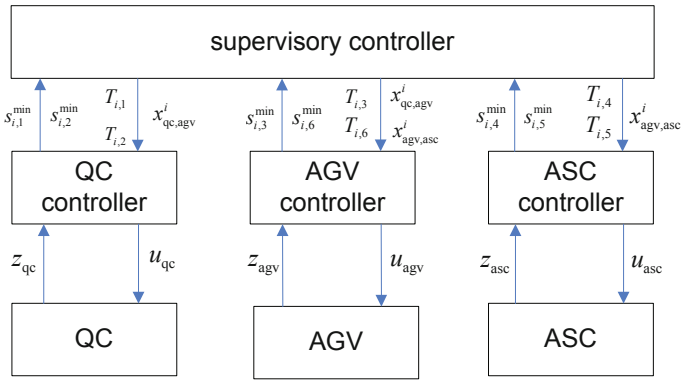



**Fig. 4.** The hierarchical controller

– The higher level
In the higher level, the supervisory controller schedules the tasks of each piece of equipment, or more specifically: the supervisory controller determines at what time which task should start and how much time is allowed for carrying out that task. In order to schedule the tasks, the higher level needs to know how much time carrying out each task to be scheduled requires. The higher level therefore requests from the controllers controlling the dynamics of the equipment at the lower level the minimal time required for doing a particular task. Based on this required time received from the lower level and the discrete-event dynamics, the supervisory controller schedules tasks.

The resulting schedule minimizes the makespan. The schedule provides the time at which each task must be finished. The time allowed for carrying out a task, as well as the meeting time is then sent by the supervisory controller to the relevant controller in the lower level.

– The lower level
In the lower level, the system is driven by the continuous-time dynamics of each piece of equipment. A controller of a piece of equipment can do particular tasks. When the supervisory controller asks a lower-level controller how much time is required to do a task, it uses minimal-time control to determine. After receiving the scheduled time for meeting and completing the task from the supervisory controller, the controller of each piece of equipment will try to finish its task within this time interval. Based on the continuous-time dynamics and a certain cost function defined over a time horizon, optimal control is applied for the continuous-time control of the equipment.

The details of the controller in the higher-level and lower-level are given next.

### 3.1 The Higher-level Controller

In the higher level, the goal is to minimize the makespan. The minimal makespan is determined by the operation time $s_{i,j}$ of task $\tau_{i,j}$ described in (1)-(6). In the higher level, the scheduling problem can be described as follows:

$$\min_{\mathbf{S}} \sum_{i=1}^{N} J_i(x_{i,j}, s_{i,j}) \tag{10}$$

subject to the discrete-event dynamics (1)-(6) and the time constraints due to the time required by the lower level to carry out tasks, where $N$ is the number of containers, $J_i$ is defined as the transport time associated with container $i$ and $\mathbf{S} = [s_{1,1}, ..., s_{1,6}, s_{2,1}, ..., s_{2,6}, s_{N,1}, ..., s_{N,6}]^{\mathrm{T}}$ describes the vector with variables representing the time required for doing task $\tau_{i,j}$ associated with container $i$.

For the sake of simplicity, we assume here that there is no reshuffling of containers in the vessel. Considering $N$ containers that are transported successively, $\sum_{i=1}^{N} J_i = \sum_{i=1}^{N-1} J_i + J_N$. Solving $\min \sum_{i=1}^{N} J_i$ is based on the result of solving $\min \sum_{i=1}^{N-1} J_i$. Therefore, the optimal control problem of $N$ containers can be solved recursively from container 1 to container $N$. The completion time of the handling of container $N$ is given by $x_{N,4}$, which gives the time at which container $N$ is unloaded in the storage yard. The scheduling problem of transporting $N$ containers in the higher level can therefore be written as follows:

$$\min_{\mathbf{S}} x_{N,4} \tag{11}$$

subject to: for $i = 1, 2, ..., N$,

$$x_{i,1} = \max(x_{i-1,2}, x_{i-1,6}) + s_{i,1} \tag{12}$$
$$x_{i,2} = x_{i,1} + s_{i,2} \tag{13}$$
$$x_{i,3} = \max(x_{i-1,6}, x_{i,2}) + s_{i,3} \tag{14}$$
$$x_{i,4} = \max(x_{i-1,5}, x_{i,3}) + s_{i,4} \tag{15}$$
$$x_{i,5} = x_{i,4} + s_{i,5} \tag{16}$$
$$x_{i,6} = \max(x_{i-1,5}, x_{i,3}) + s_{i,6} \tag{17}$$
$$s_{i,j}^{\min} \leq s_{i,j}, \quad j \in \{1, 2, 3, 4, 5, 6\}, \tag{18}$$

where $s_{i,j}^{\min}$ is a lower bound of time required $s_{i,j}$. This lower bound $s_{i,j}$ for task $\tau_{i,j}$ is requested by the supervisory controller from the lower-level controller.

Solving optimization problem (11)-(18) gives the completion time $x_{i,j}$ associated with task $\tau_{i,j}$. Both the start of task $\tau_{i,1}$ and task $\tau_{i,3}$ requires the synchronization of task $\tau_{i,2}$ (carried out by the QC) and task $\tau_{i,6}$ (carried out by the AGV), which means that task $\tau_{i,2}$ and task $\tau_{i,6}$ are finished together. We define the variable $x_{\text{qc,agv}}^{i}$ as the meeting time of the QC and the AGV. Similarly we define $x_{\text{agv,asc}}^{i}$ as the meeting time of the AGV and the ASC, at which both the AGV and ASC start their following task. The meeting time $x_{\text{qc,agv}}^{i}$ and $x_{\text{agv,asc}}^{i}$ are determined as follows:

$$x_{\text{qc,agv}}^{i} = \max(x_{i-1,6}, x_{i,2}) \tag{19}$$
$$x_{\text{agv,asc}}^{i} = \max(x_{i-1,5}, x_{i,3}). \tag{20}$$

The operation time $T_{i,j}$ allowed by the supervisory controller for carrying out task $\tau_{i,j}$ based on the meeting time above. According to Figure 3, $T_{i,1}, T_{i,2}$ are linked to the QC, $T_{i,3}, T_{i,6}$ are linked to the AGV and $T_{i,4}, T_{i,5}$ are linked to the ASC. Considering there is no synchronization of different types of equipment (here we only consider QC, AGV and ASC) between task $\tau_{i,1}$ and task $\tau_{i,2}$ and between task $\tau_{i,4}$ and task $\tau_{i,5}$, for simplicity we assume $T_{i,1} = T_{i,2}$ for the QC and $T_{i,4} = T_{i,5}$ for the ASC. Therefore, the operation time allowed by the higher level for carrying out the tasks can be computed as follows: for $i = 1, 2, .., N$

$$T_{i,1} = \frac{x_{\text{qc,agv}}^{i} - x_{\text{qc,agv}}^{i-1}}{2} \tag{21}$$
$$T_{i,2} = \frac{x_{\text{qc,agv}}^{i} - x_{\text{qc,agv}}^{i-1}}{2} \tag{22}$$
$$T_{i,3} = x_{\text{agv,asc}}^{i} - x_{\text{qc,agv}}^{i-1} \tag{23}$$
$$T_{i,4} = \frac{x_{\text{agv,asc}}^{i} - x_{\text{agv,asc}}^{i-1}}{2} \tag{24}$$
$$T_{i,5} = \frac{x_{\text{agv,asc}}^{i} - x_{\text{agv,asc}}^{i-1}}{2} \tag{25}$$
$$T_{i,6} = x_{\text{qc,agv}}^{i+1} - x_{\text{agv,asc}}^{i}. \tag{26}$$

The operation time $T_{i,j}$ and the meeting time $x^i_{\text{qc,agv}}$ and $x^i_{\text{agv,asc}}$ will be sent by the supervisory controller to the controller of each individual piece of equipment at the lower level. The relation above is shown in Fig. 5. Each individual piece of equipment should then make sure to finish its transport task $\tau_{i,j}$ before the determined operation time $T_{i,j}$. The controller of each individual piece of equipment can hereby include additional control objectives, e.g., aimed at energy saving.

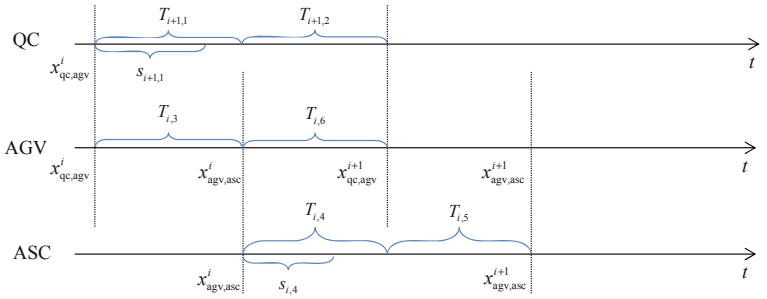



**Fig. 5.** The relation of time variables

### 3.2 The Lower-level Controllers

At the lower level, each piece of equipment has a controller that decides on the continuous-time trajectories of the piece of equipment. In each controller at the lower level, an optimal control problem is formulated so as to finish the task given by the higher level within the operation time allowed and the meeting time. The controller in the lower level can hereby take into account additional objectives, such as energy consumption minimization. The specific control problem of a piece of equipment depends on the task that it has to carry out. In particular, in this case depending on the task and the position from where to start to where to go, the moving behaviors of each equipment are different. Let $z_a$ and $z_b$ denote the origin and destination positions of a piece of equipment. For task $\tau_{i,j}$, the controller of a piece of equipment solves the following problem:

$$\min_{u_j(t)} J(\mathbf{z}_j(t), u_j(t)) \tag{27}$$

subject to

$$\dot{\mathbf{z}}(t) = g(\mathbf{z}(t), u(t)), \quad \mathbf{z}(0) = \mathbf{z}_a, \quad \mathbf{z}(T_{i,j}) = \mathbf{z}_b, \quad t \in [0, T_{i,j}), \tag{28}$$

where $J(\mathbf{z}_j(t), u_j(t)) = \int_0^{T_{i,j}} 0.5 m z_2(t)^2$ is the objective function quantifying the energy consumption with mass $m$ and velocity $z_2$. The piece of equipment starts its task at $t = 0$ and has to complete its task before $t = T_{i,j}$. Here, (28) represents the continuous-time dynamics of the piece of equipment as explained in

Section 2.3. We consider the continuous-time dynamics model for the behavior of a piece of equipment as a double integrator:

$$\dot{z}_1(t) = z_2(t), \quad z_1(t) \in [0, S] \tag{29}$$

$$\dot{z}_2(t) = u(t), \quad z_2(t) \in [-v_{\max}, v_{\max}], \quad u(t) \in [-u_{\max}, u_{\max}], \tag{30}$$

where $z_1(t)$(m) and $z_2(t)$(m/s) describe the position and the velocity of equipment, respectively, $u$ (m/s$^2$) represents the acceleration, $S$ is the traveling distance of equipment, $[-v_{\max}, v_{\max}]$ is the constraint on $z_2(t)$, and $u_{\max}$ is the constraint on the acceleration.

**Solving the Control Problem.** The continuous-time dynamic model can be discretized for optimization purposes. Due to the linearity of the original system, the energy consumption problem in that case can be formulated as a standard quadratic programing problem, which can be solved efficiently. The discretized dynamical model, based on (29) and (30) for a piece of equipment is as follows:

$$\mathbf{z}(k+1) = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \mathbf{z}(k) + \begin{bmatrix} 0.5\Delta T^2 \\ \Delta T \end{bmatrix} u(k) = \mathbf{A}\mathbf{z}(k) + \mathbf{B}\mathbf{u}(k). \tag{31}$$

where $\mathbf{z}(k) = \begin{bmatrix} s(k) & v(k) \end{bmatrix}^{\mathrm{T}}$ describes the position $s(k)$ and velocity $v(k)$ of the piece of equipment, and $u(k)$ is the acceleration of the piece of equipment.

The mechanical energy is minimized by means of minimizing the following objective function:

$$J = \sum_{k=1}^{N_s} E(k), \tag{32}$$

where $E(k) = 0.5mz_2(k)^2$ describes the unit kinetic energy of the piece of equipment at time $k$ and $N_s = \frac{T_{i,j}}{T_s} + 1$ is the number of discretized steps, with the time step size $T_s$.

For the initial condition and final condition, $\mathbf{z}(0) = \begin{bmatrix} s(0) & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{z}(N_s) = \begin{bmatrix} s(N_s) & 0 \end{bmatrix}^{\mathrm{T}}$ ($s(0)$ and $s(N_s)$ are the initial state and the final state of position). Besides, the constraints of other states can be written in the following form:

$$\mathbf{z}_{\min} \leq \mathbf{z}(k) \leq \mathbf{z}_{\max}, \ k = 1, ..., N_s - 1, \tag{33}$$

where $\mathbf{z}_{\min}$ and $\mathbf{z}_{\max}$ are the constraints of states on $\mathbf{z}(k)$ .

The prediction model of this discretized dynamical system (31) shown below is used to formulate the standard form of optimization problems.

$$\mathbf{z}(k) = \mathbf{A}^{k-1}\mathbf{z}(0) + \sum_{i=0}^{k-1} \mathbf{A}^k \mathbf{B} u(k-1-i), \ k = 1, ..., N_s \tag{34}$$

The control variables are defined as: $\tilde{\mathbf{u}} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N_s - 1) \end{bmatrix}$

Based on the prediction model in (34) and the definition of the control variables, the constraints of the state $\mathbf{z}$ in (33) can be translated into: $\mathbf{b}_{\min} \leq \tilde{\mathbf{A}}\tilde{\mathbf{u}} \leq \mathbf{b}_{\max}$.

A standard formulation of this energy consumption minimization problem can then be obtained as follows:

$$\min_{\tilde{\mathbf{u}}} \tilde{\mathbf{u}}^{\mathrm{T}}\mathbf{H}\tilde{\mathbf{u}} + \mathbf{f}^{\mathrm{T}}\tilde{\mathbf{u}} \tag{35}$$

subject to

$$\mathbf{b}_{\min} \leq \tilde{\mathbf{A}}\tilde{\mathbf{u}} \leq \mathbf{b}_{\max} \tag{36}$$

$$\tilde{\mathbf{u}}_{\min} \leq \tilde{\mathbf{u}} \leq \tilde{\mathbf{u}}_{\max} \tag{37}$$

where $\mathbf{b}_{\min}$ and $\mathbf{b}_{\max}$ are the lower bound and the upper bound of the linear inequality, respectively, $\tilde{\mathbf{u}}_{\min}$ and $\tilde{\mathbf{u}}_{\max}$ are the lower bound and the upper bound of the control variables. This quadratic programming problem can be solved by several existing solvers, such as quadprog in Matlab and SCIP in the OPTI Toolbox [6]. When this problem is solved, the lower-level controller will set the calculated trajectories as the reference for the piece of equipment.

**Determining the Minimal Time Required.** The hierarchical control architecture as proposed provides a methodology for achieving the minimal makespan in an energy-efficient way. In this control architecture, the minimal time of each task is required at the higher level for scheduling tasks. As the interaction between the higher level and the lower level of the hierarchical control architecture, the lower bound of the time required for a piece of equipment to carry out a task needs to be computed. The minimal time required for doing a task depends on the states and continuous-time dynamics of the piece of equipment. The minimal-time required to complete a certain task $\tau_{i,j}$ can be obtained from the theory of optimal control, as the result of Pontryagin's Minimum Principle. Application of Pontryagin's Minimum Principle results in the minimization of two so-called Hamiltonian functions. Details of the theory behind this principle can be found in [7]. Here we provide the outcome of applying this principle in our setting.

In our setting, application of the principle yields the control action $u(t)$ that minimizes the time for carrying out a task as follows:

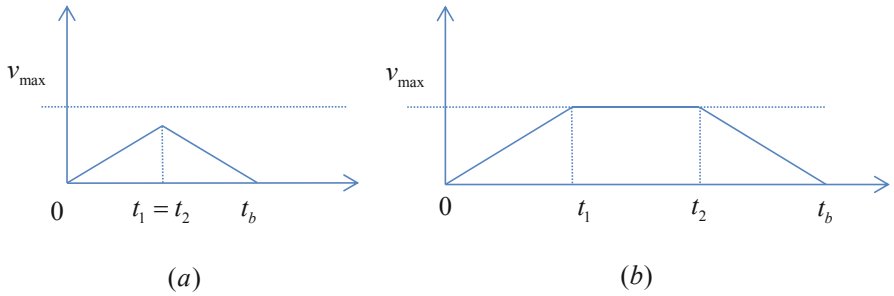
$$u(t) = \begin{cases} -u_{\max} & \text{for } t = t_2^+, \ldots, t_b \\ 0 & \text{for } t = t_1^+, \ldots, t_2^- \\ u_{\max} & \text{for } t = 0^+, \ldots, t_1^- \end{cases} \tag{38}$$

where $t_1$ and $t_2$ are so-called switching points between different control modes, $t_1^+ \geq t_1 + \epsilon$, $t_2^+ \geq t_2 + \epsilon$, $t_1^- \geq t_1 - \epsilon$ and $t_2^- \geq t_2 - \epsilon$ ($\epsilon$ is a small positive value), and where $t_1$ and $t_b$ are calculated as:

$$t_1 = \begin{cases} \frac{v_{\max}}{u_{\max}} & \text{if } S \geq \frac{v_{\max}^2}{u_{\max}} \\ \sqrt{\frac{S}{u_{\max}}} & \text{if } S < \frac{v_{\max}^2}{u_{\max}} \end{cases} \tag{39}$$

$$t_b = \begin{cases} 2\frac{v_{\max}}{u_{\max}} + \frac{S - \frac{v_{\max}^2}{u_{\max}}}{v_{\max}} & \text{if } S \geq \frac{v_{\max}^2}{u_{\max}} \\ 2\sqrt{\frac{S}{u_{\max}}} & \text{if } S < \frac{v_{\max}^2}{u_{\max}} \end{cases} \tag{40}$$

The different minimal-time profiles for carrying out the task with respect to




(*a*) (*b*)

**Fig. 6.** Different minimal-time depending on distance S. (a): $S < \frac{v_{\max}^2}{u_{\max}}$, $(b) : S \geq \frac{v_{\max}^2}{u_{\max}}$.

distance $S$ can be seen in Fig. 6. The minimal-time depends on the relationship between $S$ and $\frac{v_{\max}^2}{u_{\max}}$ (which is obtained based on the optimal control action). $t_b$ is the ending time and the distance $S$ is used as the integration over $[0, t_b]$.

In summary, the minimal-time required to do task $\tau_{i,j}$ can be obtained by Pontryagin's necessary conditions. This minimal-time is given by $t_b$ above. It is the lower bound on the time required for doing a task by a piece of equipment. This bound is sent to the higher level as $s_{i,j}^{\min}$ for scheduling the tasks.

### 3.3 Control Architecture Summary

The control problem for three pieces of equipment is decomposed into three steps. First once the lower-level controller receives the request from the higher level for the time required for doing a task, the lower bound of the transport time is obtained by the lower-level controller by solving a minimal-time control problem. Then the supervisory controller determines the operation time of each piece of equipment. The scheduling problem is formulated by the supervisory controller as a linear programming problem and solved recursively. The operation time allowed for carrying out a task by a piece of equipment is then sent from the supervisory controller to lower-level control of each piece of equipment. The pieces of equipment subsequently carry out the task, possibly also taking into account energy saving objectives. The complete procedure of the hierarchical control structure is as follows:

- the higher level requests the time required for doing a particular task;
- minimal time control of QC, AGV and ASC at the lower level is calculated locally;

- the supervisory controller receives the minimal time from the lower level;
- the supervisory controller computes the schedule for all pieces of equipment;
- the supervisory controller sends the operation transport time to each piece of equipment;
- QC, AGV and ASC receive the transport time to execute the relating task and save energy if possible.

### 3.4 Heuristic Control of Equipment

In earlier works, each piece of equipment is considered to operate at its maximal speed (e.g. [2, 3]). Such a heuristic approach is here modified taking the constraints of the speed and the acceleration into account. The schedule of the supervisory controller is determined in the same way as in Section 3.1. This alternative approach is considered for comparison in the study below.

## 4 Case Study

Next we present simulations in which three pieces of equipment are used to transport containers from the quayside area to the stacking area. The trajectories of the pieces of equipment have to be determined. We compare the performance of the proposed hierarchical control architecture and the heuristic controller. Several assumptions are made in this case study:

- The initial position of the QC is set to its unloading position. The initial position of the AGV is set to its loading position. The initial position of the ASC is set to its loading position;
- Each piece of equipment can only transport one container at a time;
- The service time of the QC, the AGV and the ASC are ignored;
- The length of a QC is 120m, the traveling distance of an AGV is 300m and the moving distance of an ASC is 100m;
- The maximum speed $v_{\max}$ of the QC, AGV and ASC are assumed to be 4 (m/s), 6 (m/s) and 4 (m/s), respectively;
- The maximum acceleration $u_{\max}$ of the QC, AGV and ASC are assumed to be 0.4 ($m/s^2$), 1 ($m/s^2$) and 0.4 ($m/s^2$), respectively;
- The weight of trolley for the QC, the empty AGV and the ASC are assumed to be 10t, 25t and 240t, respectively. The type of one container is considered to be 20TEU with the weight of 25t.

The result of the scheduling problem (11) to (18) in the higher level is shown in Tables 1 and 2. Based on the available operation time for each piece of equipment to transport containers, the minimal energy consumption problem is locally solved as a quadratic programming problem. The reduction of energy consumption are demonstrated in Table 3. The resulted behaviors of each piece of equipment are shown in Figure 7 and Figure 8.

Typically each piece of equipment inside a container terminal is operated at its maximal speed as the basis of the scheduling problem, as outlined in

**Table 1.** The operation schedule associated with the proposed supervisory controller. Each cell shows the transport time $T_{i,j}$ for task $j$ associated to container $i$.

| container no. | The hierarchical controller | | | | | | |
|---|---|---|---|---|---|---|---|
| | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | complete |
| | QC | QC | AGV | ASC | ASC | AGV | all |
| 1 | 40s | 40s | 56s | 56s | 56s | 56s | 192s |
| 2 | 56s | 56s | 56s | 56s | 56s | 56s | 304s |
| 3 | 56s | 56s | 56s | 56s | 56s | 56s | 416s |
| 4 | 56s | 56s | 56s | 56s | 56s | 56s | 528s |

**Table 2.** The operation schedule associated with the heuristic approach. Each cell shows the transport time $T_{i,j}$ for task $j$ associated to container $i$.

| container no. | The heuristic controller | | | | | | |
|---|---|---|---|---|---|---|---|
| | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | complete |
| | QC | QC | AGV | ASC | ASC | AGV | all |
| 1 | 40s | 40s | 56s | 35s | 35s | 56s | 192s |
| 2 | 40s | 40s | 56s | 35s | 35s | 56s | 304s |
| 3 | 40s | 40s | 56s | 35s | 35s | 56s | 416s |
| 4 | 40s | 40s | 56s | 35s | 35s | 56s | 528s |

**Table 3.** The energy consumption resulting from the proposed control architecture. Each cell shows the energy consumption for task $j$ associated to container $i$.

| container no. | The hierarchical controller | | | | | | |
|---|---|---|---|---|---|---|---|
| | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | complete |
| | QC | QC | AGV | ASC | ASC | AGV | all |
| 1 | 80kJ | 280kJ | 900kJ | 475kJ | 456kJ | 450kJ | 2641kJ |
| 2 | 28.8kJ | 100.8kJ | 900kJ | 475kJ | 456kJ | 450kJ | 2410.6kJ |
| 3 | 28.8kJ | 100.8kJ | 900kJ | 475kJ | 456kJ | 450kJ | 2410.6kJ |
| 4 | 28.8kJ | 100.8kJ | 900kJ | 475kJ | 456kJ | 450kJ | 2410.6kJ |

**Table 4.** The energy consumption resulting from the proposed control architecture. Each cell shows the energy consumption for task $j$ associated to container $i$.

| container no. | The heuristic controller | | | | | | |
|---|---|---|---|---|---|---|---|
| | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | complete |
| | QC | QC | AGV | ASC | ASC | AGV | all |
| 1 | 80kJ | 280kJ | 900kJ | 2000kJ | 1920kJ | 450kJ | 5630kJ |
| 2 | 80kJ | 280kJ | 900kJ | 2000kJ | 1920kJ | 450kJ | 5630kJ |
| 3 | 80kJ | 280kJ | 900kJ | 2000kJ | 1920kJ | 450kJ | 5630kJ |
| 4 | 80kJ | 280kJ | 900kJ | 2000kJ | 1920kJ | 450kJ | 5630kJ |

Section 3.2. This heuristic approach is considered for the comparison in this case study. Tables 1 and 2 show the operation time of each piece of equipment with respect to the proposed hierarchical controller and the heuristic controller. Due to the synchronization of different types of equipment in the scheduling

problem, the completion time of the hierarchical controller is the same as of the heuristic controller. In the hierarchical controller, both the QC and the ASC do not need to work at their maximal speed without loss of changes on makespan. The transport time of the QC and the ASC are determined by the synchronization among different pieces of equipment.

Tables 3 and 4 present the energy consumption of each task associated to the piece of equipment by the hierarchical controller and the heuristic controller. In each task, one piece of equipment moves between the loading and the unloading place. It is noticeable that the unit energy consumption regarding both the QC and the ASC is reduced when the hierarchical controller is employed. Apart from the QC and the ASC, the unit energy consumption of the AGV by the proposed hierarchical is the same as the unit energy consumption obtained by the heuristic controller. The reason for this is that the AGV is the bottleneck in the transport of containers, considering the transport time of AGV determines the completion of transporting containers. In this sense, the QC and ASC does not need to work at its maximal speed between the unloading and unloading place. In the heuristic approach, each piece of equipment was working at its maximal speed. There is no energy consumption considered in the heuristic approach. Fig. 7 and Fig. 8 show

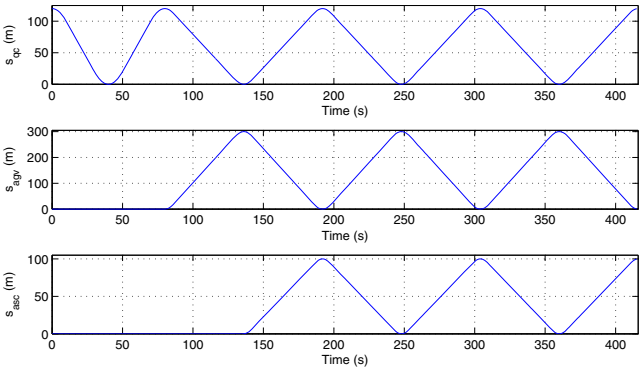



**Fig. 7.** The simulation results with respect to the position of three pieces of equipment by the hierarchical controller architecture

the behaviors of QC, AGV and ASC with respect to the position and the velocity determined by the hierarchical controller. Fig. 7 indicates these three pieces of equipment are well synchronized. Fig. 8 also presents the synchronization of these three pieces of equipment. It is seen from Fig. 8 that both the QC and the AGV were operated at their maximal operational conditions, i.e., the maximal velocity and the maximal acceleration, before the interaction of three pieces of equipment. When these three pieces of equipment are synchronized, the QC and the ASC were not working at their maximal speed to save energy consumption while the AGV was running at its maximal working condition.

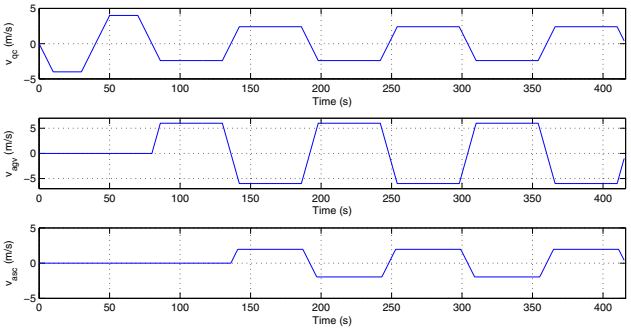



**Fig. 8.** The simulation results with respect to the velocity of three pieces of equipment by the hierarchical controller architecture

## 5 Conclusions and Future Research

In this paper, the dynamics of transporting containers in a container terminal are considered as the combination of including continuous-time and discrete-event dynamics. A hierarchical control architecture is proposed, consisting of two levels. The higher level takes care of task scheduling; the lower level consists of controller per piece of equipment for an optimal control problem. At the higher level, the minimal makespan is obtained; at the lower level, the energy consumption reduction is achieved if possible, while satisfying time constraints on carrying out a task given by the higher level. A simulation illustrates how the minimal makespan can be achieve in an energy-efficient way by the proposed hierarchical control structure. Future work will extend the application toward the transport of multiple containers by more than three pieces of equipment, taking the potential collision of AGVs into account, and detailing the models used to represent the continuous-time dynamics.

**Acknowledgements.** This research is supported by the China Scholarship Council under Grant 2010628012 and the VENI project "Intelligent multi-agent control for flexible coordination of transport hubs" (project 11210) of the Dutch Technology Foundation STW, a subdivision of the Netherlands Organization of Scientific Research (NWO).

## References

1. Alessandri, A., Cervellera, C., Cuneo, M., Gaggero, M., Soncin, G.: Modeling and feedback control for resource allocation and performance analysis in container terminals. IEEE Transactions on Intelligent Transportation Systems 9(4), 601–614 (2008)
2. Bielli, M., Boulmakoul, A., Rida, M.: Object oriented model for container terminal distributed simulation. European Journal of Operational Research 175(3), 1731–1751 (2006)

3. Cao, J.X., Lee, D., Chen, J.H., Shi, Q.: The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. Transportation Research Part E: Logistics and Transportation Review 46(3), 344–353 (2010)
4. Contu, F., Di Febbraro, A., Sacco, N.: A model for performance evaluation and sensitivity analysis of seaport container terminals. In: Proceedings of the 18th IFAC World Congress, Milan, Italy, pp. 13870–13875 (2011)
5. Crainic, T.G., Kim, K.H.: Intermodal transportation. In: Barnhart, C., Laporte, G. (eds.) Handbooks in Operations Research and Management Science. Transporation, vol. 14, pp. 467–536. Elsevier, Amsterdam (2007)
6. Currie, J., Wilson, D.I.: OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In: Sahinidis, N., Pinto, J. (eds.) Foundations of Computer-Aided Process Operations, Savannah, Georgia, USA, January 8-11 (2012)
7. Geering, H.P.: Optimal Control with Engineering Applications. Springer, Berlin (2007)
8. Godbole, D.N., Lygeros, J., Sastry, S.: Hierarchical hybrid control: A case study. In: Proceedings of the 34th IEEE Conference on Decision and Control, Louisiana, New Orleans (1994)
9. Ha, B., Park, E., Lee, C.: A simulation model with a low level of detail for container terminals and its applications. In: Proceedings of the 2007 Winter Simulation Conference, Washington, D.C., pp. 2003–2011 (December 2007)
10. Davidsson, P., Henesey, L., Persson, J.: Agent based simulation architecture for evaluating operational policies in transshipping containers. Autonomous Agents and Multi-Agent Systems 18(2), 220–238 (2009)
11. Ramadge, P.J.G., Wonham, W.M.: The control of discrete event dynamical systems. Proceedings of IEEE 77, 81–98 (1989)
12. Stahlbock, R., Voß, S.: Operations research at container terminals: a literature update. OR Spectrum 30(1), 1–52 (2007)
13. van Duin, J.H.R., Geerlings, H.: Estimating CO2 footprints of container terminal port-operations. International Journal of Sustainable Development and Planning 6(4), 459–473 (2011)
14. Wijlhuizen, E.T.J., Meeussen, F.: The sustainable container terminal (Maasvlakte II). In: Proceedings of the 2nd International Conference on Harbours, Air Quality and Climate Change, Rotterdam, The Netherlands (2008)
15. Xiao, F.Y., Li, P.K., Chun, H.C.: A distributed agent system for port planning and scheduling. Advanced Engineering Informatics 25(3), 403–412 (2011)
16. Xin, J., Negenborn, R.R., Lodewijks, G.: Hybrid MPC for balancing throughput and energy consumption in an automated container terminal. In: Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems, The Hague, The Netherlands (2013)

# A Node Flow Model for the Inflexible Visitation Liner Shipping Fleet Repositioning Problem with Cargo Flows

Kevin Tierney and Rune Møller Jensen

IT University of Copenhagen, Copenhagen, Denmark
{kevt,rmj}@itu.dk

**Abstract.** We introduce a novel, node flow based mathematical model for the fixed-time version of a central problem in the liner shipping industry called the Liner Shipping Fleet Repositioning Problem (LSFRP). We call this version of the problem the Inflexible Visitation LSFRP (IVLSFRP). During repositioning, vessels are moved between routes in a liner shipping network. Shipping lines wish to reposition vessels as cheaply as possible without disrupting the cargo flows of the network. The LSFRP is characterized by chains of interacting activities with a multi-commodity flow over paths defined by the activities chosen. We introduce two versions of a node flow based model that exploit the fixed activity times of the IVLSFRP's graph to handle cargo demands on the nodes of the graph, instead of the arcs, significantly reducing the number of variables. Using this model in CPLEX, we are able to solve 12 previously unsolved IVLSFRP instances to optimality. Additionally, we improve the solution time on every instance in the IVLSFRP dataset, sometimes by several orders of magnitude.

## 1 Introduction

Liner shipping networks are the lifeblood of the world economy, providing cheap and reliable freight services between nations around the globe. Vessels are regularly repositioned between services in liner shipping networks to adjust networks to the continually changing world economy. Repositioning vessels involves creating a plan for a set of vessels out of a number of cost saving (or revenue earning) activities that moves (*repositions*) the vessels to a particular route in the network. Since repositioning a single vessel can cost hundreds of thousands of US dollars, optimizing the repositioning activities of vessels is an important problem for the liner shipping industry.

The Liner Shipping Fleet Repositioning Problem (LSFRP) with cargo flows, first introduced in [18], consists of finding sequences of activities that move vessels between services in a liner shipping network while respecting the cargo flows of the network. The LSFRP maximizes the profit earned on the subset of the network affected by the repositioning, balancing sailing costs and port fees against cargo and equipment revenues, while respecting important liner shipping

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 18–34, 2013.

specific constraints dictating the creation of services and movement of cargo. The Inflexible Visitation LSFRP (IVLSFRP) is a simplified version of the LSFRP in which the time of every visitation (i.e. port call) a vessel may undertake is known in advance. Such visitations are called *inflexible*. In the full LSFRP, the vessel entry/exit time of some visitations is unknown, requiring solution methods to schedule these visitations if they are chosen for a repositioning.

The number of variables in the arc flow model from [16,18] grows linearly in the number of graph arcs multiplied by the number of cargo demands, and, as a result, it quickly runs out of memory or CPU time on large instances. In this paper, we introduce a novel node flow model for the IVLSFRP that fits into memory even on large instances, thanks to the fact that the number of nodes tends to be significantly lower than the number of arcs. The node flow model exploits the fact that when all visitations are inflexible, the sequencing of demands is known in advance. We use this fact in order to model demands based on the graph nodes they can pass through. We provide two versions of the node flow model that each handle equipment (i.e. empty container) flows in different ways; one version uses an arc flow formulation, and the other embeds equipment into the problem's demand structure.

The node flow model solves 12 previously unsolved IVLSFRP instances to optimality within 5 hours. The node flow model solves IVLSFRP instances to optimality significantly faster than the arc flow approach, and is even able to solve a number of instances in under a second that require 15 minutes or more with the arc flow approach.

This paper is organized as follows. We first give an overview of the IVLSFRP in Section 2, followed by the arc flow model presented in [16] in Section 3. We then introduce our node based formulation of the IVLSFRP in Section 4. We provide a computational evaluation of the new model with IBM CPLEX 12.4 in Section 5 showing the improved performance of the node flow approach. Finally, we conclude in Section 6.

## 2 Liner Shipping Fleet Repositioning

We briefly describe the IVLSFRP, and refer readers to [16] for a more detailed description, as well as a description of the full version of the LSFRP. Liner shipping networks consist of a set of cyclical routes, called services, that visit ports on a regular, usually weekly, schedule. Liner shipping networks are designed to serve customer's cargo demands, but over time the economy changes and liner shippers must adjust their networks in order to stay competitive. Liner shippers add, remove and modify existing services in their network in order to make changes to the network. Whenever a new service is created, or a service is expanded, vessels must be *repositioned* from their current service to the service being added or expanded.

Vessel repositioning is expensive due to the cost of fuel (in the region of hundreds of thousands of dollars) and the revenue lost due to cargo flow disruptions. Given that liner shippers around the world reposition hundreds of vessels per

year, optimizing vessel movements can significantly reduce the economic and environmental burdens of containerized shipping, and allow shippers to better utilize repositioning vessels to transport cargo. The aim of the IVLSFRP is to maximize the profit earned when repositioning a number of vessels from their initial services to a service being added or expanded, called the goal service.

Liner shipping services are composed of multiple *slots*, each of which represents a cycle that is assigned to a particular vessel. Each slot is composed of a number of *visitations*, which can be thought of as port calls, i.e., a specific time when a vessel is scheduled to arrive at a port. A vessel that is assigned to a particular slot sequentially sails to each visitation in the slot.

Vessel sailing speeds can be adjusted throughout repositioning to balance cost savings with punctuality. The bunker fuel consumption of vessels increases cubically with the speed of the vessel. *Slow steaming*, in which vessels sail near or at their minimum speed, therefore, allows vessels to sail more cheaply between two ports than at higher speeds, albeit with a longer duration (see, e.g., [12]).

**Phase-out and Phase-in.** The repositioning period for each vessel starts at a specific time when the vessel may cease normal operations, that is, it may stop sailing to its scheduled visitations and go somewhere else. Each vessel is assigned a time when it may begin its repositioning, called its *phase-out* time. After this time, the vessel may undertake a number of different activities to reach the goal service. In order to complete the repositioning, each vessel must join slot on the goal service before a time set by the repositioning coordinator, called the *phase in* time. After this time, normal operations on the goal service begin, and all scheduled visitations on the service are to be undertaken. In other words, the repositioning of each vessel and optimization of its activities takes place in the period between two fixed times, the vessel's earliest phase-out time and the latest phase-in time of all vessels.

**Cargo and Equipment.** Revenue is earned through delivering *cargo* and *equipment* (typically empty containers). We use a detailed view of cargo flows. Cargo is represented as a set of port to port demands with a cargo type, a latest delivery time, an amount of TEU[1] available, and a revenue per TEU delivered. We subtract the cost of loading and unloading each TEU from the revenue to determine the profit per TEU of a particular cargo demand. In contrast to cargo, which is a multi-commodity flow where each demand is a commodity with an origin and destination port, equipment can be sent from any port where it is in surplus to any port where it is in demand. We consider cargo demands and equipment consisting of either *dry* or *reefer* (refrigerated) cargo. Vessels have a limited capacity, and are therefore assigned a maximum number of reefer containers and a maximum number of all types of containers.

**Sail-On-Service (SOS) Opportunities.** While repositioning, vessels may use certain services to cheaply sail between two parts of the network. These are called *SOS opportunities*. The two vessels involved in SOS opportunities are referred to

[1] TEU stands for *twenty-foot equivalent unit* and represents a single twenty-foot container.

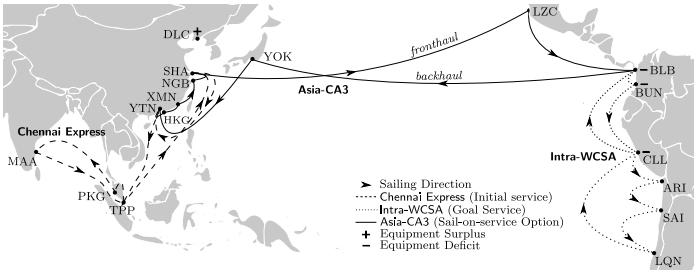



**Fig. 1.** A subset of a case study of the LSFRP, from [17]

as the *repositioning vessel*, which is the vessel under the control of a repositioning coordinator, and the *on-service vessel*, which is the vessel assigned to a slot on the service being offered as an SOS opportunity. Repositioning vessels can use SOS opportunities by replacing the on-service vessel and sailing in its place for a portion of the service. SOS opportunities save significant amounts of money on bunker fuel, since one vessel is sailing where there would have otherwise been two. Using an SOS can even sometimes earn money through the leasing of the on-service vessel. Using an SOS is subject to a number of constraints, which are described in full in [16].

**Asia-CA3 Case Study.** Figure 1 shows a subset of a real repositioning scenario in which a vessel must be repositioned from its initial service (the phase-out service), the Chennai-Express, to the goal service (the phase-in service), the Intra-WCSA. The Asia-CA3 service is offered as a SOS opportunity to the vessel repositioning from Chennai Express to Intra-WCSA. One possible repositioning could involve a vessel leaving the Chennai Express at TPP, and sailing to HKG where it can pick up the Asia-CA3, thereby replacing the on-service vessel. The repositioning vessel would then sail along the Asia-CA3 until it gets to BLB, where it can join the Intra-WCSA. Note that no vessel sails on the backhaul of the Asia-CA3, and this is allowed because very little cargo travels on the Asia-CA3 towards Asia.

### 2.1 Literature Review

The LSFRP has recently begun to receive attention in the literature, but it was not mentioned in either of the most influential surveys of work in the liner shipping domain [6,7] or container terminals [14]. Although there has been significant work on problems such as the Fleet Deployment Problem (FDP) (e.g., [13]) and the Network Design Problem (NDP) (e.g. [1,11]), these problems deal with strategic decisions related to building the network and assigning vessels to services, rather than the operational problem of finding paths for vessels through the network. Additionally, the vessel schedule recovery problem (VSRP) [3] differs in that it lacks the cost saving activities of the LSFRP due to its short time

window. Andersen's PhD thesis [2] discusses a problem similar to the LSFRP, called the Network Transition Problem (NTP), but provides no mathematical model or formal problem description, nor does the problem handle slow steaming, empty equipment flows, or SOS opportunities.

Although tramp shipping problems, such as [5,10], maximize cargo profit while accounting for sailing costs and port fees as in the LSFRP, they lack liner shipping specific constraints, such as sail-on-service opportunities, phase-in requirements and strict visitation times. Airline disruption management (see [8,9]), while also relying on time-based graphs, differs from the LSFRP in that airline disruption management requires an exact cover of all flight legs over a planning horizon. The LSFRP has no such requirement over visitations or sailing legs.

The primary previous work on the LSFRP in the literature is found in [17], [18], [16] and [15], by the authors. The first work on the LSFRP, [17], solved an abstraction of the LSFRP without cargo/equipment flows and SOS parallel sailings using a hybrid of automated planning and linear programming called Linear Temporal Optimization Planning (LTOP). However, LTOP and other automated planning methods are unable to model cargo flows and are thus inapplicable to the version of the LSFRP we solve in this work. A mathematical model of the LSFRP with cargo and equipment flows is introduced in [18], and CPLEX is used to solve the model. In [16], a simulated annealing algorithm is used to solve instances that CPLEX was unable to solve. Finally, in [15] it is shown that the late acceptance hill climbing technique from [4] does not outperform simulated annealing on the full LSFRP. Unlike in [18], [16] and [15], in this work we only investigate a subset of the LSFRP that does not include flexible time windows.

## 3 Arc Flow Mathematical Model

We describe the mathematical model of the LSFRP with cargo flows from [18] and [16], in which demands are modeled using variables representing the amount of flow of each demand on each arc in the graph.

### 3.1 Graph Description

We give an overview of the graph used in the model of the IVLSFRP with cargo flows, and refer to [16] for details. The graph is based off of the idea of modeling each visitation as a node in a graph, with arcs representing sailings between visitations. We begin by letting $V$ be the set of visitations for all vessels, and defining $S$ as the set of ships.

The overall structure of the graph involves three sets of visitations: phase-out, phase-in, and SOS visitations. The three types of visitations represent three disjoint sets that make up $V$. In addition to these visitations, we include a graph sink, $\tau$, which all vessels must reach for a valid repositioning. We let $V' = V \setminus \tau$ be the set of all graph visitations excluding $\tau$. We now describe the arc structure present in each of the three sets of visitations.

**Phase-out and Phase-in.** Each ship is assigned a particular visitation, $\sigma_s \in V'$, at which the ship $s \in S$ begins its repositioning. This visitation represents the earliest allowed phase-out time for that vessel. A visitation is then created for each subsequent port call of the ship on its phase-out slot. Each phase-out visitation is connected to the next one with an arc. Note that phase-out visitations do not connect to the phase-out visitations of other ships.

Vessels may leave phase-out nodes to sail to SOS opportunities or to a phase-in slot. Thus, arcs are created from each phase-out visitation to each phase-in visitation and SOS entry visitation such that sailing between the visitations is temporally feasible (i.e. the starting time of the phase-in/SOS visitation is greater than the end time of the phase-out visitation plus the sailing time). Finally, phase-out visitations have incoming arcs from phase-in visitations in the same trade zone, which we define as a set of ports in the same geographical region. This allows ships to avoid sailing back and forth between ports when transferring directly between the phase-out and phase-in.

We create visitations for each port call along a phase-in slot, and connect subsequent phase-in visitations to each other. The final visitation in a slot, which represents the time at which regular operations must begin on a service, is connected to the graph sink, $\tau$. Due to the node disjointness of the vessel paths, this structure ensures that each slot on the goal service receives a single vessel.

**Sail-On-Service.** SOS opportunities are modeled with a special structure that ensures each one is only used by at most a single vessel. The structure partitions the visitations of an SOS into three sets: *entry ports*, where vessels may join the SOS, *through ports*, in which a vessel must already be on the SOS, and *end ports* where a vessel may leave the SOS.

**Sailing Cost.** The fuel consumption of a ship is a cubic function of the speed of the vessel. We precompute the optimal cost for each arc using a linearized bunker consumption function. All arcs in the model are assigned a sailing cost for each ship that is the optimal sailing cost given the total duration of the arc. Since ships have a minimum speed, if the duration of the arc is greater than the time required to sail on the arc at a ship's minimum speed, the cost is calculated using the minimum speed and then the ship simply waits for the remainder of the duration. This is a common practice for shipping lines in order to add buffer to their schedules, thus making the network more robust to disruptions.

### 3.2 Mixed-Integer Programming Model

We now present the MIP model from [18,16] excluding flexible node/arc components. We use the following parameters and variables for the model.

**Parameters**

| | |
|---|---|
| $T$ | Set of equipment types. $T = \{dc, rf\}$. |
| $S$ | Set of ships. |
| $V'$ | Set of visitations minus the graph sink, $\tau$. |

| | |
|---|---|
| $\sigma_s \in V'$ | Starting visitation of vessel $s \in S$. |
| $V^{t+}$ | Set of visitations with an equipment surplus of type $t$. |
| $V^{t-}$ | Set of visitations with an equipment deficit of type $t$. |
| $V^{t^*}$ | Set of visitations with an equipment surplus or deficit of type $t$ ($V^{t^*} = V^{t^+} \cup V^{t^-}$). |
| $In(i) \subseteq V'$ | Set of visitations with an arc connecting to visitation $i \in V$. |
| $Out(i) \subseteq V'$ | Set of visitations receiving an arc from $i \in V$. |
| $c_i^{Mv} \in \mathbb{R}^+$ | Cost of a TEU move at visitation $i \in V'$. |
| $f_{s,i}^{Port} \in \mathbb{R}$ | Port fee associated with vessel $s$ at visitation $i \in V'$. |
| $r_t^{Eqp} \in \mathbb{R}^+$ | Revenue for each TEU of equipment of type $t \in T$ delivered. |
| $u_s^t \in \mathbb{R}^+$ | Capacity of vessel $s$ for cargo type $t \in T$. Note that $u_s^{dc}$ is the capacity of all slots on the vessel, including reefer slots. |
| $A'$ | Set of arcs $(i,j) \in A$, where $i,j \in V'$. |
| $c_{i,j}^s$ | Fixed cost of vessel $s$ utilizing arc $(i,j) \in A'$. |
| $(o,d,t) \in \Theta$ | A demand triplet, where $o \in V', d \subseteq V'$ and $t \in T$ are the origin visitation, destination visitations and the cargo type, respectively. |
| $a^{(o,d,t)} \in \mathbb{R}^+$ | Amount of demand available for the demand triplet. |
| $r^{(o,d,t)} \in \mathbb{R}^+$ | Amount of revenue gained per TEU for the demand triplet. |

**Variables**

| | |
|---|---|
| $x_{i,j}^{(o,d,t)} \in [0, a^{(o,d,t)}]$ | Amount of flow of demand triplet $(o,d,t) \in \Theta$ on $(i,j) \in A'$. |
| $x_{i,j}^t \in [0, \max_{s \in S} u_s^{dc}]$ | Amount of equipment of type $t \in T$ flowing on $(i,j) \in A'$. |
| $y_{i,j}^s \in \{0,1\}$ | Indicates whether vessel $s$ is sailing on arc $(i,j) \in A$. |

**Objective and Constraints**

$$\max \; -\sum_{s \in S} \sum_{(i,j) \in A'} c_{i,j}^s y_{i,j}^s - \sum_{j \in V'} \sum_{i \in In(j)} \sum_{s \in S} f_{s,j}^{Port} y_{i,j}^s \tag{1}$$

$$+ \sum_{(o,d,t) \in \Theta} \left( \sum_{j \in d} \sum_{i \in In(j)} \left( r^{(o,d,t)} - c_o^{Mv} - c_j^{Mv} \right) x_{i,j}^{(o,d,t)} \right) \tag{2}$$

$$+ \sum_{t \in T} \left( \sum_{i \in V^{t+}} \sum_{j \in Out(i)} \left( r_t^{Eqp} - c_i^{Mv} \right) x_{i,j}^t - \sum_{i \in V^{t-}} \sum_{j \in In(i)} c_i^{Mv} x_{j,i}^t \right) \tag{3}$$

$$\text{s.t.} \sum_{s \in S} \sum_{i \in In(j)} y_{i,j}^s \leq 1 \qquad \forall j \in V' \tag{4}$$

$$\sum_{i \in Out(\sigma_s)} y_{\sigma_s,i}^s = 1 \qquad \forall s \in S \tag{5}$$

$$\sum_{i \in In(\tau)} \sum_{s \in S} y_{i,\tau}^s = |S| \tag{6}$$

$$\sum_{i \in In(j)} y_{i,j}^s - \sum_{i \in Out(j)} y_{j,i}^s = 0 \qquad \forall j \in \{V' \setminus \bigcup_{s \in S} \sigma_s\}, s \in S \tag{7}$$

$$\sum_{(o,d,rf)\in\Theta} x_{i,j}^{(o,d,rf)} - \sum_{s\in S} u_k^{rf} y_{i,j}^{s} \leq 0 \qquad \forall (i,j) \in A' \tag{8}$$

$$\sum_{(o,d,t)\in\Theta} x_{i,j}^{(o,d,t)} + \sum_{t'\in T} x_{i,j}^{t'} - \sum_{s\in S} u_s^{dc} y_{i,j}^{s} \leq 0 \qquad \forall (i,j) \in A' \tag{9}$$

$$\sum_{i\in Out(o)} x_{o,i}^{(o,d,t)} \leq a^{(o,d,t)} \sum_{i\in Out(o)} \sum_{s\in S} y_{o,i}^{s} \qquad \forall (o,d,t) \in \Theta \tag{10}$$

$$\sum_{i\in In(j)} x_{i,j}^{(o,d,t)} - \sum_{k\in Out(j)} x_{j,k}^{(o,d,t)} = 0 \qquad \forall (o,d,t) \in \Theta, j \in V' \setminus (o \cup d) \tag{11}$$

$$\sum_{i\in In(j)} x_{i,j}^{t} - \sum_{k\in Out(j)} x_{j,k}^{t} = 0 \qquad \forall t \in T, j \in V' \setminus V^{t^*} \tag{12}$$

The domains of the variables are as previously described. The objective consists of four components. First, objective (1) takes into account the precomputed sailing costs on arcs between inflexible visitations and the port fees at each visitation. Note that the fixed sailing cost on an arc does not only include fuel costs, but can also include canal fees or the time-charter bonus for entering an SOS. Second, the profit from delivering cargo (2) is computed based on the revenue from delivering cargo minus the cost to load and unload the cargo from the vessel. Note that the model can choose how much of a demand to deliver, even choosing to deliver a fractional amount. We can allow this since each demand is an aggregation of cargo between two ports, meaning at most one container between two ports will be fractional. Third, equipment profit is taken into account in (3). Equipment is handled similarly to cargo, except that equipment can flow from any port where it is in supply to any port where it is in demand.

Multiple vessels are prevented from visiting the same visitation in constraints (4). The flow of each vessel from its source node to the graph sink is handled by constraints (5), (6) and (7), where (5) starts the vessel flow, (6) ensures that all vessels arrive at the sink, and (7) balances the flow at each node.

Arcs are assigned a capacity if a vessel utilizes the arc in constraints (8), which assigns the reefer container capacity, and in (9), which assigns the total container capacity, respectively. Note that constraints (8) do not take into account empty reefer equipment, since empty containers do not need to be turned on, and can therefore be placed anywhere on the vessel. Cargo is only allowed to flow on arcs with a vessel in constraints (10). The flow of cargo from its source to its destination, through intermediate nodes, is handled by constraints (11). Constraints (12) balance the flow of equipment in to and out of nodes. Since the amount of equipment carried is limited only by the capacity of the vessel, no flow source/sink constraints are required.

## 4 Node Flow Model

When the number of demands and arcs grows, the number of variables in the arc flow model can become too large to fit in memory, and even when the model fits in memory, it is still often too large to solve. In both [18] and [16], the authors are unable to solve the LSFRP to optimality on problems with more than 9 vessels.

The instances with 9 vessels or more all have a graph of around 10,000 arcs or more, and the number of demands is above 400. Problems of such size require at least four million variables just to model the demand flow, making the problems difficult to solve. In contrast, the number of nodes in the graph is much less than the number of arcs (between 300 and 400 nodes on large instances), meaning a model that can take advantage of flows using knowledge of the nodes used along a path can significantly reduce the number of variables required.

We provide two different node flow based models of the IVLSFRP, each of which uses a different modeling of equipment. In our first model, which we call the *equipment as flows* model, we do not change the way equipment is modeled from the arc flow model. We also provide a model in which we model equipment flows as demands, which we call the *equipment as demands* model.

In order to prevent a vessel from carrying too many containers, the amount of containers loaded on the vessel must be accounted for throughout its repositioning. In the arc flow model, this is done by explicitly keeping track of the amount of demand flowing on each arc. In contrast, the node flow model is able to keep count of the number of containers on the vessel implicitly based on the visitations on a vessel's path, and therefore only needs to determine how many containers from a particular demand are flowing. That is, instead of a variable for each demand on each arc, the node flow model has a variable for each demand on each vessel. In order to ensure a vessel is not overloaded over the course of its repositioning, it suffices to ensure it is not overloaded as it enters each visitation on its path, which corresponds to constraining the incoming arcs of a visitation. Since demands must be loaded and unloaded at visitations, which in the IVLSFRP have a fixed begin and end time, the times a demand can be loaded and unloaded are fixed as well. We can therefore determine which demands can be carried on each arc with a reachability analysis.

Since we know in advance which demands can be on the vessel at what times and where, we can post constraints for each visitation to ensure the vessel is not overloaded without explicitly modeling the path of the vessel. These constraints represent the state of a vessel as it enters a visitation, and they neither over or under constrain the problem. They are clearly sufficient to prevent the vessel from being loaded over capacity, since they cover all demands that can possibly be on a vessel as it enters each visitation on its path. They do not over constrain the problem because only those demands which can be loaded on the vessel are constrained. Due to the fixed visitations times, there is never a situation in which two demands are loaded in sequence on one potential vessel path, and are loaded simultaneously on a different path. This means an optimal solution can always be found, if the problem is not infeasible. Consider the following example.

*Example 1.* Figure 2 shows two graphs. In the first graph (a), node $b$ has no fixed visitation time and must be scheduled, and in the second graph (b), all nodes have a fixed visitation time. A single vessel must sail through the graph, starting at $a$ and ending at $d$. The demands in the instance are $\Theta = \{(a, c), (b, d)\}$[2].

[2] We ignore container types, as they are not relevant.


(a) A non-fixed time graph. (b) A fixed time graph.

**Fig. 2.** Subsets of an LSFRP graph with potential vessel paths (red, blue)

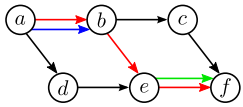



**Fig. 3.** Sequential demand delivery in a multiple demand destination problem

First, consider the non-fixed time case in Figure 2(a). The red path $(a, b, c, d)$ and blue path $(a, c, b, d)$ show two potential voyages of a vessel. On the red path, demand $(a, c)$ is loaded while the vessel is at $a$, and then the vessel continues to $b$, where it loads $(b, d)$. Thus, on the red path both demands are loaded on the vessel simultaneously. On the blue path, the vessel first loads the $(a, c)$ demand, sails to $c$ where it is delivered, and continues to $b$ where it loads $(b, d)$. In this case, the demands are loaded sequentially. Now consider the fixed time case in Figure 2(b), in which the time when $b$ occurs is known in advance. The red path visits nodes $a, b, c, d$, and demand $(a, c)$ and $(b, d)$ are on the ship simultaneously at $b$. In fact, there is no path in the fixed time case where $(a, c)$ and $(b, d)$ can be loaded sequentially; they are either both on the ship or only $(a, c)$ is on the ship.

In the LSFRP, a single demand can be delivered to any visitation in a set of destinations. These destinations all correspond to a single real-world port being visited by different services at different times. The arc flow model takes these multiple destinations into account by simply having each destination visitation of a demand act as a sink for that demand. This is advantageous for the model, since modeling each origin-destination pair individually would require many variables. However, in the node flow model, multiple destinations can cause a situation in which certain incoming arcs of nodes are over-constrained. This could result in the optimal solution not being found.

*Example 2.* Consider Figure 3, which shows a graph with the vessel path $a, b, e, f$ shown with red, and the demands $\{(a, \{b, f\}), (e, \{f\})\}$. Since it is possible for both demands to be flowing on arc $(e, f)$, a constraint must be posted ensuring that $x^{(a,\{b,f\})} + x^{(e,\{f\})} \leq u_s^{dc}$. When a vessel travels on the path shown, the first demand, rather than being delivered to $f$, is dropped off at $b$. The vessel then continues to $e$ where it loads the second demand. Consider the case where the capacity of the vessel is 50 TEU and both demands have 50 TEU available. Due to the constraint we must post on arc $(e, f)$, we can only take a maximum of 50 TEU of both demands, even though it is possible to carry both demands in full.

We remedy this problem by splitting each multiple destination demand into a demand for each origin-destination pair, and add a constraint in the node flow model to ensure that the amount of containers delivered to all of the destinations is not greater than the amount of demand available. In the following models, we use the set $\Theta'$ to represent the set of single origin-destination pairs of demands in the following node flow models. Formally, let $\Theta' = \bigcup_{(o,d,t)\in\Theta} \bigcup_{d'\in d}(o, d', t)$.

### 4.1 Preprocessing

We base our node based model on the same graph and notation as in Section 3. In order to model the flows based on nodes, we must determine which nodes a particular demand can traverse. We do this using a simple reachability analysis based on the transitive closure of the graph. Let $G^T = (V^T, A^T)$ be the transitive closure of the graph $G = (V, A)$, and

$$\Theta_i^{Vis'} = \{(o, d, t) \in \Theta' \mid (o, i) \in A^T \wedge \exists d' \in d \text{ s.t. } \big((i, d') \in A^T \vee i = d'\big)\}.$$

Each node is thereby assigned a set of demands ($\Theta_i^{Vis'}$) based on whether the node is reachable from the demand origin and at least one of the destinations of the demand. We further define $\Theta_{i,rf}^{Vis'} = \{(o, d, t) \in \Theta_i^{Vis'} \mid t = rf\}$ to be the set of reefer demands at node $i \in V'$. Using these sets of demands, we can now model demand flows on the nodes of the IVLSFRP graph.

### 4.2 Equipment as Flows

We extend the parameters in Section 3.2 with the following three parameters: $\Theta'$, the set of single origin single destination demands, $\Theta_i^{Vis'}$, which is the set of demands (dry and reefer) that could traverse node $i$, and $\Theta_{i,rf}^{Vis'}$, the set of reefer demands that could traverse node $i$.

#### Variables

| | |
|---|---|
| $x_{i,j}^t \in [0, \max_{s\in S} u_s^{dc}]$ | Amount of equipment of type $t \in T$ flowing on $(i, j) \in A'$. |
| $y_{i,j}^s \in \{0, 1\}$ | Indicates whether vessel $s$ is sailing on arc $(i, j) \in A$. |
| $z_s^{(o,d,t)} \in [0, a^{(o,d,t)}]$ | Amount of demand triplet $(o, d, t) \in \Theta'$ carried on ship $s \in S$. |

#### Objective and Constraints

$$\max \sum_{s\in S} \sum_{(o,d,t)\in\Theta'} \left(r^{(o,d,t)} - c_o^{Mv} - c_d^{Mv}\right) z_s^{(o,d,t)} - (1) + (3) \tag{13}$$

s. t. (4); (5); (6); (7); (12);

$$z_s^{(o,d,t)} \leq a^{(o,d,t)} \sum_{i\in Out(o)} y_{o,i}^s \qquad \forall (o, d, t) \in \Theta', s \in S \tag{14}$$

$$z_s^{(o,d,t)} \leq a^{(o,d,t)} \sum_{d'\in d} \sum_{i\in In(d')} y_{id'}^s \qquad \forall (o, d, t) \in \Theta', s \in S \tag{15}$$

$$\sum_{(o,d,t)\in\Theta_i^{Vis'}} z_s^{(o,d,t)} + \sum_{t\in T}\sum_{j\in In(i)} x_{i,j}^t \le u_s^{dc} \qquad \forall s\in S, i\in V' \tag{16}$$

$$\sum_{(o,d,t)\in\Theta_{i,rf}^{Vis'}} z_s^{(o,d,t)} \le u_s^{rf} \qquad \forall s\in S, i\in V' \tag{17}$$

$$\sum_{t\in T} x_{i,j}^t \le \sum_{s\in S} u_s^{dc} y_{i,j}^s \qquad \forall (i,j)\in A' \tag{18}$$

$$\sum_{d\in d'} z_s^{(o,d',t)} \le a^{(o,d,t)} \qquad \forall s\in S, (o,d,t)\in\Theta'. \tag{19}$$

The objective (13) contains the same calculation of sailing costs, equipment profits, and port fees as in the arc flow model. However, the demand profit is now computed using the demand flow variables. Note that unlike in $\Theta$, all $(o,d,t)\in\Theta'$ have $|d|=1$. Thus, the constant $c_d^{Mv}$ refers to the cost at a particular visitation.

We copy constraints (4) through (7) and (12) directly from the arc flow model in order to enforce node disjointness along vessel paths and create the vessel and equipment flows. We refer readers to Section 3.2 for a full description of these constraints.

Constraints (14) and (15) allow a demand to be carried only if a particular vessel visits both the origin and a destination of the demand. Note that we do not need to limit the demands to be taken only by a single vessel because of the node disjointness enforced by constraints (4). Only a single vessel can enter the origin node of a demand, ensuring that only one vessel can carry a demand.

In constraints (16) and (17) we ensure that the capacity of the vessel is not exceeded at any node in the graph in terms of all containers and reefer containers, respectively. Equipment flows are handled in the dry capacity constraints (16). Due to the equipment balance constraints, ensuring that the equipment capacity is not exceeded at a node is sufficient for ensuring that the vessel is not overloaded.

Constraints (18) prevent equipment from flowing on any arc that does not have a ship sailing on it. When a vessel utilizes an arc, the constraint allows as much equipment to flow as the capacity of the vessel. When an arc has no vessel, the corresponding equipment flow variables have an upper bound of 0. And, finally, constraints (19) ensure that the amount of demand carried for each single origin-destination demand does not exceed the amount of containers that are actually available.

### 4.3 Equipment as Demands

As an alternative to modeling equipment as flows, we present a model that creates demands for each equipment pair and adds them to $\Theta'$. We let $\Theta_t^{E'} = \{(o,\{d\},dc) \mid o\in V^{t+}\wedge d\in V^{t-}\}$ be the set of demands corresponding to every pair of visitations with an equipment surplus/deficit for type $t\in T$. Note that we set the demand type of all of the equipment demands to be dry ($dc$). We then append the equipment demands to $\Theta'$ as follows: $\Theta'\leftarrow\Theta'\cup\bigcup_{t\in T}\Theta_t^{E'}$. In addition, let $a^{(o,d,dc)}=\sum_{s\in S}u_s^{dc}$ and $r^{(o,d,dc)}=r_{dc}^{Eqp}$ for all $t\in T,(o,d,dc)\in\Theta_t^{E'}$. Thus, the maximum amount of equipment available for each equipment

demand is equal to the sum of the capacities of all ships, and the revenue per TEU delivered is equal to the equipment revenue for each equipment type. Our model uses the same parameters as the arc flow model in Section 3 and the equipment as flow model in Section 4.2. The majority of the model is the same as the previous two models, however we include all of the objectives and constraints for completeness.

**Objective and Constraints.** We use the variables $y^s_{i,j}$ and $z_s^{(o,d,t)}$ from the equipment as flows model and require no additional variables. The model is as follows:

$$\max -(1) + (13) \tag{20}$$

subject to constraints (4), (5), (6), (7), from the arc flow model in Section 3.2, and (14), (15), (17), and (19) from the equipment as flows model in Section 4.2. Instead of the dry capacity constraint in the equipment as flows model, we impose the following constraint:

$$\sum_{(o,d,t) \in \Theta_i^{Vis'}} z_s^{(o,d,t)} \leq u_s^{dc} \qquad \forall s \in S, i \in V'. \tag{21}$$

The objective, (20), combines the sailing costs and port fees from the arc flow model with the cargo demand objective from the equipment as demands model. Note the lack of any objective relating to equipment, as it is now a part of the demand structure.

As in the equipment as flows model, we include several constraints from the arc flow model to enforce node disjointness along vessel paths and control the vessel flows. However, we omit the equipment flow balance constraints (12). We also include the node demand constraints from the equipment as flows model, along with the reefer capacity constraint, as they are unaffected by modeling equipment as demands. We modify the dry capacity constraints (16) to produce constraints (21), in which the sum of the demands carried at a particular node must respect the vessel capacity.

## 5 Computational Evaluation

We evaluated our node flow model, with both versions of equipment handling, on the 37 confidential and 37 public LSFRP instances where only inflexible visitations are present [16]. The instances include two real-world repositioning scenarios as well as a number of crafted scenarios based on data from Maersk Line. Table 1 shows the results of running the arc flow model (AF), equipment as flows (EAF), and equipment as demands (EAD) models on the confidential instance set[3] with a timeout of 5 hours of CPU time and a maximum of 10 GB of memory. We used CPLEX 12.4 on AMD Opteron 2425 HE processors. The table's columns describe the following instance information. Each instance has

[3] The missing instance IDs correspond to instances with flexible arcs, which the EAD and EAF models do not solve.

**Table 1.** Time required to solve the arc flow (AF), equipment as flows (EAF), and equipment as demands (EAD) models to optimality in CPLEX 12.4, along with instance statistics for all of the confidential instances with no flexible arcs from [16]

| ID | $\|S\|$ | $\|V\|$ | $\|A\|$ | $\|\Theta\|$ | $\|E\|$ | $\|SOS\|$ | AF | EAF | EAD |
|---|---|---|---|---|---|---|---|---|---|
| repo1c | 3 | 30 | 94 | 27 | 0 | 1 | 0.05 | 0.48 | 0.04 |
| repo2c | 3 | 30 | 94 | 27 | 0 | 2 | 0.04 | 0.04 | 0.04 |
| repo3c | 3 | 37 | 113 | 25 | 0 | 2 | 0.03 | 0.03 | 0.03 |
| repo4c | 3 | 40 | 143 | 21 | 0 | 3 | 0.03 | 0.37 | 0.03 |
| repo5c | 3 | 47 | 208 | 24 | 0 | 3 | 0.05 | 0.02 | 0.03 |
| repo6c | 3 | 47 | 208 | 24 | 0 | 3 | 0.05 | 0.04 | 0.03 |
| repo7c | 3 | 53 | 146 | 51 | 0 | 4 | 0.07 | 0.04 | 0.04 |
| repo10c | 4 | 58 | 389 | 150 | 0 | 0 | 15.98 | 0.34 | 0.33 |
| repo12c | 4 | 74 | 469 | 174 | 0 | 2 | 93.65 | 0.88 | 0.86 |
| repo13c | 4 | 80 | 492 | 186 | 0 | 4 | 175.32 | 0.87 | 0.82 |
| repo14c | 4 | 80 | 492 | 186 | 24 | 4 | 127.48 | 0.93 | 1.05 |
| repo15c | 5 | 68 | 237 | 214 | 0 | 0 | 0.37 | 0.23 | 0.23 |
| repo16c | 5 | 103 | 296 | 396 | 0 | 5 | 0.9 | 0.36 | 0.35 |
| repo17c | 6 | 100 | 955 | 85 | 0 | 0 | 5.09 | 0.73 | 0.71 |
| repo18c | 6 | 133 | 1138 | 101 | 0 | 9 | 6.77 | 0.79 | 0.75 |
| repo19c | 6 | 133 | 1138 | 101 | 33 | 9 | 7.25 | 0.87 | 0.98 |
| repo20c | 6 | 140 | 1558 | 97 | 0 | 4 | 262.21 | 1.45 | 1.41 |
| repo21c | 6 | 140 | 1558 | 97 | 13 | 4 | 53.95 | 1.62 | 1.55 |
| repo22c | 6 | 140 | 1558 | 97 | 37 | 4 | 94.46 | 1.19 | 1.65 |
| repo24c | 7 | 75 | 395 | 196 | 0 | 3 | 2.44 | 0.35 | 0.32 |
| repo25c | 7 | 77 | 406 | 195 | 0 | 0 | 2.64 | 0.32 | 0.30 |
| repo26c | 7 | 77 | 406 | 195 | 16 | 0 | 1.95 | 0.41 | 0.34 |
| repo27c | 7 | 78 | 502 | 237 | 0 | 0 | 97.12 | 0.53 | 0.51 |
| repo28c | 7 | 89 | 537 | 241 | 0 | 4 | 174.44 | 0.66 | 0.55 |
| repo29c | 7 | 89 | 537 | 241 | 19 | 4 | 126.97 | 0.62 | 0.59 |
| repo30c | 8 | 126 | 1154 | 165 | 0 | 0 | 2058.45 | 3.76 | 4.15 |
| repo31c | 8 | 126 | 1300 | 312 | 0 | 0 | 105.49 | 4.93 | 4.76 |
| repo32c | 8 | 140 | 1262 | 188 | 0 | 3 | 494.39 | 6.97 | 7.12 |
| repo34c | 9 | 304 | 9863 | 435 | 0 | 0 | CPU | 2256.99 | 2532.11 |
| repo36c | 9 | 364 | 11078 | 1280 | 0 | 4 | Mem | CPU | 16203.26 |
| repo37c | 9 | 371 | 10416 | 1270 | 114 | 7 | Mem | 7033.32 | 6330.48 |
| repo39c | 9 | 379 | 10660 | 1371 | 0 | 7 | Mem | 7125.89 | 7142.55 |
| repo40c | 9 | 379 | 10660 | 1371 | 118 | 7 | Mem | 15857.61 | 10049.79 |
| repo41c | 10 | 249 | 7654 | 473 | 0 | 0 | CPU | 2543.09 | 3954.17 |
| repo42c | 11 | 275 | 5562 | 1748 | 0 | 5 | CPU | CPU | CPU |
| repo43c | 11 | 320 | 11391 | 1285 | 0 | 0 | Mem | CPU | CPU |
| repo44c | 11 | 328 | 11853 | 1403 | 0 | 4 | Mem | CPU | CPU |

**Table 2.** Time required to solve the arc flow (AF), equipment as flows (EAF), and equipment as demands (EAD) models to optimality in CPLEX 12.4, along with instance statistics for all of the public instances with no flexible arcs from [16]

| ID | $\|S\|$ | $\|V\|$ | $\|A\|$ | $\|\Theta\|$ | $\|E\|$ | $\|SOS\|$ | AF | EAF | EAD |
|---|---|---|---|---|---|---|---|---|---|
| repo1p | 3 | 36 | 150 | 28 | 0 | 1 | 0.06 | 0.08 | 0.06 |
| repo2p | 3 | 36 | 150 | 28 | 0 | 2 | 0.06 | 0.05 | 0.05 |
| repo3p | 3 | 38 | 151 | 24 | 0 | 2 | 0.04 | 0.04 | 0.04 |
| repo4p | 3 | 42 | 185 | 20 | 0 | 3 | 0.05 | 0.23 | 0.03 |
| repo5p | 3 | 51 | 270 | 22 | 0 | 3 | 0.07 | 0.05 | 0.05 |
| repo6p | 3 | 51 | 270 | 22 | 0 | 3 | 0.07 | 0.06 | 0.05 |
| repo7p | 3 | 54 | 196 | 46 | 0 | 4 | 0.08 | 0.05 | 0.05 |
| repo10p | 4 | 58 | 499 | 125 | 0 | 0 | 74.87 | 0.27 | 0.26 |
| repo12p | 4 | 74 | 603 | 145 | 0 | 2 | 132.8 | 0.31 | 0.28 |
| repo13p | 4 | 80 | 632 | 155 | 0 | 4 | 101 | 0.33 | 0.31 |
| repo14p | 4 | 80 | 632 | 155 | 24 | 4 | 171.79 | 0.35 | 0.33 |
| repo15p | 5 | 71 | 355 | 173 | 0 | 0 | 0.47 | 0.31 | 0.29 |
| repo16p | 5 | 106 | 420 | 320 | 0 | 5 | 1.09 | 0.39 | 0.38 |
| repo17p | 6 | 102 | 1198 | 75 | 0 | 0 | 4.59 | 1.01 | 0.94 |
| repo18p | 6 | 135 | 1439 | 87 | 0 | 9 | 6.52 | 0.95 | 0.82 |
| repo19p | 6 | 135 | 1439 | 87 | 33 | 9 | 5.67 | 1.03 | 1.08 |
| repo20p | 6 | 142 | 1865 | 80 | 0 | 4 | 13.68 | 1.44 | 1.15 |
| repo21p | 6 | 142 | 1865 | 80 | 13 | 4 | 16.24 | 2.06 | 1.36 |
| repo22p | 6 | 142 | 1865 | 80 | 37 | 4 | 19.13 | 1.69 | 1.44 |
| repo24p | 7 | 75 | 482 | 154 | 0 | 3 | 1.65 | 0.37 | 0.34 |
| repo25p | 7 | 77 | 496 | 156 | 0 | 0 | 3.95 | 0.41 | 0.38 |
| repo26p | 7 | 77 | 496 | 156 | 16 | 0 | 3.74 | 0.47 | 0.40 |
| repo27p | 7 | 79 | 571 | 188 | 0 | 0 | 1265.98 | 0.57 | 0.50 |
| repo28p | 7 | 90 | 618 | 189 | 0 | 4 | 1319.39 | 0.55 | 0.46 |
| repo29p | 7 | 90 | 618 | 189 | 19 | 4 | 1039.57 | 0.57 | 0.54 |
| repo30p | 8 | 126 | 1450 | 265 | 0 | 0 | 286.7 | 13.10 | 12.60 |
| repo31p | 8 | 130 | 1362 | 152 | 0 | 0 | 23.38 | 30.88 | 28.89 |
| repo32p | 8 | 144 | 1501 | 170 | 0 | 3 | 50.95 | 45.99 | 41.46 |
| repo34p | 9 | 304 | 10577 | 344 | 0 | 0 | CPU | 7652.67 | 7388.75 |
| repo36p | 9 | 364 | 11972 | 1048 | 0 | 4 | Mem | CPU | CPU |
| repo37p | 9 | 371 | 11371 | 1023 | 114 | 7 | Mem | 1408.75 | 790.74 |
| repo39p | 9 | 379 | 11666 | 1109 | 0 | 7 | Mem | 1701.53 | 1911.40 |
| repo40p | 9 | 379 | 11666 | 1109 | 118 | 7 | Mem | 3178.03 | 1859.09 |
| repo41p | 10 | 249 | 8051 | 375 | 0 | 0 | CPU | 659.75 | 727.78 |
| repo42p | 11 | 279 | 6596 | 1423 | 0 | 5 | CPU | 4930.85 | 4006.27 |
| repo43p | 11 | 320 | 13058 | 1013 | 0 | 0 | Mem | CPU | CPU |
| repo44p | 11 | 328 | 13705 | 1108 | 0 | 4 | Mem | CPU | CPU |

$|S|$ ships, $|V|$ is the number of visitations, $|A|$ is the number of arcs, $|\Theta|$ is the number of demands, $|E| = |\bigcup_{t \in T} V^{t^*}|$ is the number of visitations with either an equipment surplus or deficit, and $|SOS|$ is the number of SOS structures in the graph. The EAD model is able to solve all but three of the confidential IVLSFRP instances within 5 hours of CPU time, while the EAF model solves 3 previously unsolved instances. Additionally, for instances with between one and eight ships, the CPU time is significantly improved, with an average decrease in CPU time of 138 seconds for both EAF and EAD over AF. The largest speed improvement is on instance repo30c, where EAF and EAD are roughly 500 times faster than AF.

Even on the instances where EAD or EAF timeout, CPLEX is able to provide feasible solutions, unlike in the case of the AF model, where no feasible solution is found for any instance that exceeds the memory allotment or CPU timeout. The EAD model is able to find a solution with a 10.16% gap (to the LP relaxation) for repo42c, a 96.41% gap on repo43c, and an 88.96% gap on repo44c. Although EAD requires over an hour to solve several instances, it finds the optimal solution within an hour on both repo37c and repo40c, but requires extra time to prove optimality. On repo39c and repo40c, EAD is able to find an optimality gap of 8.31% and 10.07% within an hour, respectively.

Table 2 gives our results for the public instances in the IVLSFRP dataset. The columns of the table are identical to Table 1. As in the case of the confidential instances, the node flow model achieves significant performance gains, both in the EAF and EAD cases. With the EAD model, the node flow model is able to provide an optimal answer to every instance in the datset. The largest time improvements are on instances repo27p, repo28p and repo29p, where both the EAD and EAF models allow CPLEX to find the optimal solution in under a second, but the AF model requires over 15 minutes.

## 6 Conclusion

We introduced a novel, node flow based mathematical model for the IVLSFRP, a fundamental problem to the operations of the world's shipping lines. We provided two versions of the node flow model, in which we modeled equipment flows using an arc flow approach, as well as by converting the equipment flows into demands. Both versions of the model showed significant speedups over the arc flow model, and were able to solve IVLSFRP instances to optimality in CPLEX that the arc flow model cannot even load into memory. In addition, our node flow model offers multiple orders of magnitude improvements on a number of instances in the LSFRP dataset. For future work, we will try to apply the ideas from the node flow model to the full LSFRP.

**Acknowledgements.** We would like to thank our industrial collaborators Mikkel Muhldorff Sigurd and Shaun Long at Maersk Line for their support and detailed description of the fleet repositioning problem. This research is sponsored in part by the Danish Council for Strategic Research as part of the ENERPLAN research project.

## References


1. Álvarez, J.F.: Joint routing and deployment of a fleet of container vessels. Maritime Economics and Logistics 11(2), 186–208 (2009)
2. Andersen, M.W.: Service Network Design and Management in Liner Container Shipping Applications. PhD thesis, Technical University of Denmark, Department of Transport (2010)
3. Brouer, B.D., Dirksen, J., Pisinger, D., Plum, C.E.M., Vaaben, B.: The Vessel Schedule Recovery Problem (VSRP) – A MIP model for handling disruptions in liner shipping. European Journal of Operational Research 224(2), 362–374 (2013)
4. Burke, E.K., Bykov, Y.: A late acceptance strategy in hill-climbing for exam timetabling problems. In: PATAT 2008 Conference, Montreal, Canada (2008)
5. Christiansen, M.: Decomposition of a combined inventory and time constrained ship routing problem. Transportation Science 33(1), 3–16 (1999)
6. Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D.: Maritime transportation. In: Handbooks in Operations Research and Management Science, vol. 14, pp. 189–284 (2007)
7. Christiansen, M., Fagerholt, K., Ronen, D.: Ship routing and scheduling: Status and perspectives. Transportation Science 38(1), 1–18 (2004)
8. Clausen, J., Larsen, A., Larsen, J., Rezanova, N.J.: Disruption management in the airline industry–concepts, models and methods. Computers & Operations Research 37(5), 809–821 (2010)
9. Kohl, N., Larsen, A., Larsen, J., Ross, A., Tiourine, S.: Airline disruption management–perspectives, experiences and outlook. Journal of Air Transport Management 13(3), 149–162 (2007)
10. Korsvik, J.E., Fagerholt, K., Laporte, G.: A large neighbourhood search heuristic for ship routing and scheduling with split loads. Computers & Operations Research 38(2), 474–483 (2011)
11. Løfstedt, B., Alvarez, J.F., Plum, C.E.M., Pisinger, D., Sigurd, M.M.: An integer programming model and benchmark suite for liner shipping network design. Technical Report 19, DTU Management (2010)
12. Meyer, J., Stahlbock, R., Voß, S.: Slow steaming in container shipping. In: Proceedings of the 45th Hawaii International Conference on System Science (HICSS), pp. 1306–1314. IEEE (2012)
13. Powell, B.J., Perakis, A.N.: Fleet deployment optimization for liner shipping: An integer programming model. Maritime Policy and Management 24(2), 183–192 (1997)
14. Stahlbock, R., Voß, S.: Operations research at container terminals: a literature update. OR Spectrum 30(1), 1–52 (2008)
15. Tierney, K.: Late Acceptance Hill Climbing for the Liner Shipping Fleet Repositioning Problem. In: Proceedings of the 14th EUME Workshop, pp. 21–27 (2013)
16. Tierney, K., Áskelsdóttir, B., Jensen, R.M., Pisinger, D.: Solving the liner shipping fleet repositioning problem with cargo flows. Technical Report TR-2013-165, IT University of Copenhagen (January 2013)
17. Tierney, K., Coles, A.J., Coles, A.I., Kroer, C., Britt, A.M., Jensen, R.M.: Automated planning for liner shipping fleet repositioning. In: McCluskey, L., Williams, B., Silva, J.R., Bonet, B. (eds.) Proceedings of the 22nd International Conference on Automated Planning and Scheduling, pp. 279–287 (2012)
18. Tierney, K., Jensen, R.M.: The Liner Shipping Fleet Repositioning Problem with Cargo Flows. In: Hu, H., Shi, X., Stahlbock, R., Voß, S. (eds.) ICCL 2012. LNCS, vol. 7555, pp. 1–16. Springer, Heidelberg (2012)

# An LNS Approach for Container Stowage Multi-port Master Planning

Dario Pacino

Deparment of Transport, Technical University of Denmark
darpa@transport.dtu.dk

**Abstract.** The generation of competitive stowage plans have become a priority for the shipping industry. Stowage planning is NP-hard and is a challenging optimization problem in practice. Two-phase decomposition approaches have proved to give viable solutions. We propose a large neighborhood search (LNS) to solve the first of the two phases, the multi-port master planning problem. Our approach combines the strength of mathematical modeling with the flexibility of a local search. We show how the new approach can solve more instances than previous mathematical models, and present an analysis of its performance.

## 1 Introduction

Over the past two decades, the demand for cost efficient containerized transportation has seen a continuous increase. In answer to this demand, shipping companies have deployed bigger vessels, that nowadays can transport over 15,000 containers and are wider than the extended Panama Canal. Like busses, container vessels sail from port to port through a fixed route loading and discharging thousands of containers. Before the vessel arrives at a port, it is the job of stowage coordinators to devise a stowage plan. A stowage plan is a document describing where each container should be loaded in the vessel once terminal operations start. In general, a stowage plan must ensure that the loaded container ship is seaworthy. The vessel must have sufficient *transversal stability* when sailing, its *draft* (immersion depth) and *trim* (longitudinal inclination) must be within limits, and the weight distribution must satisfy the stress limits of the structure (*shear forces* and *bending moment*). The stowage of containers must respect stacking constraints, such as allocating refrigerated containers (*reefers*) near power plugs, disallowing 40' long containers to be on top of 20' long containers, and obeying weight and capacity limits.

When planning, stowage coordinators aim at producing stowage plans that minimize port stay. Major factors impacting the ship's time at port are the crane utilization and the number of overstowages. Overstowage happens when a container destined to a later port is stowed on top of containers destined to an earlier port. Crane utilization depends on the distribution of containers along the vessel and the number of cranes assigned to the ship. Even though stowage plans are still mainly performed manually, they are hard to produce in practice.

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 35–44, 2013.

First, they are made under time pressure just hours before the vessel calls the port. Second, deep-sea vessels are large and often require thousands of container moves in a port. Third, complex interactions between low-level stacking rules and high-level stress limits and stability requirements make it difficult to minimize the makespan of cranes and, at the same time, overstowage. Fourth, containers to be loaded at later ports must be taken into consideration to minimize potential negative impacts of the current stowage plan.

In previous work [13], we presented a hierarchical decomposition approach to stowage planning. The decomposition solves the problem in two phases. First, the *multi-port master planning* phase finds a distribution of container types to sub-sections of the vessel. It is here that high-level stability constraints are satisfied, the crane utilization is maximized, and *hatch-overstowage* (overstowage between on-deck and below-deck containers) is minimized. The master planning solution of the first port, the port we are making the plan for, becomes the input of the second phase, *slot planning*. In the second phase, the container types assigned to a vessel sub-section are assigned a specific positions following the stacking rules. This phase minimizes overstowage and includes a number of heuristic rules for improving the robustness of the plan.

The work presented in this paper focuses on the multi-port master planning problem. In [13] an IP model for these problems was presented, which was solved heuristically by relaxing the integrality constraints of some of the decision variables. The contribution of this paper is twofold. First we propose a different heuristic approach, Large Neighborhood Search (LNS), that can find integer solutions to the multi-port master planning problem within the 10 minutes time limit proposed in [13]. Second, we propose a simple warm start procedure that allows to find feasible solutions for all instances. This research is motivated by the results published in [13] where it is possible to see that integer solutions to the multi-port master planning problem can lead to better stowage plans.

The remainder of this paper is organized as follows. Section 2 presents the multi-port master planning problem, followed by Section 3 with a review of related works. Section 4 describes the solution approach, which is then evaluated in Section 5. Conclusions and proposals for future work are presented in Section 6.

## 2 The Multi-port Master Planning Problem

The most common ISO containers are either 20', 40', or 45' long. Other special types exist, such as *High-cube* containers (higher than a standard container), *pallet-wide* containers (slightly wider), *reefer* containers (refrigerating units that must be placed near power plugs), and *IMO* containers for dangerous cargo.

The capacity of a container ship is given in Twenty-foot Equivalent Units (TEU). As shown in Figure 1, the cargo space of a vessel is divided into sections called *bays*, and each bay is divided into an *on-deck* and a *below-deck* part by a number of *hatch-covers*, which are flat, leak-proof structures. Each bay consists of a row of *container stacks* divided into *slots* that can hold a 20' ISO container. Figure 2 (a) and (b) show the container slots of a bay and stack, respectively.

Below deck, cell guides secure containers transversely. Containers on deck are secured by lashing rods and twist locks with limited strength. Thus, container weights must normally decrease upwards in stacks on deck. Moreover, stack heights must be under the vessel's minimum line of sight.

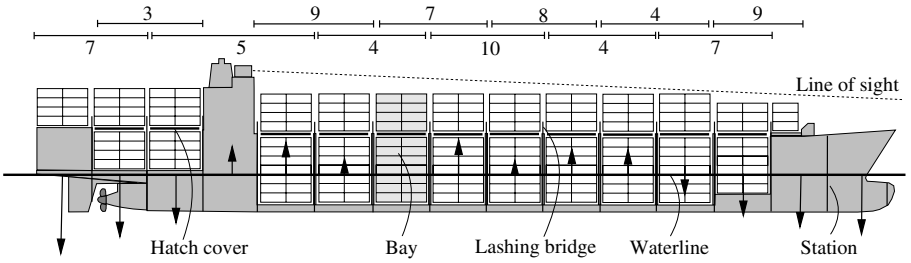



**Fig. 1.** Arrangement of container vessel bays. The vertical arrows show an example of the resulting forces acting on the ship sections between calculation points (stations). Single crane work hours for adjacent bays are shown at the top.

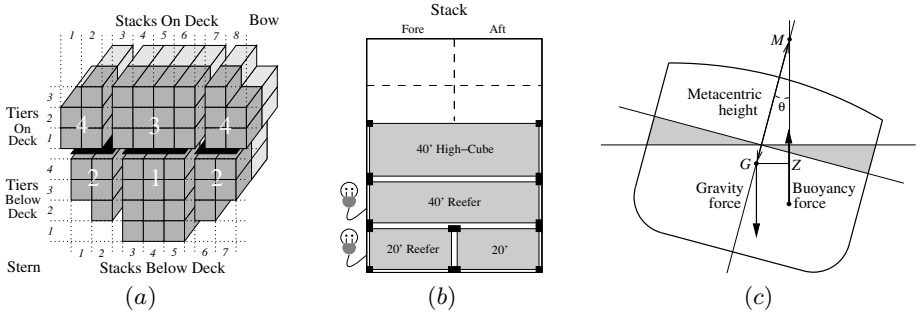



**Fig. 2.** (a) A bay seen from behind. (b) A side view of a stack of containers. Power plugs are usually situated at bottom slots. (c) Transverse stability.

A container ship must have sufficient *transverse stability*. Figure 2 (c) shows a cross section of a ship. For small inclination angles, the buoyancy force intersects the center line in a fixed position called the *metacenter*, $M$ [15]. The *metacentric height* is the distance between $M$ and $G$ (the center of gravity). The ship must have sufficient $GM$ to be stable. Maximum and minimum draft restrictions apply due to port depths, working height of cranes, and the propeller. The *trim* is the difference between the aft and fore draft and must be kept within a given span. For a station position $p$, the *shear force* is the sum of the resulting vertical forces on vessel sections (see Figure 1) acting aft of $p$, and the *bending moment* is the sum of these forces times the horizontal distance to them from $p$. Both of these stresses must be within limits.

A *stowage plan* assigns the containers to load in a terminal to slots on the vessel. In contrast, a *master plan* is the assignment of containers, grouped into

types, to sub-sections of bays (*locations*). A master plan is the intermediary result of the hierarchical decomposition previously introduced, but it can also be seen as a strategic plan that can be used to evaluate different loading scenarios. Given a list of containers to load (*loadlist*) and a forecast of future loads, the multi-port master planning problem is the problem of finding a feasible master plan that minimizes hatch-overstowage and crane makespan (crane utilization).

It is impractical to study large optimization models that include all details of stowage planning. On the other hand, all major aspects of the problem must be modeled for the results to be valuable. For container types this includes 20', 40', and reefer containers. In addition, since stability, trim, draft and stress moment limits should not fully be ignored, some weight classes of containers must be introduced. It is also important to take into consideration the containers already onboard the vessel (the *release*) when arriving at the current port.

## 3 Related Work

The number of publications on stowage planning has grown substantially within the last few years. Contributions can be divided into two main categories: single-phase and multi-phase approaches. Multi-phase approaches decompose the problem hierarchically. They are currently the most successful in terms of model accuracy and scalability. The earliest work can be traced back to a 3-phase heuristic [16], but the first contribution that models several major aspects of the problem is a 2-phase approach that solves a master planning phase for multiple ports with a branch-and-bound algorithm and uses tabu search for bay planning [17]. Other approaches that use a similar decomposition include solving multi-port master planning with an iterative improvement approach based on the transportation simplex method [10] and a bin-packing heuristic [19]. 3-phase approaches include combinations of constructive heuristics, 0/1 IP, and meta-heuristics (e.g., [1]) and heuristics combined with LS (e.g., [18]). To the best of the authors knowledge, the model that covers the most important aspect of stowage planning is that of Pacino et al. [13], which also represents the basis of this research.

Single-phase approaches represent the stowage planning problem (or parts of it) in a monolithic optimization model. Those works do not include a master planning problem, however, for completeness we would like to point the reader to approaches (among others) based on IP models [3,9,11], constraint programming [2,6,12], and heuristics [5,7,8,4].

## 4 Large Neighborhood Search (LNS)

LNS is a search procedure which has been proved effective in solving large-scale combinatorial optimization problems [14]. LNS is based on a destruction/ construction principle. Once an initial solution is found, part of the variable assignments are relaxed (*destruction*), while keeping the remaining variables fixed. A new solution is then found by re-optimizing the assignment of the free variables,

(*construction*). These two steps are then iterated until some termination criterion, in our case a time limit. An outline of the search procedure is presented in Algorithm 1. Details of the initial solution generation (line 2), and the neighborhood selection (line 4), are explained further in Sections 4.1 and 4.2. Notice, that line 7 forces the re-optimization to find only, equal, or improving solutions, and line 8 re-optimizes the model.

**Algorithm 1:** LNS

```
1 M = IP model;
2 π = find initial solution;
3 while there is time do
4     V = select variables from π;
5     forall the variable v ∉ V do
6         fix variable v with the corresponding value from π;
7         post objective of M ≥ objective of π;
8     solve M;
```

This work is based on the IP model presented in [13], which is also used during the re-optimization procedure. We briefly introduce the decision variables used in the model, and point the reader to [13] for a more in-depth description of the model. Let $P$ be the number of ports in the route, $L$ the locations in the vessel and $T$ the supported container types. The decision variable $x^{\tau}_{tl} \in \mathbb{Z}^+$ defines the number of container of type $\tau \in T$ to be loaded in location $l \in L$ during transport $t \in TR$, where $TR = \{\langle o, d\rangle | o, d \in P, o < d\}$ is the set of origin destination pairs defined by the loadlist. The container types include 20', 40' and reefer containers, considering two weight classes (light and heavy). A different average weight for each of the types at each transport is calculated. We model the weight and capacity limits of the locations and the handling of release containers. All stability constraints are modelled including shear forces as an example of stress moments. The model minimizes a weighted sum of cost variables

$$\sum_{p \in P} \sum_{l \in L} \left(\mathcal{C}^O y^O_{pl} + \mathcal{C}^P y^P_{pl}\right) + \sum_{p \in P} \mathcal{C}^T y^T_p \tag{1}$$

where $y^O_{pl}$ is the hatch-overstowage of location $l \in L$ at port $p \in P$, $y^P_{lp}$ is an indication of possible overstowage within location $l \in L$ at port $p \in P$, and $y^T_p$ is the crane makespan at port $p \in P$. The weights $(\mathcal{C}^O, \mathcal{C}^P, \mathcal{C}^T)$ have been derived from the deployed system of our industrial partner and thus reflect a refined adjustment to the economy of stowage planning and the preferences of stowage coordinators.

### 4.1 Initial Solution

Having an initial solution is essential for the use of LNS, however, it is not trivial to devise a heuristic procedure that can guarantee feasible solutions to

the multi-port master planning problem. We propose a simple procedure based on the original IP model:

1. Solve the LP relaxation of the IP model.
2. Let IPf be the IP model without an objective function (e.g. $\min 0$).
3. Solve the IPf using the LP relaxation from 1. as warm-start.

The idea is quite simple, but effective. When solving an IP model, the solution process is often guided by the LP relaxation and the objective function. By removing the objective function we force the solver to focus on feasibility. At the same time, in order to avoid having the solver find bad solutions, we warm start it with the LP solution that includes the objective function. The experimental section will show how this procedure allows the LNS to find feasible solutions for all the tested instances.

### 4.2 Neighborhood

The neighborhood operator in an LNS is the selection of a subset of current assignments to be relaxed. In [14] it was argued that if the relaxed assignments do not interact on the objective function, the re-optimization will probably not find an improving solution. In this work we propose a neighborhood based on the hatch-overstowage objective, where we mix assignments related to the objective with random ones. First we divide the assignments into two categories: assignments to locations with hatch-overstowage, and those without. From each of the groups we select assignments with the same probability $\mathcal{P}$ (changing $\mathcal{P}$ will change the size of the neighborhood). Assignments resulting in hatch-overstowage use, however, an adjusted probability proportional to the amount of hatch-overstwage. The probability adjustment thus changes from location to location and it is equal to $\frac{\hat{y}_l^{O+}}{\hat{Y}^O}$, where $\hat{y}_l^{O+}$ is the maximum hatch-overstowage of location $l \in L$ between all ports, and $\hat{Y}^O$ is the maximum hatch-overstowage between all locations. This probability function ensures that assignments to locations with higher hatch-overstowage are more likely to be selected. The selection probability $\mathcal{P}$ has been experimentally selected to be 0.2.

## 5 Computational Experiments

To evaluate our approach, we have used 80 industrial instances. The instances are based on real stowage problems that coordinators have solved manually, and thus have very high quality data. Characteristics of the data are summarized in Table 1. All experiments were run on an AMD Opteron processor at 2.8 Ghz with 128 GB of memory. Results for the LNS are reported as average over 5 runs of the algorithm. In order to evaluate the LNS approach we compare its results with the integer programming model (IP). We also provide results of the IP model warm started with the initial solution provided with the procedure in Section 4.1 (identified by the acronym IPWS). The first thing to notice, while

**Table 1.** *Instance characteristics.* The first row represents a size categorization based on the size of the vessel. For each size category, the successive rows present respectively the capacity of the vessel, the range of locations, the number of instances, and the average percentage of 20', 40' and release containers. The last row gives a summary of the ports in the routes and vessel utilization percentages.

| | Small | Medium | Large |
|---|---|---|---|
| Capacity (TEU) | 2000-5000 | 5000-8000 | 8000-10,000 |
| Locations | 40-74 | 65-90 | 80-100 |
| Instances | 20 | 22 | 38 |
| Avg.% 20' | 26 | 30 | 19 |
| Avg.% 40' | 40 | 44 | 46 |
| Avg.% Release | 34 | 26 | 35 |
| Ports: 3-21 | Utilization: 8-88% | | |

comparing the IP and the LNS approach, is that the IP model is only able to find feasible instances for 50 out of 80 instances, of which only 6 reached optimality within 3600 seconds. In contrast, the LNS is able to find feasible solutions to all instances. Since the LNS uses the warm starting procedure to find an initial solution, it is a logical consequence that the IPWS model also finds feasible solutions to all instances. In order to have a better overview of the algorithms performance, Figure 3 shows the objective value (y axis) of for each instance (x axis) found by the three approaches within 600 seconds (the time-limit given by the industry). The figure shows how the LNS and IPWS solve more instances than the IP model and how the LNS, in most of the instances, outperforms the IPWS. Instances for which a better solution is found by the IPWS, can be solved to optimality in few seconds.

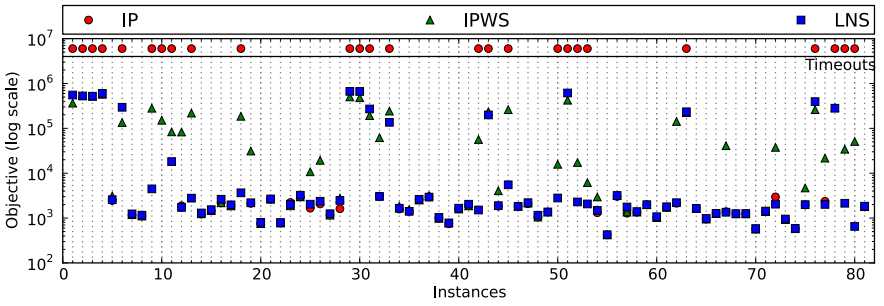



**Fig. 3.** Approach comparison. Each vertical dotted line represents an instance, and the position in the line is the objective value. The lower the value the better the approach. Timeouts are shown at the top.

Even if the warm starting procedure allows the IPWS model to find an initial solution to all the instances, in only 69 of the 80 instance the solution is further improved, and only 9 are solved to optimality. Figure 4 compares the performance of the LNS and the IPWS in those 69 instances (after an initial solution has been found). The graph shows how the average objective value decreases in time. Figure 4 (a) shows how the LNS outperforms the IPWS within the 600 seconds time-limit. It is worth noticing that most of the optimization is done within 100 seconds. This suggests that the neighborhood used converges fast towards a local minima from which it cannot escape. This result is confirmed in Figure 4 (b) where it is possible to see that the LNS is outperformed by the IPWS after ca. 1600 seconds.

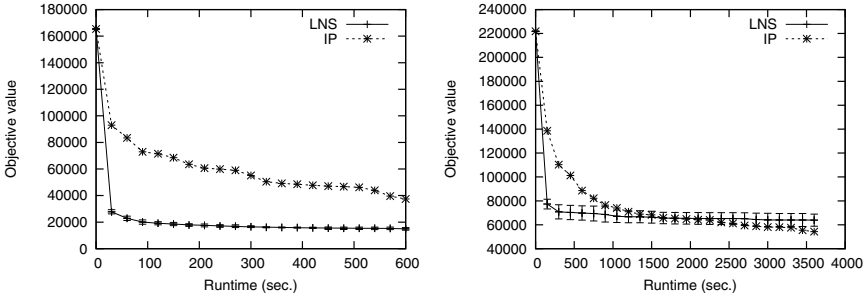



**Fig. 4.** Convergency of the algorithms in (a) 600 sec. (b) 3600 sec. Data is collected only for the instances where the IPWS improves the initial solution.

Figure 5 shows how the warm-starting procedure leads to better solutions. We compare the LNS that uses the warm-starting procedure, with a version of the LNS where the initial solution is given by solving[M U0 the IP model without the objective function.

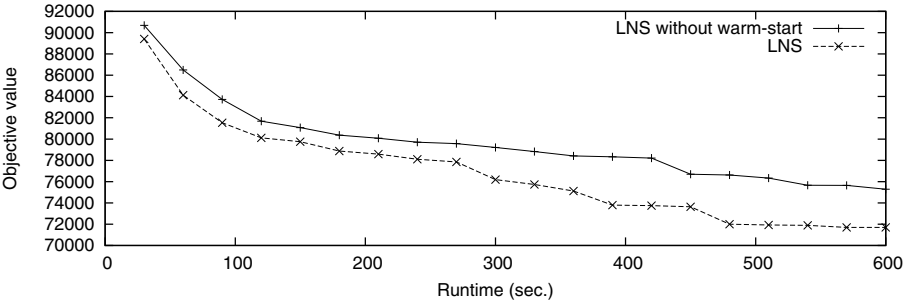



**Fig. 5.** Impact of the warm-start procedure. Comparing the LNS with a modified version that does not use the warm-start procedure.

## 6 Conclusions

In this paper we have presented an LNS approach for the container stowage multi-port master planning problem. The new heuristic approach uses a warm-starting procedure that can also be applied to the IP model presented in [13]. Our experimental results show that the LNS can be used as an anytime algorithm and that the warm-starting procedure improves the general results of the original IP model. In future work we plan to improve the warm-starting procedure by using a faster heuristic approach, and at the same time increase the number of neighborhood operators of the LNS. Using more neighborhood operators should improve the probability of escaping the local minima to which the LNS currently converges to.

## References

1. Ambrosino, D., Anghinolfi, D., Paolucci, M., Sciomachen, A.: An experimental comparison of different heuristics for the master bay plan problem. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 314–325. Springer, Heidelberg (2010)
2. Ambrosino, D., Sciomachen, A.: A constraint satisfaction approach for master bay plans. In: International Conference on Maritime Engineering and Ports, vol. 5, pp. 175–184 (1998)
3. Ambrosino, D., Sciomachen, A.: Impact of yard organization on the master bay planning problem. Maritime Economics and Logistics 5, 285–300 (2003)
4. Avriel, M., Penn, M., Shpirer, N., Witteboon, S.: Stowage planning for container ships to reduce the number of shifts. Annals of Oper. Research 76, 55–71 (1998)
5. Davidor, Y., Avihail, M.: A method for determining a vessel stowage plan, Patent Publication WO9735266 (1996)
6. Delgado, A., Jensen, R.M., Schulte, C.: Generating optimal stowage plans for container vessel bays. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 6–20. Springer, Heidelberg (2009)
7. Dubrovsky, O., Penn, G.L.M.: A genetic algorithm with a compact solution encoding for the container ship stowage problem. J. of Heuristics 8, 585–599 (2002)
8. Flor, M.: Heuristic Algorithms for Solving the Container Ship Stowage Problem. Master's thesis, Technion, Haifa, Isreal (1998)
9. Giemesch, P., Jellinghaus, A.: Optimization models for the containership stowage problem. In: Proceedings of the Int. Conference of the German Operations Research Society (2003)
10. Kang, J., Kim, Y.: Stowage planning in maritime container transportation. Journal of the Operational Research Society 53(4), 415–426 (2002)
11. Li, F., Tian, C.H., Cao, R., Ding, W.: An integer programming for container stowage problem. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part I. LNCS, vol. 5101, pp. 853–862. Springer, Heidelberg (2008)
12. Pacino, D., Jensen, R.M.: A local search extended placement heuristic for stowing under deck bays of container vessels. In: The 4th Int. Workshop on Freight Transportation and Logistics, ODYSSEUS 2009 (2009)
13. Pacino, D., Delgado, A., Jensen, R., Bebbington, T.: Fast generation of near-optimal plans for eco-efficient stowage of large container vessels. In: Böse, J.W., Hu, H., Jahn, C., Shi, X., Stahlbock, R., Voß, S. (eds.) ICCL 2011. LNCS, vol. 6971, pp. 286–301. Springer, Heidelberg (2011)

14. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M.J., Puget, J.-F. (eds.) CP 1998. LNCS, vol. 1520, pp. 417–431. Springer, Heidelberg (1998)
15. Tupper, E.C.: Introdution to Naval Architecture. Elsevier (2009)
16. Webster, W.C., Van Dyke, P.: Container loading. A container allocation model: I - introduction background, II - strategy, conclusions. In: Proceedings of Computer-Aided Ship Design Engineering Summer Conference. University of Michigan (1970)
17. Wilson, I.D., Roach, P.: Principles of combinatorial optimization applied to container-ship stowage planning. Journal of Heuristics (5), 403–418 (1999)
18. Yoke, M., Low, H., Xiao, X., Liu, F., Huang, S.Y., Hsu, W.J., Li, Z.: An automated stowage planning system for large containerships. In: Proceedings of the 4th Virtual Int. Conference on Intelligent Production Machines and Systems (2009)
19. Zhang, W.Y., Lin, Y., Ji, Z.S.: Model and algorithm for container ship stowage planning based on bin-packing problem. Journal of Marine Science and Application 4(3) (2005)

# How Will the Marine Emissions Trading Scheme Influence the Profit and $CO_2$ Emissions of a Containership

Zehui Huang[1], Xiaoning Shi[2], Jingfei Wu[3], Hao Hu[1], and Jinsong Zhao[4]

[1] Shanghai Jiao Tong University, Shanghai, China
{sjtuhzh0205,hhu}@sjtu.edu.cn
[2] Shanghai Jiao Tong University, Shanghai, China & University of Hamburg, Germany
sxn@sjtu.edu.cn, shi@econ.uni-hamburg.de
[3] Shanghai University, Shanghai, China
afeiwu@126.com
[4] East China University of Political Science and Law, Shanghai, China
sgsdmu@163.com

**Abstract.** Greenhouse Gas (GHG) emissions from international shipping has become an increasing concern to the whole world. Although the shipping industry has not been included in the mandatory GHG emissions reduction list in the *Kyoto Protocol*, it should do something to cut down its emissions. This paper proposes a marine emissions trading scheme which mainly concerns $CO_2$ and studies its influence on the annual profit and $CO_2$ emissions of containerships based on speed reduction. Through a case study, we demonstrate that speed reduction is really an effective measure in cutting down ship GHG emission. Meanwhile, the auction rate of allowances has a great impact on the annual profit of a ship. Furthermore, the conducted scenario and sensitivity analysis confirm that in different shipping market situations, the ship always has a most profitable speed, no matter what the auction rate is.

**Keywords:** Marine Emissions Trading Scheme, Auction Rate, Speed Reduction, Profitability.

## 1 Introduction

Global climate change has already had observable effects on the environment. Early researchers generally suggested that the atmosphere had been warmed over the past century, and human activities were likely to be the most important driving factor to that change. Global emissions of $CO_2$ from international shipping are substantial and have a long history dating back to the industrial revolution [1]. It is estimated that in 2007 international shipping have emitted 870 million tons, or about 2.7% of the global emissions of $CO_2$ [2]. Currently, emissions from shipping are around 1000 million tons annually, nearly 130 million tons more comparing with 2007, and in the absence of action they are expected to be more than double by 2050 [3].

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 45–57, 2013.

### 1.1 Development of Greenhouse Gas (GHG) Emissions Reduction Programs for International Shipping

As GHG emissions call more attention from international society, the programs to cut down international shipping GHG emissions from the International Marine Organization (IMO) became a hot topic. In 1977, state parties of the International Convention for the Prevention of Pollution from Ships (MARPOL) adopted a new protocol on ship GHG emissions, which entitled MEPC to examine the feasibility of GHG emissions reduction proposal.

In the 42nd session of the MEPC in 1998, a project steering committee was founded to guide the research on ship GHG emissions. At the same time, this session entitled the IMO to make policies on shipping emissions reduction.

In the 59th session of the MEPC in July 2009, the IMO adopted "EEDI (Energy Efficiency Design Index) Calculation Provisional Guidance," "EEDI Voluntary Certification Interim Guidance," "Ship of Energy Efficiency Management Plan (SEEMP)" and "Guidelines to EEOI (Energy Efficiency Operational Indicator) Voluntary Use." Mandatory measures to reduce emissions of GHGs from international shipping were adopted by Parties to MARPOL Annex VI represented in the 62nd session of the MEPC.

In the 64th session of the MEPC in October 2012, the IMO improved further the draft guidance on treatment of innovative energy efficiency technologies and reviewed the interim guidelines for the calculation of the coefficient *fw* for decrease of ship speed in representative sea conditions for trial use.

### 1.2 Literature Review

Generally, there are three ways to reduce shipping emissions, i.e., technical, operational and market-based measures.

From the technical perspective, the options for ship operators to reduce fuel consumption of vessels can be broadly divided into five categories: engine and transmission technology, auxiliary power systems, propulsion systems, superstructure aerodynamics and hull shape [4]. In 2000, study of GHGs emissions from ships analyzed model tests from MARINTEK's database, which indicated that if the behavior of the hull in still water is optimized, there is a potential for energy savings in the range of 5-20% [5]. With respect to modernizing engines, using power turbines to recover energy from exhaust, or some other measures, such as energy efficiency in the power generation system, could be increased [6, 7]. Changing energy type could also be an efficient option to improve engine efficiency. Corbett and Winebrake [8] suggested that bunker oil refining efficiency influences $CO_2$ emissions significantly. Non-renewable energy (such as LNG, LPG, and nuclear power [9]) and renewable energy (such as wind power, solar energy, and biofuels) have been used to substitute the traditional bunker fuel in recent years. Furthermore, renewable sources could meet a greater share of the rapidly growing international shipping energy demand in the long term, while planning for 100% renewable energy for transport plays an import part in combination with equally important technologies and proposals [10] .

From the perspective of operational measures, policy makers, such as MEPC have designed EEOI and SEEMP to guide energy efficiency increase of vessels through

optimizing ship operations. International ship operators also try to exploit potential operational improvements to achieve emissions reduction, e.g. fleet composition and speed reduction [2]. Psaraftis et al. [11] demonstrated that speed reduction is an effective emissions reduction measure for fast ships. Corbett et al. [12] highlighted that a speed reduction mandate targeted to achieve 20% $CO_2$ reduction in the container fleet costs between \$20 and \$200 per ton $CO_2$, depending on how the fleet responds to a speed reduction mandate. Based on the research of Corbett et al. [12], Crist [4] calculated the impact of speed reduction on 8500 TEU container vessels. The analysis of speed reductions from 10% to 50% showed that when the speed is reduced to a half, fuel consumption decreases by 87%. Furthermore, according to a study of Eide et al. [13] in 2009, cost effective GHG reductions for a fleet may be in the order of 30% for technical measures, and above 50% when including speed reductions.

From a market perspective, there are also some literatures on market-based measures to reduce international shipping $CO_2$ emissions [14-16]. However, there is a lack of research combining market-based measures with operational or technical measures. One important reason is that there have not existed any worldwide market-based measures to limit international shipping GHG by now. In order to propel the development of market-based measures, Miola et al. [17] analyzed limits and opportunities of current market-based measures through a discussion at the international level in 2012. Meanwhile, they estimated the cost-effectiveness of the main technical GHG reduction measures in order to give a magnitude of such costs within a market-based policies framework [17].

From the literature review above, we can see that research on the joint influence of market-based and the other two measures are not extensive. Even if there are a few, they mainly focused on GHG emissions reduction or cost effectiveness, disregarding the profitability of the ship. Moreover, rules of market-based measures are not decided by ship operators compared with technical and operational measures, thus it is worth to study some potential market-based measures to help ship operators choose the most profitable GHG reduction options in advance. This paper mainly focuses on proposing a potential market-based measure, especially an emissions trading scheme. Combined with slow steaming [18], which is an operational measure, this paper discusses the joint influence of these two measures on the profit and $CO_2$ emissions of a ship.

This article is organized as follows. Since the background of this article is introduced in Section 1, Section 2 briefly reviews emissions trading schemes and proposes a marine emissions trading scheme based on existing research. Section 3 develops a model to evaluate the influence of the marine emissions trading scheme on the annual profit and $CO_2$ emissions of a ship when it responses as speed reduction. Section 4 carries out a case study on a containership. Discussions of the results are given in Section 5. Conclusions and further research are drawn in Section 6.

## 2 Potential Marine Emissions Trading Scheme

Among the existing literature, there are mainly two market-based measures, respectively, emissions taxes and emissions trading [19-21]. Between these two measures, emissions taxes are not considered on a global scale as they have political characteristics. Furthermore, how much emissions taxes should be levied, who will manage the fund, etc., are also issues hindering their implementations.

Generally, in an emissions trading scheme, a central authority sets a cap on the amount of a pollutant that can be emitted. The cap is allocated or sold to involved firms in the form of emissions allowances. Firms can exchange allowances in the market. This allows the system to reduce emissions without significant government intervention. Since an emission trading scheme is more marketable than emissions taxes, this paper mainly tries to propose an international, potential marine emissions trading scheme.

Existing emissions trading schemes are mostly regional, such as the European Union Emissions Trading Scheme (EU ETS) and the New Zealand Emissions Trading Scheme (NZ ETS) [22]. Taking international shipping as a global industry, regional emissions trading schemes cannot take an ideal effect on cutting down its emissions. In order to solve this issue, a global marine emissions trading scheme should be put forward. However, there still exist some issues hindering its implementation. The key one is the allowance allocation method. In theory, the efficiency of an emissions trading scheme is independent on the choice of allocation criteria and the design of the initial allocation method [23, 24]. However, the allocation method determines the financial burden to international ship operators. This will be crucial for the acceptability of the marine emissions trading scheme.

Generally, five basic methods have been proposed to allocate allowances, namely grandfathering, benchmarking, auctioning, baseline, no allocation. Each allocation method gets some support or objections, while auctioning appears to be the most attractive option [14, 15, 24]. However, if international shipping carriers get all the allowances through auctions that would be a huge financial burden to them.

Considering the financial burden to ship operators and the GHG emissions reduction pressure to the international shipping sector, the initial allocation of allowances to the shipping sector could, in principle, be done by a combination of grandfathering and auctioning [14]. A hybrid system of these two allocation measures could provide a starting point for a slow transition from allocation free of charge to an auctioning system. Moreover, the obligation to surrender allowances for emissions growth shows parallels with both allocating free of charge and auctioning. One parallel lies in the fact that involved firms do not have to pay for emissions within a certain quantity for free. Once the free quantity is exceeded, they must auction allowances.

After the initial allocation of allowances has been determined, we take a final decision on the size of the cap and auction rate. In this paper, the equivalent cap for a liable entity in the marine emissions trading scheme could be 80% of its 2012 emissions and the ship itself considered as the liable entity [14].

**Table 1.** Marine emissions trading scheme, in detail

| Liable Entity | Cap | Free Rate | Auction Rate |
|---|---|---|---|
| Ship | 80% of its 2012 emissions | 0.8 | 0.2 |
| | | 0.6 | 0.4 |
| | | 0.4 | 0.6 |
| | | 0.2 | 0.8 |
| | | 0 | 1 |

As the auction rate of allowance is hard to decide, this paper mainly refers to auction rates in EU ETS to propose a scenario analysis, which may consider some

possible auction rates in the future marine emissions trading scheme. Table 1 summarizes the key factors of the marine emissions trading scheme here proposed.

## 3 Profitability of Containerships with Different Speed Reduction Rates under the Marine Emissions Trading Scheme

Although there are several kinds of ship delivering cargos between continents, this article mainly focuses on containerships. Containerships are usually operated on closed routes (also known as cycles, strings or loops) and follow a published schedule of sailings [25]. A route is a specified sequence of calling ports that each containership assigned to that route repeats on each voyage.

This paper assumes that all the parameters used to calculate ship annual revenue, cost and $CO_2$ emissions are constant under the period of this study. Given this assumption, a model is developed to calculate maximum annual profit speed of a containership and its corresponding profit and $CO_2$ emissions.

Definition of each parameter used in the model is summarized as follows.

$i$: the origin port
$j$: the destination port
$k$: an individual containership serving the $ij$ route
$F_{ijk}$: the whole fuel consumption of ship $k$ from $i$ to $j$ (tons)
$MF_k$: main engine maximum daily fuel consumption (tons/day)
$AF_{ks}$: auxiliary engine daily fuel consumption when ship is at sea (tons/day)
$AF_{kp}$: auxiliary engine daily fuel consumption when ship stays in port (tons/day)
$S_{0k}$: the design speed (maximum speed) at sea of containership $k$ (knot)
$S_{ijk}$: the operational speed at sea of containership $k$ from $i$ to $j$ (knot)
$d_{ij}$: the distance from $i$ to $j$ port (nautical miles)
$\pi_{ijk}$: the profit from the origin $i$ to the destination $j$ per year (USD)
$R_{ij}$: the freight rate of per TEU from the origin $i$ to the destination $j$ (USD)
$W_{ijk}$: the number of TEUs per ship in one trip from $i$ to $j$
$C_k$: the fixed cost per day for containership $k$ excepting auxiliary engine fuel cost (USD)
$P_M$: the price of bunker fuel for main engine (USD/ton)
$P_A$: the price of bunker fuel for auxiliary engine (USD/ton)
$D_{ijP}$: total days ship stays in port from $i$ to $j$,
$T_{ij}$: the number of trips made by containership $k$ between $i$ to $j$ per year

Based on these parameters, equation (1) represents an annual profit function for ship $k$ from the origin $i$ to the destination $j$ [12],

$$\pi_{ijk} = \left(R_{ij} * W_{ijk} - C_{ijks} - C_{ijkp}\right) * T_{ij} \tag{1}$$

where $C_{ijks}$ (cost at sea) and $C_{ijkp}$ (cost in port), respectively, are

$$C_{ijks} = \left[C_k + \mathrm{P}_M * MF_k * \left(\frac{S_{ijk}}{S_{0k}}\right)^3 + \mathrm{P}_A * AF_{ks}\right] * \frac{d_{ij}}{24S_{ijk}} \tag{2}$$

$$C_{ijkp} = \left(C_k + \mathrm{P}_A * AF_{kp}\right) * D_{ijP} \tag{3}$$

Our main purpose is to get the maximum annual profit speed, so we define $S_{ijk}$ as the only variable in Equation (2) and recognize marginal profit equals zero. Therefore, the maximum annual profit speed function is:

$$S_{ijk} = \left[\frac{(\mathrm{P}_A * AF_{ks} + C_k) * S_{0k}{}^3}{2 * \mathrm{P}_M * MF_k}\right]^{\frac{1}{3}} \tag{4}$$

Applying Equation (4) to determine the optimal speed, we could investigate how the motivation for profit maximization of shipping carriers influences ship speed under the marine emissions trading scheme proposed above.

### 3.1 $CO_2$ Emissions

International containerships emit several kinds of exhaust gases, such as $CO_X$, $NO_X$ and $SO_X$. Among these types of emissions, the quantity of $CO_2$ is the maximum and it is the one that influences global climate most. Therefore, this paper focuses on calculating the quantity of $CO_2$ from international containerships.

Marine fuels are a petroleum byproduct with carbon segment which converts to $CO_2$ after burning. Normally, fuel consumption is used to calculate the $CO_2$ emissions rate. The fuel consumption of a ship is the sum of fuel used by main and auxiliary engines and it is calculated by Equation (5).

$$F_{ijk} = \left\{\left[MF_k * \left(\frac{S_{ijk}}{S_{0k}}\right)^3 + AF_{ks}\right] * \frac{d_{ij}}{24S_{ijk}} + AF_{kp} * D_{ijP}\right\} \tag{5}$$

The fuel consumption of a ship is then multiplied by the fuel's carbon fraction (defined at 0.8645) and a conversion factor from carbon to $CO_2$ (equal to 44/12) in order to calculate $CO_2$ emissions in tons per trip [12]. The $CO_2$ emissions of each trip from $i$ to $j$ then is:

$$CO_{2_{ij}} = (0.8645) * \left(\frac{44}{12}\right) * F_{ijk} = 3.17 * F_{ijk} \tag{6}$$

Inserting Equation (5) into Equation (6), we obtain:

$$CO_{2_{ij}} = 3.17 * \left\{\left[MF_k * \left(\frac{S_{ijk}}{S_{0k}}\right)^3 + AF_{ks}\right] * \frac{d_{ij}}{24S_{ijk}} + AF_{kp} * D_{ijP}\right\} \tag{7}$$

### 3.2 Annual Profit of a Ship in the Marine Emissions Trading Scheme

As mentioned before, there are three basic measures to cut down exhaust emissions of containerships, respectively, technical, operational and market-based measures. Usually, technical measures need a long time to plan before their final implementation. For this

reason, this paper mainly considers the combined influence of operational and market-based measures on the annual profit of a containership.

There are several types of specific operational measures to mitigate $CO_2$ emissions from containerships. Ship operators tend to prefer speed reduction when there is an emissions trading scheme [12]. Speed reduction is inexorably linked with fuel cost and ship profits. Based on the original maximum profit speed, we propose several ship speed reduction rates and calculate their corresponding annual profits and $CO_2$ emissions. Ship speed keeps stable in each round trip.

Based on Equation (7), we construct a model to calculate the annual emissions and profit of a ship when it responds with speed reduction. $CO_2$ emissions coming from a containership in a cycle route between $i$ and $j$ will be

$$CO_2 = 3.17 * \left(F_{ijk}{}' + F_{jik}{}' + AF_{kp} * D_P\right) \tag{8}$$

$$F_{ijks}{}' = \left[MF_k * \left(\frac{S_{ijk*(1-x)}}{S_{0k}}\right)^3 + AF_{ks}\right] * \frac{d_{ij}}{24S_{ijk*(1-x)}} \tag{9}$$

$$F_{jiks}{}' = \left[MF_k * \left(\frac{S_{jik*(1-x)}}{S_{0k}}\right)^3 + AF_{ks}\right] * \frac{d_{ji}}{24S_{jik*(1-x)}} \tag{10}$$

where $x$ is the rate of speed reduction. $S_{jik}$ is the operational speed at sea of containership $k$ from $j$ back to $i$. $d_{ji}$ denotes the distance from $j$ to $i$ ports. $D_P$ is the total days ship $k$ stays in port in the cycle route between $i$ and $j$. $F_{ijks}{}'$ and $F_{jiks}{}'$ are fuel consumptions at sea after speed modification.

Based on the marine emissions trading scheme proposed above, $CO_2$ emissions exceeding the free quantity should be auctioned, and those exceeding the cap of allowances should be purchased in the market. This paper assumes that ship operators will not reduce speed sharply to make profit on the saving allowances. Then annual profit of a containership after slow steaming under the scheme can be calculated as follows:

$$\pi_k = T_{ij} * \left\{R_{ij} * W_{ijk} + R_{ji} * W_{jik} - \left[C_{ijks}{}' + C_{jiks}{}' + C_{kp}\right] - C_{co2}\right\} \tag{11}$$

$$C_{ijks}{}' = \left[C_k + P_M * MF_k * \left(\frac{S_{ijk*(1-x)}}{S_{0k}}\right)^3 + P_A * AF_{ks}\right] * \frac{d_{ij}}{24S_{ijk*(1-x)}} \tag{12}$$

$$C_{jiks}{}' = \left[C_k + P_M * MF_k * \left(\frac{S_{jik*(1-x)}}{S_{0k}}\right)^3 + P_A * AF_{ks}\right] * \frac{d_{ji}}{24S_{jik*(1-x)}} \tag{13}$$

$$C_{kp} = \left(P_A * AF_{kp} + C_k\right) * D_P \tag{14}$$

$$C_{co_2} = \left\{P_a * \left[CO_{2(F)} - CO_2\right] * n + P_p * \left[CO_{2(C)} - CO_2\right] * m\right\} \tag{15}$$

where $P_a$ denotes the auction price of $CO_2$ allowance. $P_p$ denotes the purchase price of $CO_2$ allowance. $CO_{2(F)}$ denotes the free allowance a ship gets. $CO_{2(C)}$ denotes $CO_2$ allowance cap for a ship. $C_{co_2}$ is $CO_2$ cost. $\pi_k$ is the total annual profit of ship $k$. $n$ and $m$ are constants, which values are either 0 or 1.

When

$$CO_{2(F)} > CO_2$$

$n$=0, $m$=0, which means there is no $CO_2$ emissions cost. When

$$CO_{2(F)} < CO_2 < CO_{2(C)}$$

$n$=1, $m$=0, there is $CO_2$ auction cost, but no purchase cost. When

$$CO_2 > CO_{2(C)}$$

$n$=1, $m$=1, there are $CO_2$ auction and purchase costs.

## 4 Case Study

This paper takes an international liner route, i.e., Asia to North Europe, as an example to evaluate the influence of the marine emissions trading scheme on the annual profit of a ship. There are three reasons for the choice of this route. At first, it is one of the typical shipping lines for international containerships. Moreover, one end of this shipping line is Europe, which is active in the marine emissions trading scheme. Fig.1 shows the explicit information of the route.


**Fig. 1.** Shipping Route for the Case Study (Source: China Ocean Shipping (Group) Company)

We assume a containership operating in this line with a capacity of 8000 TEUs. Its design speed is 24.1 knots and the corresponding main engine fuel consumption is 11.9 tons/h. Hence, the main engine maximum daily fuel consumption ($MF_k$) is 285.6 tons. Its auxiliary engine fuel consumption is 2 tons/day at sea and 1 ton/day in port. In general, trips of the Asia to north Europe route are differentiated by westbound and eastbound. For the case study in this paper, westbound means a voyage from Shanghai to Felixstowe. On the contrary, eastbound is a trip from Felixstowe back to

Shanghai. The ship works 350 days each year. To each round trip, the ship stays in port for 11 days in total. Moreover, according to a survey from the ship operator, the average daily fixed cost ($C_k$) of this containership operating on the Asia-Europe route is 114667 USD and the average loading factor is 85%.

From Table 2 we can see that the volumes rate of container trade in 2013 between eastbound and westbound is nearly 0.5. Combining the average freight rate of a 40ft container for Asia-Europe route in 2013 (shown in Table 3), and average loading factor, we can evaluate the annual revenue of the ship.

**Table 2.** Forecast development of Asia to North Europe Container Trade Volumes in 2013 (Source: Container Market Review and Forecaster, Quarter 1/2013, Drewry Maritime Research)

| Quarter | 1Q | 2Q | 3Q | 4Q | Total |
|---|---|---|---|---|---|
| Westbound (000 TEU) | 2210 | 2231 | 2394 | 2185 | 9020 |
| Eastbound (000 TEU) | 1172 | 1122 | 1103 | 1160 | 4557 |

**Table 3.** Container Freight Rate Benchmarks (Spot Market) for Asia-Europe Route in 2013, all-in (USD per 40ft container) (Source: Sea & Air Shipper Insight, Drewry Maritime Research, April 2013 - Issue no. 04)

| Time | Jan. 2013 | Feb. 2013 | Mar. 2013 | Average |
|---|---|---|---|---|
| Westbound | 2959 | 2575 | 2572 | 2702 |
| Eastbound | 1263 | 1230 | 1267 | 1253 |

Note: All-in rates include base rate, BAF, other surcharges and terminal handling charges at origin and destination.

In this paper, the bunker fuel prices are referred to Rotterdam bunkers price on May 3rd, 2013. Price of MDO[1] and 380 cst[2] is, respectively, 830 USD/ton and 576 USD/ton. Auction price and purchase price for $CO_2$ allowance in this paper is 25 USD/ton and 30 USD/ton, because analysts pointed out that only when the $CO_2$ allowances price is 20 to 30 EUR/ton it can promote entities search measures to cut down GHG emissions [26].

Applying Equation (4) to this case, we calculate the profit-maximizing speed of the ship. As profit-maximizing speed only relates to ship's engine and fuel prices, profit-maximizing speed in westbound and eastbound of the ship are both 17 knots when there is no emissions costs. If the ship operated in this speed in 2012, its corresponding $CO_2$ emissions should be 95275.3 tons.

When the Marine Emissions Trading Scheme proposed in this paper put into effect, the $CO_2$ emissions allowances for this ship will be 76220.24 tons, 80% of 2012. In addition, annual profit and $CO_2$ emissions of the ship with different speed reduction rates are shown in Tables 4 and 5.

[1] Fuel for auxiliary engine.

[2] Fuel for main engine.

## 5 Discussion

Table 5 demonstrates that speed reduction is a useful measure in cutting down $CO_2$ emissions of a ship. Fig. 2 illustrates two phenomena. Firstly, with the rise of $CO_2$ allowances auction rate, the annual profit of the ship is descending quickly when the ship speed is fixed. Due to this phenomenon, auction for $CO_2$ allowances could be a financial burden to ship carriers. Secondly, between all the auction rates proposed in this paper, when loading the factor is 85%, ship speed reductions of 5%, based on its original profit-maximizing speed, should be the most profitable choice for a ship operator comparing with other speed reduction rates.

**Table 4.** Annual profit (USD) of the ship with different speed reduction rates in five allowance auction rates

| Auction Rate | Speed Reduction Rate | | | |
|---|---|---|---|---|
| | 0 | 5% | 10% | 15% |
| 0.2 | 1495739.2 | 1645158.9 | 1362113.4 | 1021391.1 |
| 0.4 | 1114638.0 | 1264057.8 | 981012.2 | 259188.6 |
| 0.6 | 733536.8 | 882956.66 | 599911.0 | -121912.6 |
| 0.8 | 352435.6 | 501855.3 | 218809.8 | -503013.8 |
| 1 | -28665.6 | 120754.1 | -162291.4 | -884115.0 |

**Table 5.** Annual $CO_2$ emissions (tons) of the ship in different speed reduction rates

| Speed Reduction Rate | 0 | 5% | 10% | 15% |
|---|---|---|---|---|
| Annual $CO_2$ emissions | 95275.3 | 82744.8 | 71251.2 | 60853.3 |

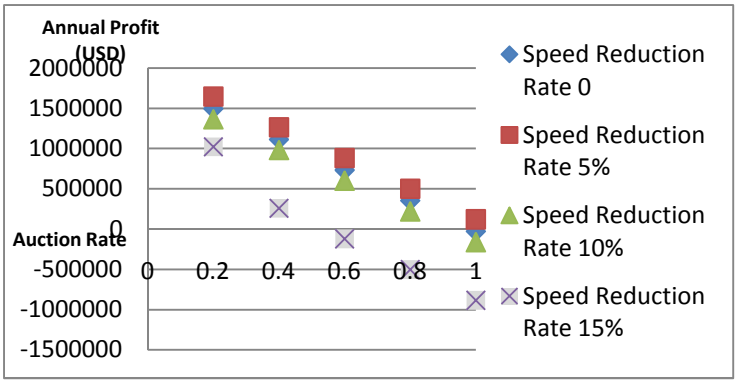



**Fig. 2.** Trends of the annual profit of the ship with different speed reduction rates when the auction rate increases

Annual profit of a ship is not only influenced by $CO_2$ allowances auction rate, but also by the shipping market. Generally, the shipping market has a great impact on the profit of a ship, which might influence the implementation of this scheme. This article studies the acceptability of this trading scheme from the perspective of profit.

In the international shipping market, loading factor and freight rate are key influential factors of ship profit, which fluctuates with market. However, in most cases, the loading factor denotes the situation of the shipping market better than the freight rate since modifications of the freight rate are aroused by fluctuation of the loading factor. Hence, this paper tries to find out how the marine emissions trading scheme influences ship profit when the loading factor is different and to discuss its acceptability.

As this article has calculated the annual profit of the case ship when its loading factor is 85%, in the subsequent context, we continue to study its annual profit based on other four loading factors as comparison. Like for the 85% loading factor, the ship has the most profitable speeds in each of the other four loading factors studied, no matter what the auction rate is.

**Table 6.** Profitability of different loading factors under the Marine Emissions Trading Scheme

| | Loading factor | | | | |
|---|---|---|---|---|---|
| | 95% | 90% | 85% | 80% | 75% |
| Speed reduction rate | 0% | 5% | 5% | 5% | 10% |
| Maximum Annual profit (USD) | 8556193.8 | 5030691.8 | 1645159.0 | -1740373.9 | -5107001.2 |
| Auction rate | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| Minimum Annual profit (USD) | 7031789.0 | 3506287.0 | 120754.1 | -3264778.7 | -6631406.0 |
| Auction rate | 1 | 1 | 1 | 1 | 1 |

Table 6 shows obviously that with rising loading factor, ship operators prefer higher speeds to gain more profits, ignoring $CO_2$ emissions. As to the acceptability of this scheme, loading factors from 85% to 95% finally result profits to the ship which demonstrate it could be accepted in theory. Of course, from the profit perspective, lower auction rates are more acceptable by ship operators. However, when shipping market is declining, such as loading factor is 80% or even less, the scheme will cause negative profits to the ship, which increases the difficulty of its implementation. Therefore, in a depressing market, in order to increase the incentives of ship operators to adopt this scheme, it can be implemented with revenue recycling or some policies.

## 6 Conclusions and Further Research

This paper proposes a potential marine emissions trading scheme with a focus on $CO_2$ emissions to research its influence on the annual profit and $CO_2$ emissions of a containership when ship speed reduces correspondingly.

The initial allocation measure in the marine emissions trading scheme is a combination of grandfathering and auctioning, which considers not only the financial burden to ship operators, but also emissions reduction pressure to the whole industry. Based on scenario analysis and referred to EU ETS, this paper proposes five possible auction rates. A case study based on a containership running the Asia-Europe route with 8000 TEUs capacity is conducted to test the influence of the marine emissions trading scheme on the annual profit and $CO_2$ emissions of a containership with the response of speed reduction. The results of this case study demonstrate that speed reduction is an effective measure in cutting down GHG emissions of a ship. In addition, the auction rate of allowances has a strong impact on the annual profit of a ship which drops sharply as the auction rate increases. Furthermore, from the profit perspective, ship operators' choice on speed reduction and their incentive to adopt the scheme turn out to be related with the situation of the shipping market. Hence, the auctioning rate among the cap of emissions for a ship may refer to the international shipping industry or be supported by policies such as revenue recycling.

In order to verify the feasibility of this scheme, in the further research multiple containerships and other typical routes should be considered. As far as acceptability of this scheme, this paper discusses it mainly from the profit perspective. More attention should be paid on social welfare, a combination of other technical or operational measures to further study the implementation of the marine emissions trading scheme.

**Acknowledgement.** This research is performed under the financial supports of the Ministry of Science and Technology of the People's Republic of China with Project No. 2012BAC20B03-08 and the China Clean Development Mechanism Fund with Project No. 1213094-3.

## References

1. AEA Group: Greenhouse gas emissions from shipping: trends, projections and abatement potential (2008)
2. IMO: Second IMO GHG Study. MEPC 59/INF.10 (2009)
3. European Commission (2013), http://ec.europa.eu/clima/policies/transport/shipping/index_en.htm
4. Crist, P.: Greenhouse gas emissions reduction potential from international shipping. Paper Presented at the International Transport Forum, Leipzig (2009)
5. IMO: Study of Greenhouse Gas Emissions from ships. MEPC 45/8 (2000)
6. Tanaka, Y.: Economical speed and life cycle value of ship. In: 4th Japan Towing Tank Conference, Japan (2003)
7. Wik, C., Hallback, B.: Utilisation of 2-stage turbo charging as an emission reduction means on a Wärtsilä 4-stroke medium-speed diesel engine. In: Proceedings of the 25th CIMAC World Congress on Combustion Engines, Vienna (2007)
8. Corbett, J.J., Winebrake, J.J.: Emissions tradeoffs among alternative marine fuels: total fuel cycle analysis of residual oil, marine gas oil, and marine diesel oil. Journal of the Air & Waste Management Association 58(4), 538–542 (2008)

9. Sims, R.E.H., Rogner, H.H., Gregory, K.: Carbon emission and mitigation cost comparisons between fossil fuel, nuclear and renewable energy resources for electricity generation. Energy Policy 31(13), 1315–1326 (2003)
10. Mathiesen, B.V., Lund, H., Nørgaard, P.: Integrated transport and renewable energy systems. Utilities Policy 16(2), 107–116 (2008)
11. Psaraftis, H.N., Kontovas, C.A., Kakalis, N.M.P.: Speed reduction as an emissions reduction measure for fast ships. Paper Presented at the 10th International Conference on Fast Sea Transportation, Athens (2009)
12. Corbett, J.J., Wang, H., Winebrake, J.J.: The effectiveness and costs of speed reductions on emissions from international shipping. Transportation Research Part D: Transport and Environment 14(8), 593–598 (2009)
13. Eide, M.S., Endresen, Ø., Skjong, R., Longva, T., Alvik, S.: Cost-effectiveness assessment of $CO_2$reducing measures in shipping. Maritime Policy & Management 36(4), 367–384 (2009)
14. Kageson, P.: Linking $CO_2$ Emissions from international shipping to the EU ETS (2007), http://www.natureassociates.se/pdf/nya/CO2%20shipping%20final.pdf
15. Kageson, P.: The maritime emissions trading scheme (2008), http://www.natureassociates.se/pdf/METS%20final.pdf
16. Aldy, J.E., Stavins, R.N.: The promise and problems of pricing carbon: theory and experience. The Journal of Environment & Development 21(2), 152–180 (2012)
17. Miola, A., Marra, M., Ciuffo, B.: Designing a climate change policy for the international maritime transport sector: Market-based measures and technological options for global and regional policy actions. Energy Policy 39(9), 5490–5498 (2011)
18. Meyer, J., Stahlbock, R., Voß, S.: Slow steaming in container shipping. In: Sprague, R.H. (ed.) Proceedings of the 45th Annual Hawaii International Conference on System Sciences, pp. 1306–1314. IEEE, Piscataway (2012), doi:10.1109/HICSS.2012.529
19. Burniaux, J.M., Duval, R., Chateau, J., Jamet, S.: The economics of climate change mitigation. OECD Economics Department Working Papers (2008)
20. Goers, S.R., Wagner, A.F., Wegmayr, J.: New and old market-based instruments for climate change policy. Environmental Economics and Policy Studies 12(1-2), 1–30 (2010)
21. Litman, T.: Evaluating carbon taxes as an energy conservation and emission reduction. Transportation Research Record: Journal of the Transportation Research Board 2139, 125–132 (2009)
22. Perdan, S., Azapagic, A.: Carbon trading: current schemes and future developments. Energy Policy 39(10), 6040–6054 (2011)
23. Bäuerle, O., Graichen, J., Meyer, K., Seum, S.: Integration of marine transport into the European emissions trading system: environment, economic and legal analysis of different options (2010), http://www.umweltdaten.de/publikationen/fpdf-l/3942.pdf
24. Wit, R.C.N., Boon, H.B.: Giving wings to emission trading-inclusion of aviation under the European Emission Trading System (ETS): Design and impacts, Delft CE (2005)
25. Ronen, D.: The effect of oil price on containership speed and fleet size. Journal of the Operational Research Society 62(1), 211–216 (2010)
26. YICAI (2013), http://www.yicai.com/

# Dynamic Routing on OpenStreetMap Using Ant Colonies

Alexander Bertram[1,⋆] and Sebastian Iwanowski[2]

[1] TXS GmbH, Sonninstrasse 28, 20097 Hamburg, Germany
alexander.bertram@txs.de
[2] FH Wedel, University of Applied Sciences, Feldstr. 143, 22880 Wedel, Germany
iw@fh-wedel.de

**Abstract.** This paper presents the software AntScout for dynamic road navigation: The objective is to find the shortest path between arbitrarily chosen nodes in a road network considering the current traffic situation. AntScout reacts immediately to any change in the query or in the current traffic situation. This is achieved by the continuous application of ant colonies which are specially designed for problems with unexpected dynamic input change. The navigation tasks are performed on OpenStreetMap data which have been specially processed for that purpose.

AntScout provides an interface ready for use by third parties. It is published under an open source license. Modern implementation techniques such as the Scala actor concept are applied in order to make the software applicable for real world situations. Conceptually, some deficiencies of ant algorithms have been detected and repaired. The experiments with AntScout discovered the positive side effect that ant colony systems tend to toggle between several preferred routes nondeterministically when these routes are of similar quality. Thus, ant based navigation can also be used to distribute traffic uniformly among such routes without any further control actions.

**Keywords:** ant colonies, probabilistic approach, dynamic routing, OpenStreetMap, uniform traffic distribution.

## 1 Motivation

Nowadays, it is standard that cars have got a navigation device on board. For static conditions where the quality of roads is well-known and does not change the systems work fairly well.

Perhaps the greatest challenge is to keep the digital maps up-to-date with the real world. Another interesting topic of investigation is the question how to measure the quality of roads. This need not necessarily be the highest feasible maximum speed, but may also consider other criteria like road comfort, economic evaluations, interesting scenery, etc. In recent years, least effort of research has

⋆ This work was done while the author was working on his Master's degree at FH Wedel.

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 58–72, 2013.

been put into the algorithmic processing of the data available: The algorithms used work fairly well on present hardware even for big networks.

However, in the dynamic case, the situation is much different. By dynamic routing we understand the problem to find the best path between arbitrarily chosen points in the network considering the current quality of the road segments which is subject to dynamic and not predictable change. The major focus of application is to consider the current penetrability of the road as quality measure. The objective for this case is to find the shortest path considering the highest feasible speed for each road segment.

The question how to get the dynamic information for all road segments is not solved yet. Current technological approaches are Floating Car Data (FCD), Floating Phone Data (FPD) and Traffic Message Channel (TMC).

This work does not address the question how to get the dynamic data. The cited technologies make rapid progress. As an example, we refer to the Google FPD approach which shows a partially rather up-to-date traffic situation even for city streets. We assume that this problem will be solved with sufficient accuracy within a couple of years. Instead, we ask a different question: How do we process the huge amount of dynamic data such that we always have an up-to-date answer for the currently best route?

Note that the answer to this problem is not trivial, even if dynamic information is readily available for each road segment because the answer for the currently best route may be influenced by any road segment "between" start and destination where "between" may even consider deviations towards a different direction.

Ant algorithms have been specially designed to answer routing problems for such dynamic scenarios. They have already been successfully applied to computer networks and logistic problems. This work investigates how they behave in real road scenarios.

Also other work (e.g. [11]) has already investigated this question. However, the other authors did not work with real digital maps, but artificial maps. We decided to use OpenStreetMap (OSM) data, because the open standard of OSM makes it easy to extract the data we need and to insert any new dynamic information. Furthermore, the wiki approach of OSM gives hope that OSM will also solve the entry question addressed above with best quality, that is, how to keep the digital map up-to-date with reality.

## 2 State-of-the-Art and Its Limitations

### 2.1 Routing in State-of-the-Art Navigation Systems

Routing in static navigation systems is usually performed on a map which is available on-board. The algorithms used follow the basic idea of Dijkstra's algorithm which is easy to implement (the original is found in [6]). That algorithm is even theoretically optimal for networks with a constantly bounded number of neighbors for each junction (which is a realistic assumption).

Several improvements of the original algorithm such as the A* algorithm as well as special preprocessings of maps make it affordable to compute the optimal answer to each query within a very short response time. For a fairly recent analysis see [12].

Even small on-board-devices containing maps with more than 100000 junctions get the answer within fractions of a minute. This is why traditional navigation systems start computation only after the query is asked.

Nowadays, dynamic information referring to the current status of the road network is also provided. The most popular method applied in nearly all devices is a repeated update of information about traffic conditions on highways via Traffic Message Channel: The updated information is plugged into the local map of the on-board-device which recomputes the route for a target the vehicle is currently heading to. Computationally, this is feasible only, because the dynamic information provided is very scarce in time and does not apply to all road segments. Thus, recomputation does not occur very often. Since it is normally used for highways only, the next junction ahead where the old and the new answer may differ, is normally far away which leaves enough time for the complete new computation.

However, the situation would be different if we admit real-time information for all road segments of the network (not only highways) and if we also apply dynamic routing for city navigation. Then a recomputation would not be feasible, because the update would have been needed more frequently, and the driver would have to react within seconds to a new situation. Furthermore, and this is the most important argument, it is not feasible to transmit updates about each road segment of the total road network every minute to each mobile device in order to be integrated into on-board computation.

This suggests that future dynamic road navigation must be computed off-board on a supercomputer (which may be a virtual one using cloud computing) being provided with the most recent information about the status of all road segments. For this scenario, the on-board navigation devices would not do the actual computation anymore. Instead, it would rather transmit the queries of the user to the supercomputer and receive the computed route from the supercomputer and display it to the driver. In fact, in the front end the driver would not see any difference to present systems. In order to be fault-tolerant to mobile transmission failures, the traditional functionality of the navigation device may be applied as backup solution.

Smartphones using Google or other digital maps already comply to this off-board principle. Google maps can already display the current traffic situation which is retrieved from the mobile cell-phone users. This can be achieved not only for highways but also for city traffic. However, Google apparently does not integrate the dynamic information into the optimization algorithm: Google shows several (usually 3) routes which are reasonable for normal traffic conditions. For these suggestions, Google also shows the current situation. But Google would not include alternative routes when all routes suggested are disadvantageous under current conditions. And even if Google did, the open question would be how the

continuously changing current status of the road segments is integrated into the optimization process.

### 2.2 How Ant Systems Perform Routing Tasks

Ant algorithms are explicitly designed to perform dynamic routing tasks. They resemble the behaviour of ant colonies in nature. The difference to classical routing algorithms is the following:

**Eager computing:** Ant systems compute routing information continuously between any potential query points. This is called the eager computing strategy: The result is already computed for the current situation before the query is asked.

**Off-board middleware:** Ant systems are performed on an off-board middleware which is the necessary prerequisite for passing current information gathered by thousands of individuals to thousands of applying users.

**Statistical compression of individual information:** The dynamic information obtained by individuals is not stored individually but collected in **pheromones** which are information chunks containing quality information for links with respect to a certain target. For each target, each link has got its own pheromone value. This considerably condenses the information individually obtained. Since the pheromones are target-oriented, the pheromones do not just evaluate single road segments, but the total route from the place they are stored to the desired target.

**Local information first:** The local use of pheromones enables the system to give the answer in which direction to proceed within fractions of a second even if the total route has not been computed yet. This even applies to dynamic changes that just occurred.

First, we describe general properties all artificial ant systems share. Ant systems operate on an underlying graph and solve dedicated constraint problems. These problems may be scheduling or path finding tasks or combinations thereof. In principle, any constraint problem may be solved with ant systems as long as the underlying graph is properly defined.

An (artificial) ant is a software unit. Such units are continuously generated over time by the ant system. Each ant uses the current state of data of the underlying graph, considers the current constraints to be solved at the time of its generation, and tries to find a single solution for this problem. Each ant is influenced by the pheromones of the graph links lying directly ahead. It will not automatically use the link with the best pheromone values, but rather decide that probabilistically: The better a pheromone value, the more likely an ant will use it. After having found a solution, each ant gives feedback modifying the pheromones attached to the links this ant has used for its solution. The amount of modification depends on the quality of the result discovered by the ant. Thus, the pheromones represent the collected memory of previous ants having used the respective edge. Subsequently generated ants are biased by these pheromones for

their own construction of a solution. This is how they resemble the behaviour of real ants.

Ant systems found quite a few applications in logistics (for a real life application, cf. [2]). In these applications, the ant systems solve different variants of the vehicle routing problem.

In the following, we confine to the application of path finding in navigation systems: In this application, ant systems use the normal road network as graph. The nodes are the junctions and the edges are the individual road segments. For each node, there is a routing matrix evaluating the quality of using a certain adjacent edge in order to reach a certain selected target. This corresponds to the standard computer network protocols where ant systems have already been successfully applied (cf. [4]). The quality measures of this matrix are the pheromones. Note that the pheromones attached to an edge are not the same as the current cost function for this edge, because the pheromones do not only consider this edge but also all edges possibly behind on the way to the target.

For each (source, target) pair, ants are continuously produced at their source node. The task of each ant is to find a path to the specified target node.

For each path finding, the probability that an ant selects a certain edge depends not only on the quality of pheromones attached so far, but also on some heuristic value, which may directly involve the graph's cost function. The tradeoff may be tuned by several parameters. In general, the mutual consideration of task oriented pheromones and network oriented heuristic values ensures a balance between the emergence of stable paths and the adaptivity to dynamic changes.

In navigation, the path finding and updating functionalities of ants are temporally interweaved. This enables a highly parallel process, but this requires a suitable software environment to support that. In other applications of ant algorithms like scheduling, the solution finding and updating phases are more separated.

After an individual ant has found a path to the target, the update of the pheromones works as follows: The pheromone values of the edges used are increased, and the pheromones of the edges not used (but adjacent to the nodes used) are decreased. The latter process resembles evaporation which has been proven very practical in nature, because this gives more recent data a higher influence than older data. However, the difference gap by which the pheromones of the edges used are increased and the others are decreased depends on the quality of the overall route: The longer the travel time, the smaller the difference gap. This technique results in a rather fast emerging of good routes used by the majority of the ants.

For further information about ant systems and the details of the formulae used, we refer to the standard literature such as [9], [7], [8], [13]. Also in the master's thesis [1] which is the long version of this paper, the formulae are elaborated explicitly.

### 2.3 OpenStreetMap

OpenStreetMap[1] is the largest free geographic database of the world which is rapidly growing. OpenStreetMap gathers and updates its content from volunteer helpers throughout the world. Thus, it applies the same principle as wikipedia. The number of registered helpers is almost 1 million already. This makes OpenStreetMap the most accurate geographic database among all digital maps at least in countries with a lot of digitally oriented people, e.g., in most areas of Europe. Unlike other maps like GoogleMaps, OpenStreetMap has a specific strength also in foot paths and areas beyond roads.

Unlike other free maps like GoogleMaps, OpenStreetMap does not only provide the maps, but also the geodata themselves for free use. This makes it very prone for applying new technologies which is the main reason why we chose this map. Meanwhile, several routing providers are linked to at least the German version of OpenStreetMap[2]. However, they deal with static routing only.

The data of OpenStreetMap is provided in XML which makes it very convenient to be processed by any programming language with a web service interface.

Open StreetMap provides the concepts of `node` and `way`: A `node` is an arbitrary point defined by its geographic coordinates. A `way` is defined by a sequence of nodes. This is used to model roads, borders and rivers and any other object of 1 dimension.

`Road segments` are a special kind of `ways`. They are classified in several categories: The highest category is `motorway` and the lowest `residential`. In between there are the categories `primary`, `secondary`, `tertiary` and some more categories.

### 2.4 Conceptual Challenges

Ant systems are based on the eager computation principle: Each route query is computed in advance. Since the system must be prepared to answer any query, it must continuously update the best route between any two query points. This results in a number of queries that is quadratic in the number of nodes of the underlying graph. In order to ensure fast convergence of the pheromones to any dynamic situation, this quadratic number of queries must be repeated frequently within a minute.

In principle, this results in a run time that is a quadratic multiple of the run time of a single query. But there is hope to reduce this by parallelization and reduction of query nodes.

## 3 AntScout - Applying Ant Colonies on OpenStreetMap

AntScout is an open source software[3] published under the apache license version 2.0.

[1] `http://www.openstreetmap.org/`
[2] `http://www.openstreetmap.de/`
[3] `https://github.com/abertram/AntScout`

AntScout consists of two parts:

1. A preprocessing unit that extracts data from OpenStreetMap in order to enable the functionalities of the run time unit
2. A run time unit providing two functionalities:
   **user functionality:** performing navigation queries on the preprocessed map
   **operator functionality:** changing the feasible speed of selected road segments

The user functionality of the run time unit is the benchmark for the ant colony solution and the main part to be evaluated. The operator functionality corresponds to a traffic simulator and resembles dynamic behaviour, because this is the purpose for which we are using ant colonies.

### 3.1 Software Environment

As programming environment we used Scala which is a language integrating the principles of object-oriented and functional programming. The compiler produces Java byte code which may be executed on a Java Virtual Machine. Thus, Scala bears all advantages of Java as a programming language.

The reason for applying rather Scala than Java is its ability to provide parallelization: In contrast to Java's thread concept, Scala provides the actor concept of Erlang which is specially designed for parallelization. Originally, Scala had got an own actor library for that, but we rather used an open source framework called Akka[4]. In the meantime, Akka has been integrated in Scala's next version. Thus, our implementation is safe for future extensions. The strength of Akka is the smooth support of distributing parallel processes to several computers which enables cloud computing.

### 3.2 Preprocessing Functionality

The aim of the preprocessing functionality is to extract a network of nodes and links that is eligible for routing. In principle, each OpenStreetMap node that is part of a road segment may be a network node. But in order to reduce the complexity of the run time component, we reduce the number of nodes considerably by the following:

1. If an OpenStreetMap node is in between a road segment or connecting exactly two road segments, routing does not have a choice between alternatives. This is why we omit all such nodes and only consider nodes which are part of at least three road segments. Using graph terminology, we only consider nodes of degree at least 3.
2. We restrict to certain geographic areas using a rectangle placed upon OpenStreetMap data: Only nodes within this rectangle are considered, and only links between such nodes are considered, i.e., links stretching out of the rectangle are not considered.
3. We confine to predefined categories of road segments: Only road segments of this category or higher are considered.

[4] `http://akka.io/`

In order to efficiently reduce the number of nodes, the preprocessing unit combines these three principles simultaneously, i.e. it uses a predefined admissible rectangle and a predefined admissible lowest category and considers as admissible nodes only the nodes that are part of at least three road segments of that category or higher. A link in our graph is identified with a sequence of OSM road segments of the admissible categories. Note that the network obtained from this also contains nodes of degree 1 and 2, because some of the links incident to admissible nodes (which originally were all of degree at least 3) may stretch out of the admissible rectangle and, thus, have to be removed. For further reduction, we also merge two links meeting at a node of degree 2, thus avoiding nodes of degree 2. However, we leave nodes of degree 1 in the network which are dead ends.

The admissible rectangle may be arbitrarily chosen, and the admissible lowest categories are `motorway`, `motorway-link`, `trunk`, `trunk-link`, `primary`, `primary-link`, `secondary`, `secondary-link` and `tertiary`.

We applied the preprocessing to several areas of the town of Hamburg which is a densely populated city. Our reduction principles above applied to rectangles with areas up to 50 $km^2$. Using tertiary roads as lowest admissible categories provided a maximum of 150 nodes.

In off-city areas the area covered may be much higher, but we wanted to demonstrate that our dynamic routing software may even be applied in cities which we consider the hardest task due to the density of alternatives and the need for fast reaction to dynamic changes.

### 3.3 Run Time Functionality

Since our system should show how ant systems react to spontaneous traffic changes, we considered it crucial to integrate user and operator mode in the same interface, i.e., there should be no explicit switching between them: An operator action should be executed at any time, and also a user query should be specified at any time. Both types may be interweaved in an arbitrary order. The operator action referring to some road segment remains valid until a new action is executed for the same road segment. A user query is valid until a new query is asked.

AntScout always displays the route using the best pheromones between the currently valid source and target. This is considered the currently suggested route.

This gave us an easy method to measure the response time to any dynamic change represented by an operator action: We simply needed to measure the time between operator entry and visible route change.

Note that a car driver who is identified with the user of our setting has nothing to do with the individual ants of the colony system which are anonymous software units and not displayed at the interface at all: The car drivers always get the currently best answer, but the ants make a probabilistic decision for their route. The latter guarantees that there will always be ants using unfavourable routes which is important for the quick detection of dynamic changes.

Figure 1 is a screen shot of the user interface illustrating how the run time component of AntScout works:

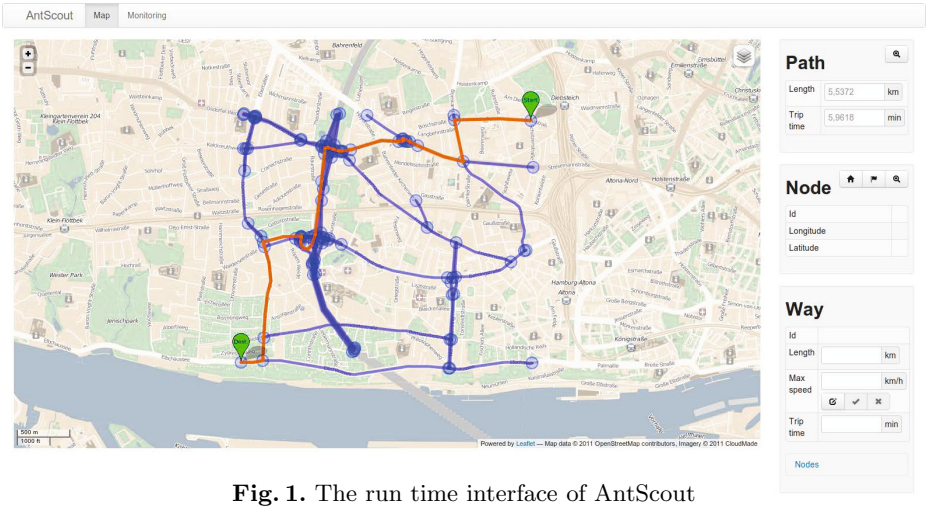



**Fig. 1.** The run time interface of AntScout

The user may select any node and make it to the source or target. AntScout will always show the currently best route in between. The interface entry **Path** shows the total time currently needed for the preferred route.

The operator may select any road segment: The interface shows the current maximum speed and the time for traversing the whole segment. The operator may change the maximum speed to any value simulating traffic jam or acceleration. The interface will respond with the new current traversing time in **Path** and possibly change the preferred route for the current user request.

### 3.4 Implementation Arrangements of AntScout

Due to the results of previous work at our department (cf. [14], English summary in [10]), we decided to implement the AntNet version which is the system described by Dorigo et al. [9]. For each node, AntNet works with a prepheromone matrix which is updated by the ants continuously and a pheromone matrix which is used by the ants for navigation. The prepheromone matrix is based on a statistical model using dynamic information only such as previous experiences of ants and flexible time windows. The pheromone matrix integrates static information into the values of the prepheromone matrix such as segment lengths. Both, prepheromone and pheromone matrices serve like a routing table for its node: There is a line for each possible target and a column for each adjacent node. A matrix entry (i,j) denotes the quality of using the link to adjacent node j if the target is node i.

The continuous generation of ants from each possible source node to each possible target node is the complexity bottleneck of the entire method. This is

why we did not model each ant as an independent actor for efficiency reasons. We rather modelled an ant to be a state of the nodes which are communicated and transformed among the neighbors. Thus, an ant colony system may be regarded a discrete state automaton. The handling of ants as passive states instead of active actors is a software acceleration which we see crucial for reducing the run time of a classical ant algorithm. Thus, an "ant" in our implementation is nothing else than a unit adding the current cost measures of the edges used until the target is reached.

### 3.5 Conceptual Improvements by AntScout

At initialization, the pheromone matrix has to start with some values. According to the theory, all prepheromones should start with the same values, because no ants have passed so far. If the segment lengths are added to that, this would result in a greedy strategy at the beginning.

We decided not to start with such a crude strategy and then let the ant colonies find suitable values statistically, but rather with initial values obtained from deterministic methods. Since we had to solve the all pairs shortest path problem, we did not apply Dijkstra's algorithm for each source, but rather applied the algorithm of Floyd-Warshall (for a modern description cf. [3], ch. 25.2) which is a little faster for that problem. This guarantees much faster convergence at the beginning.

Since the statistical path finding does not guarantee that each ant reaches its target at some time (e.g., the ant may get stuck in a loop), ants have to be removed after a while. [9] suggests to remove an ant, after the ant run into a loop. We first implemented this suggestion, but then found much better convergence when instead an ant is removed after a certain time has elapsed. For different networks, this time should depend on the size of the network.

Since the number of (source, target) pairs grows quadratic in the size of the network, it is important to control the number of ants generated for a given (source, target) pair. The trade-off is the following: The fewer ants are generated, the less computation load to the system (which is a trivial observation), but the more ants are generated in a certain time interval, the faster the expected convergence.

We applied the following approach to solve this dilemma: At each source $s$, ants are generated heading towards a predefined target $t$. Note that ants heading to a further distant target $t$, also pass other targets $t_i$ on their path from $s$ to $t$. The closer $t_i$ from $s$, the higher the probability that an ant originally heading to $t$ will also pass $t_i$. Catching up an original idea of Dorigo, Walther [10] already mentioned that the pheromones referring to targets $t_i$ should also be updated by ants heading to $t$ which would increase the accuracy of the pheromones referring to the trip from $s$ to $t_i$. Thus, if the generation frequency is the same for all (source, target) pairs, the accuracy for closer relations is much better than for further relations. This is exactly the opposite of what a user wants to see, because in closer relations the absolute quality difference between different routes is much less than for more distant relations. We compensate this by creating more ants

between more distant distance relations. For the implementation of this principle, we work with the following parameters:

**Distance groups:** The (source, target) pairs are collected into groups of "similar" distance. The distances are obtained using the static values computed at initialization.

**Launch interval:** This is the time interval between two single ants generated: The smaller the interval, the more ants are generated. In our implementation this interval is applied to the group of most distant pairs only.

**Launch delay:** This is the delay added to the launch interval for pairs of closer distance: The closer the distance, the greater the delay.

## 4 Tests and Results

Our test consisted of two parts: First, we did some experiments with parameters in order to adjust the scalability. Second, we evaluated the quality of AntScout from a user's point of view: How well and how fast does the ant colony react to a sudden change introduced by an operator.

The hardware on which we tested our software included an Intel Core i5-2520M processor with 4 x 2.5 GHz and a memory of 7.6 GB applied on a 64-bit system.

### 4.1 Adjusting the Scalability

In this investigation, we tested different values for launch interval and launch delay and tested their effect to quality which was measured by the number of ants who really reached their target and, thus, updated the pheromones. We compared this with the scalability of the system which was measured by the frequency in which a full garbage collection was started.

In a Java Virtual Machine, a full garbage collection is executed automatically when there are too many unreferenced objects. Such an event is dependent on the number of objects generated which in our case depends on the size of the network. The effect of a full garbage collection is an intermediate halt of the overall simulation. This has direct influence on the performance. This is why a full garbage collection is a good measure for scalability.

As an example, Fig. 2 shows the result for the map shown above consisting of 104 nodes.

Considering the dashed graph, it is a little surprising that the quality is not highest with the lowest launch interval. This effect is even more evident in larger maps. Our explanation is that a too frequent launch of ants is too fast for processing them, and this implies that quite a few of the generated ants will not reach their goal. On the other hand, the behaviour for larger launch intervals is expected because it is clear that fewer ants will reach their goal if fewer ants are generated.

Considering the solid graph, it is clear that a higher frequency of ant launch (corresponding to smaller launch intervals) will result in more garbage collection. This explains the monotonous decline of the corresponding graph. Since a

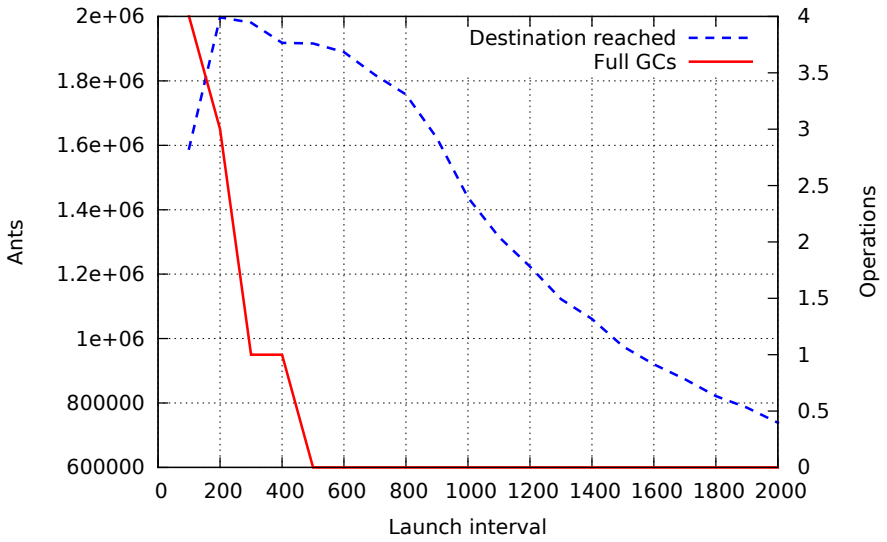



**Fig. 2.** Induced quality and scalability of ant generation

reasonable response time can only be achieved when there is no global garbage collection at all, we search for the minimum launch interval where global garbage collection does not occur. We denote this to be the *ideal interval*. In the respective diagram, this ideal interval length is 500 ms (where the solid graph intersects the abscissa).

We did similar investigations for other networks having the same number, less or more nodes. We achieved ideal launch intervals between 100 ms (for 61 nodes) and 1100 ms for 144 nodes. This meets our expectations that larger networks require longer launch intervals due to the quadratic increase of (source, target) pairs. For up to 150 nodes the quality of result (dashed graph) was still nearly optimum, but for larger networks it declined considerably. Thus, our approach successfully only applies to networks with up to 150 nodes, at least with the hardware we used.

For the launch delay, we showed in several tests for different launch intervals that it is best to keep the delay in a range of 20 % of the launch interval.

### 4.2 Evaluation of Results and Improving the Algorithm

The goal of our tests was to answer the following questions:

1. How does the system react when a preferred route deteriorates?
2. How fast does AntScout switch back into the initial state when a deteriorated route has improved again?
3. How does the system react when a so far nonpreferred route improves considerably such that now it would be a preferred candidate?

We tested this by a high variety of experiments to various (source, target) pairs in several maps. In many cases, the system behaved as wanted. The response time varied between instantly and less than 5 seconds. Only in some cases, we realized that the system got stuck in a nonoptimal route.

Careful analysis revealed the following: AntNet tends to prefer routes with fewer decision alternatives, i.e. paths with nodes of lower degree. This even holds when the selected route is considerably longer than the optimal one (more than 20 %).

By the following techniques, this behaviour can be avoided: Ants passing nodes with a higher degree get a higher relevance for updating the pheromones. An alternative or supplementing strategy is to make the number of generated ants dependent on the number of neighbors of the source: The more neighbors, the more ants are generated.

### 4.3 Oscillation

Due to the nondeterministic behaviour, the preferred routes oscillate between several candidates continuously.

We observed that such an oscillation only occurs when the routes do not differ more than 7 % from the average among all routes ever suggested. This implies that oscillating occurs only between routes that differ at most 14 %.

All experiments showed the following correlation: The more similar two routes were concerning their quality, the more oscillation occurred between them.

This is a reasonable behaviour which is easy to explain for a probabilistic algorithm. We suggest to utilize this for the following:

In real life, a driver who asked for a route should retain this route once he is suggested one. But the oscillating behaviour results in different routes for different drivers. Due to the observed correlation between frequency of oscillation and similarity of route qualities, this results in a perfect distribution of traffic among almost equal routes. Since this distribution is organized at random, we expect also a high social acceptance.

Thus, ant algorithms are very suitable for traffic management as a side effect.

## 5 Conclusion

This paper described the construction and evaluation of the software AntScout which applies ant colonies for routing in real road maps extracted from OpenStreetMap. AntScout is prototypic, but it offers an interface comfort that makes it testable by third parties.

AntScout proves that ant algorithms may be applied even without hierarchies in real maps with up to 150 nodes on a modern laptop. The response to sudden changes in the map (simulating traffic jam and its resolution) is rather fast (less than 5 seconds). The quality of the suggested routes is very reasonable: The simulation switches between the best route and alternatives that were within 14 per cent off the optimum.

Conceptually, the improvements to the state-of-the-art are the following:

1. We showed how to adapt OpenStreetMap maps such that they can be used for dynamic routing via ant colonies.
2. We showed that the published versions of ant algorithms tend to prefer routes with nodes of lower degrees and developed a concept how to avoid that.
3. We showed that the statistically fuzzy behaviour of ant algorithms can be used for traffic management with the goal of equally distributing the traffic. This makes congestions less likely to occur.

Next work to be done towards an integration into real navigation systems is the following:

1. The number of nodes extracted from OpenStreetMap can be further reduced: Quite a few OpenStreetMap nodes physically belong to the same junction. These nodes should be summarized to a supernode that serves as single source or target for the ants of the colony.
2. In order to enable a higher degree of parallelism, the ant simulations should be distributed to several computers. AntScout is perfectly prepared to this using the actor concept of the Akka library.
3. In order to improve the requirements of real navigation systems, AntScout should integrate other criteria than the mere passing time of road segments.
4. First attempts have already been published how to integrate hierarchies into ant algorithms conceptually (cf. [10], [5]). These suggestions should be elaborated and integrated into AntScout.

While the first three items are matters of implementation which do not require conceptual adjustments, the last item requires more conceptual research.

The physical visibility of the functionality of AntScout shows that a conceptually innovative and also very exotic paradigm such as ant colonies may be used for real world applications. We are convinced that navigation systems of the future will apply the following advantages of ant algorithms (elaborated in Section 2.2) regardless if they are called ant-based or not:

1. eager computing
2. off-board middleware
3. statistical compression of individual information
4. local information first

## References

1. Bertram, A.: Ant Scout - Dynamisches Routing auf OpenStreetMap. Master's thesis, FH Wedel (2012) (in German), `http://www.fh-wedel.de/fileadmin/mitarbeiter/iw/Abschlussarbeiten/MasterthesisBertram.pdf`
2. Blöcker, C., Iwanowski, S.: Utilizing an ant system for a competitive real-life planning scenario. In: COMPUTATION TOOLS 2012: Third International Conference on Computational Logics, Algebras, Programming, and Benchmarking, pp. 7–13 (2012)

3. Cormen, T., Stein, C., Leiserson, C., Rivest, R.: Introduction to Algorithms, 3rd edn. MIT Press (July 2009)
4. Di Caro, G., Dorigo, M.: Distributed stigmergetic control for communication networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
5. Dibowski, H.: Hierarchical routing system using ant based control. Master's thesis, TU Dresden (July 2003), http://www.kbs.twi.tudelft.nl/docs/MSc/2003/Dubowski_Henrik/thesis.pdf
6. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
7. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization – artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine 1, 28–39 (2006)
8. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1, 53–66 (1997)
9. Dorigo, M., Stützle, T.: Ant colony optimization. The MIT Press (2004)
10. Iwanowski, S., Walther, T.: Dynamic road navigation with ant algorithms. Technical report, Fachhochschule Wedel (2008), http://www.fh-wedel.de/fileadmin/mitarbeiter/iw/Abschlussarbeiten/DynamicRoadNavigationWalther.pdf
11. Lerebourg, S., Dutot, A., Bertelle, C., Olivier, D.: Management of the road traffic by an ant algorithm. World Conference in Transport Research Society, vol. 9 (July 2004), http://scott.univ-lehavre.fr/~bertelle/publications/mas2003_7-road_traffic.pdf
12. Sanders, P., Schultes, D.: Engineering fast route planning algorithms. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 23–36. Springer, Heidelberg (2007), http://algo2.iti.kit.edu/documents/routeplanning/weaOverview.pdf
13. Taillard, É.D.: An introduction to ant systems. In: Laguna, M., González Velarde, J.L. (eds.) Computing Tools for Modeling, Optimization and Simulation, pp. 131–144. Kluwer, Boston (2000)
14. Walther, T.: Dynamische Fahrzeugnavigation auf Basis von Ameisenkolonien. Master's thesis, Fachhochschule Wedel (February 2006), http://www.fh-wedel.de/fileadmin/mitarbeiter/iw/Abschlussarbeiten/MasterarbeitWalther.pdf

# On the Complexity of Computing Optimal Private Park-and-Ride Plans

Martin Olsen

AU Herning
Aarhus University, Denmark
martino@hih.au.dk

**Abstract.** We consider the problem where a group of people scattered in a weighted graph $G(V, E)$ has to travel by car to a common destination (the number of vertices may be much bigger than the number of people) where the weight of an edge is the cost of traveling along the edge with a car. Every member of the group has a car at their disposal and the objective is to minimize the total cost for getting all the members to the common destination. The members have the possibility of meeting and parking at the vertices and continue the journey sharing a car. We examine the computational complexity of the problem. Among other things, we show that the problem is APX-complete for the realistic setting where the cars have capacity 4 even restricted to undirected graphs where all edges have cost 1.

## 1 Introduction

In today's era of environmental awareness the concept of "ridesharing" where people share rides in private cars gains more and more attraction. The rising prices on fuels and the implementation of road pricing also pushes people in the direction of ridesharing and the activity of web services for coordinating ridesharing like, for example, www.carpooling.com is growing rapidly. When the author of this paper drives to work he passes a roundabout with a so-called "park-and-ride lot" seen in Fig. 1 where people driving can meet and park in order to continue their journey in one car (or fewer cars). This triggered the question: "How hard is it to compute an optimal scheme for private ridesharing?" In this paper we focus on the problem of computing an optimal plan for the situation where several people want to travel to the same destination and where they all have a car at their disposal and are willing to "park-and-ride" (using, e.g., dedicated "park-and-ride" lots) as just described. If we could solve this problem efficiently we could lower the cost for every participant. We will refer to this problem as the PRIVATE-PARK-AND-RIDE problem since no public transport is involved unless the common destination is a train station, bus station, airport, etc. The main contribution of this paper is a theorem stating that this problem is APX-complete even restricted to cars with capacity 4 and simple undirected graphs where the cost of driving along an edge with a car is 1. This shows that a constant $\epsilon > 0$ exists such that for *any* polynomial time algorithm computing

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 73–82, 2013.

**Fig. 1.** A "park-and-ride lot" at a roundabout

a feasible plan we have an instance $I$ such that the algorithm computes a plan with cost at least $(1+\epsilon)OPT(I)$ where $OPT(I)$ is the cost of an optimal plan (unless NP = P).

### 1.1 Related Work

Even though the literature on computational logistics is huge (we refer to the survey by Parragh et al. [13,14]) we have not been able to find literature that deals specifically with the computational complexity of the problem we focus at in this paper. Our problem is related to the "dial-a-ride"-problem where some *portables* have to be transported by *mobiles* from a given source to a given destination (the portables may have different sources and destinations). Our problem is also related to the special case of this problem where the mobiles are placed at a *depot*. A typical argument for the hardness of the problems mentioned above is that they are NP-hard because they generalize the traveling salesman problem (TSP). The key property making our problem special is that we have a mobile (car) placed at the location of each portable (person) for every instance which makes it impossible for us to use an argument similar to the TSP-argument mentioned above. Helmert [10] studies the computational complexity of the general case where mobiles can have arbitrary initial locations and the portables can have arbitrary sources and destinations and Helmert proves NP-completeness for several variants of the problem – Helmert also uses portables having no mobiles

at their initial location in the proofs. Helmert et al. [11] study the computational complexity for several logistics problems related to our problem. Baldacci et al. [4] propose an exact method based on integer programming for solving the problem where a set of so-called *servers* can pick up one or more *clients* and drive them to the same destination. Baldacci et al. do not allow "park-and-ride" as an option and they solve instances with only a few hundred people involved. Baldacci et al. also use an argument for NP-hardness relying on the fact that the sets of servers and clients are disjoint.

The Steiner tree problem is the problem of computing a minimum cost tree spanning a given set of vertices $V' \subset V$ – called *terminals* – in a graph $G(V, E)$. A fairly simple reduction shows that the problem we focus at in this paper generalizes the Steiner tree problem in the pretty unrealistic setting where the cars have infinite capacity. This means that our problem is harder than the Steiner tree problem if we allow cars with unlimited capacity. The reduction works by letting one of the terminals be the common destination and put a car with unlimited capacity and a person on every other terminal. Bern and Plassmann [5] show that the Steiner tree problem is APX-complete for undirected complete graphs with edge lengths 1 and 2 and Charikar et al. [6] argue that it is hard to approximate the directed Steiner tree problem to a factor better than $\ln k$ where $k$ is the number of terminals.

### 1.2 APX, PTAS and *L*-Reductions

We now give a very brief introduction to the complexity classes APX and PTAS and $L$-reductions that will be used in Sec. 2 to prove our main result. We refer to [3,8] for more details. A $(1+\epsilon)$-approximation algorithm for a minimization problem $\Pi$ is an algorithm that, given any instance $I$ of $\Pi$, computes a feasible solution $x$ to $I$ with $cost(x) \leq (1+\epsilon)OPT(I)$ where *cost* denotes the objective function and $OPT(I)$ denotes the minimum achievable cost for instance $I$. The problem $\Pi$ is a member of the complexity class APX if the problem admits a polynomial time $(1+\epsilon)$-approximation algorithm for *some* constant $\epsilon > 0$. If such an algorithm exists for *any* $\epsilon > 0$ then $\Pi$ is a member of the complexity class PTAS (Polynomial Time Approximation Scheme).

The hardest members of APX are the so-called APX-complete problems. If we could prove membership of PTAS for any APX-complete problem it would imply APX = PTAS having NP = P as a consequence [2]. Analogously to the NP/P-setting, we typically use reductions from problems known to be hard to prove completeness and the $L$-reduction introduced by Papadimitriou and Yannakakis [12] is often used. A problem $\Pi$ can be reduced to $\Pi'$ using a $L$-reduction written $\Pi \leq_L \Pi'$ if there are polynomial time functions $f$ and $g$ and constants $\alpha$ and $\beta$ such that the following conditions hold:

1. The function $f$ transforms any instance $I$ of $\Pi$ into an instance $f(I) = I'$ of $\Pi'$ such that $OPT'(I') \leq \alpha OPT(I)$.
2. The function $g$ transforms any feasible solution $y$ of $I'$ into a solution $x$ of $I$ such that $|OPT(I) - cost(x)| \leq \beta |OPT'(I') - cost'(y)|$.

If we have a polynomial time $(1+\epsilon)$-approximation algorithm for $\Pi'$ it is not hard to use $f$ and $g$ and obtain a polynomial time $(1+\alpha\beta\epsilon)$-approximation algorithm for $\Pi$. This shows that an $L$-reduction preserves membership of PTAS in the sense $\Pi' \in \text{PTAS} \Rightarrow \Pi \in \text{PTAS}$. We can show that a problem $\Pi'$ is APX-complete if we can prove the following for a problem $\Pi$ known to be APX-complete: 1) $\Pi' \in APX$ and 2) $\Pi \leq_L \Pi'$.

### 1.3 Contribution and Outline

We formally define the PRIVATE-PARK-AND-RIDE problem in Section 2 and establish a positive result when we restrict the problem to a constant number of persons. We also present a positive result for the case where the capacity of the cars is a fixed constant and where people only coordinate with their co-passengers in the car arriving at the common destination. Finally, we present our main result in Section 2 where we show that the problem is APX-complete in the realistic setting where all cars have capacity 4 even restricted to simple unweighted undirected graphs.

## 2 Preliminaries

In the following we refer to the persons involved in the PRIVATE-PARK-AND-RIDE problem as workers and we might think of the common destination as the workplace of the workers. We now formally define the problem of getting all the workers to their workplace at a minimum cost if they all have a car at their disposal and can pick each other up and drop each other any place. The costs could be any combination of fuel costs, tolls, congestion charges, etc. It is intuitively clear that a polynomial $p$ exists such that an optimal plan exists for any instance $I$ with a number of steps bounded by $p(|I|)$ and this is the polynomium we refer to below. Helmert [10] provides a relatively simple argument that such a polynomial exists when he considers the more general case. As a technicality the polynomial upper bound on the number of steps ensures membership of the complexity class NPO for the problem. We are now ready to state the formal definition of our problem.

**Definition 1.** *An instance of the PRIVATE-PARK-AND-RIDE problem is a* 6*-tuple* $(G(V,E), W, cap, cost, loc, t)$ *where*

- $G$ *is a graph where* $V$ *is the set of vertices in the transportation network and* $E \subseteq V \times V$ *is a set of edges.*
- $W$ *is a set of workers. Every worker has a car at his disposal.*
- $cap : W \to \mathbb{N}$*. The car owned by worker* $w$ *has capacity* $cap(w)$*.*
- $cost : E \to \mathbb{R}^+$*. The cost of traveling along an edge* $e$ *with a car is* $cost(e)$*.*
- $loc : W \to V$*. The starting location of a worker* $w$ *and the car owned by the worker is* $loc(w)$*.*
- $t \in V$ *is the target location for all workers. For each worker* $w$ *there is a path from* $loc(w)$ *to* $t$*.*

*A feasible solution of the PRIVATE-PARK-AND-RIDE problem is a plan for getting all the workers from their starting location to the target vertex $t$ and the objective is to compute a minimum cost plan. A plan is a sequence of no more than $p(|I|)$ steps where only the following steps are allowed and the cost of a plan is the sum of the costs of the steps:*

1. *A car at vertex $u$ can drive along the edge $e(u, v)$ and transport the workers in it to vertex $v$ with a cost of $cost(e)$.*
2. *A worker can leave a car at a vertex with a cost of 0. A worker can only leave his own car if no other workers are on board in which case the car can not move anymore.*
3. *A worker can enter a car at a vertex if the capacity of the car allows it. The cost of such a move is 0.*

For our first result we examine the computational complexity for the case where the number of workers is a fixed constant.

**Theorem 1.** *The PRIVATE-PARK-AND-RIDE problem can be solved in polynomial time if $|W|$ is a constant.*

*Proof.* We provide a sketch of the proof. We refer to a given distribution of the workers and cars on the vertices as a *state*. We construct a state transition graph where the transitions are the steps mentioned in Definition 1. We assign a small cost to leaving and entering a car and the transition corresponding to driving along an edge $e$ is assigned the cost $cost(e)$. We now find the solution by computing the shortest path from the initial state to a goal state (all workers at vertex $t$). A simple example with a part of the state transition graph is shown in Fig. 2 where two workers have to travel to a neighbour vertex for the smallest possible cost. The algorithm runs in polynomial time since the number of states is bounded by a polynomial ($|W|$ is constant). □

In some situations, we maybe choose only to consider plans where a worker $w$ only has to coordinate with the workers arriving to $t$ in the same car as $w$ and we will refer to such a plan as a *low-coordination* plan. There are cases where the optimal plan is not a low-coordination plan as can be seen from the example in Fig. 3. We now present our second positive result where $H(k) = \sum_{i=1}^{k} \frac{1}{k}$ denotes the $k$-th harmonic number:

**Theorem 2.** *There is a polynomial time $H(k)$-approximation algorithm for the PRIVATE-PARK-AND-RIDE problem restricted to low-coordination plans if all cars have capacity $k$ for some constant $k$.*

*Proof.* Once again, we provide a sketch of the proof. For each subset $S$ of workers satisfying $|S| \leq k$ we can use the state transition technique from the proof of Theorem 1 with $W = S$ to examine whether the workers in $S$ can coordinate and arrive to $t$ in the same car – and compute an optimal way of doing it and the corresponding cost if it is possible. In this way we can solve our problem using a polynomial time reduction to WEIGHTED SET COVER [7] with

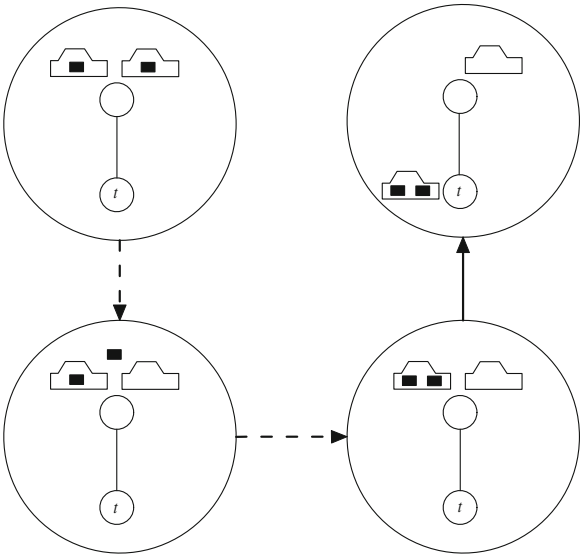



**Fig. 2.** Two workers have to travel to a neighbour vertex for the smallest possible cost. The big circles correspond to states and the dashed arrows indicate leave- or enter-transitions. The optimal plan has cost 1.

set cardinality bounded by $k$. For a given solution of the WEIGHTED SET COVER instance we can efficiently construct a low-coordination plan for the PRIVATE-PARK-AND-RIDE instance with the same cost – or a lower cost if the sets overlap (this is possible because all workers have a car with the same capacity). The WEIGHTED SET COVER problem admits a polynomial time $H(k)$-approximation algorithm if the set cardinality is bounded by $k$ [7]. □

## 3 Inapproximability for Simple Graphs and Car Capacity 4

In this section we show that the PRIVATE-PARK-AND-RIDE problem is APX-complete when restricted to cars with capacity 4 and unweighted and undirected graphs meaning that all edges have cost 1. We give this problem a special name:

**Definition 2.** *The PRIVATE-PARK-AND-RIDE-UNW-4 problem is the restriction of the PRIVATE-PARK-AND-RIDE problem to undirected unweighted graphs (cost*$(e) = 1$ *for every edge* $e$*) and cars with capacity* 4*.*

Before introducing the problem that will form the starting point for our reduction we kindly remind the reader on some terminology: A cubic graph is a graph where all vertices have exactly three neighbours and a dominating set of a graph $G(V, E)$ is a set of vertices $V' \subset V$ such that any member of $V \setminus V'$ has at least one neighbour in $V'$. The following problem will form the basis for our $L$-reduction:

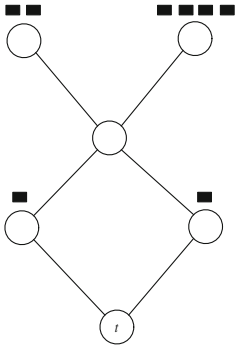



**Fig. 3.** An example of a PRIVATE-PARK-AND-RIDE instance where the optimal plan is not a low-coordination plan. The workers indicated by black rectangles all have a car with capacity 4 and all the edges have cost 1. The 4 workers to the right and the 2 workers to the left at the top have to coordinate to obtain an optimal plan with cost 6. The 2 workers at the bottom can be picked up by the 6 workers at the top if one of the 4 workers to the right at the top changes car after driving along the first edge.

**Definition 3.** *An instance of the MIN-DOM-SET-CUBIC problem is a cubic unweighted undirected graph $G(V, E)$ and the solution is a dominating set $V' \subset V$ with minimum cardinality (the cost of a feasible solution $V'$ is $|V'|$).*

Alimonti and Kann [1] have shown that the MIN-DOM-SET-CUBIC problem is APX-complete.

The main part of the remainder of this paper will be dedicated to showing MIN-DOM-SET-CUBIC $\leq_L$ PRIVATE-PARK-AND-RIDE-UNW-4. We use the notation from Sec. 1.2 so MIN-DOM-SET-CUBIC and PRIVATE-PARK-AND-RIDE-UNW-4 will play the role of $\Pi$ and $\Pi'$, respectively. The first thing we do is to show how to transform an instance $I$ of MIN-DOM-SET-CUBIC into an instance $f(I) = I'$ of PRIVATE-PARK-AND-RIDE-UNW-4. For each vertex $u$ in $I$ we add vertices $u'$ and $u''$ to $I'$. We connect every $u'$-vertex with the corresponding $u''$-vertex. We also add a target vertex $t$ to $I'$ and connect $t$ to all the $u''$-vertices. For each edge $\{u, v\}$ in $I$ we insert two edges in $I'$: $\{u', v''\}$ and $\{v', u''\}$. Figure 4 illustrates how an edge is transformed by the function $f$. Finally, we place a worker with a car with capacity 4 at each $u'$-vertex. The reader might benefit from thinking of the $u''$-vertices as parking lots where the workers can meet and share one car for the final ride to $t$. The following lemma introduces the function $g$ and expresses the crucial property the function has:

**Lemma 1.** *There is a polynomial time function $g$ that for any feasible solution $y$ to $I'$ produces a feasible solution $x$ to $I$ such that $cost(x) + n \leq cost'(y)$ where $n$ is the number of vertices in $I$.*

*Proof.* Now assume that we are given a feasible solution $y$ to $I'$. In other words, $y$ is a plan containing a sequence of steps bringing all the workers to $t$. For each worker $w$ we let $h(w)$ denote the vertex $u''$ that according to the plan $y$ receives

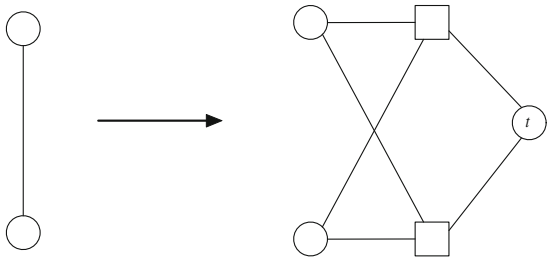



**Fig. 4.** The figure shows how an edge in $I$ (on the left) is processed by the reduction. The circles and squares on the right correspond to $u'$-vertices and $u''$-vertices, respectively.

the first visit of $w$ after $w$ leaving $loc(w)$. We now define $U''$ in the following way: $U'' = \{h(w) : w \in W\}$. Please note that $w$ may have been picked up by another worker at $loc(w)$ and made the first visit to $h(w)$ as a passenger. We now define $x$ as the set of vertices in $I$ corresponding to $U''$. The set $x$ is a dominating set and thus a feasible solution for $I$ – if a vertex $v$ is not in the set $x$ then one of the $u''$-vertices of the neighbours of $v$ must have received the first visit of the worker located at $v'$. We now make a new and cheaper plan $y^*$ by letting all workers $w$ go *by their own car* to the vertex $h(w)$. When all workers have moved to their corresponding vertex in $U''$ the plan $y^*$ dictates them to share *one* car and travel directly to $t$ – it is possible to share one car since the graph defining $I$ is cubic and all cars have capacity 4. It is not hard to see that $x$ can be produced in polynomial time and that the following holds

$$n + cost(x) = cost'(y^*) \leq cost'(y) \ .$$

□

**Lemma 2.** *PRIVATE-PARK-AND-RIDE-UNW-4* $\in$ *APX.*

*Proof.* There is a simple polynomial time 4-approximation algorithm for PRIVATE-PARK-AND-RIDE-UNW-4: Let all workers drive in their own car along the shortest path from their location to $t$. This plan has total cost $\sum_{w \in W} d(loc(w), t)$ where $d(u, v)$ is the minimum distance from $u$ to $v$ in the graph. If we imagine that all workers (including the driver) in a car share the cost of driving along an edge evenly then we can see that $\sum_{w \in W} \frac{d(loc(w),t)}{4}$ is a lower bound for the minimum achievable total cost of a plan since all workers pay the minimum possible amount. □

We are now ready to state and prove our main result:

**Theorem 3.** *The PRIVATE-PARK-AND-RIDE-UNW-4 problem is APX-complete.*

*Proof.* Membership of APX is already established by Lemma 2. We now prove that the functions $f$ and $g$ presented above can be used to prove that MIN-DOM-SET-CUBIC $\leq_L$ PRIVATE-PARK-AND-RIDE-UNW-4.

From Lemma 1 we conclude that $OPT(I)+n \leq OPT'(I')$. On the other hand, there is a straightforward plan for $I'$ with cost $OPT(I)+n$ and we conclude the following:

$$OPT'(I') = OPT(I) + n \ . \tag{1}$$

The graph in $I$ is cubic implying $OPT(I) \geq \frac{n}{4}$ since a vertex cannot dominate more than 3 vertices. From (1) we now see that

$$OPT'(I') \leq 5OPT(I) \tag{2}$$

showing that condition 1 for $L$-reductions from Sec. 1.2 is satisfied with $\alpha = 5$.

From Lemma 1 and (1) we finally conclude that

$$cost(x) - OPT(I) \leq cost'(y) - OPT'(I') \tag{3}$$

showing that condition 2 for $L$-reductions from Sec. 1.2 is satisfied with $\beta = 1$. This shows that $f$ and $g$ realize a $L$-reduction and we conclude that PRIVATE-PARK-AND-RIDE-UNW-4 is APX-complete since this is the case for MIN-DOM-SET-CUBIC. □

The result stated by Theorem 3 also holds restricted to low-coordination plans.

Finally, we mention one direction for future work. As mentioned in Sec. 1.1 the problem of computing an optimal park-and-ride-plan generalizes the Steiner tree problem. It thus seems natural to consider the possibility of generalizing the well known Dreyfus-Wagner algorithm [9] for computing minimum weight Steiner trees. One of the challenges – at least for the author — for doing this is the existence of optimal plans requiring much coordination.

## References

1. Alimonti, P., Kann, V.: Some APX-completeness results for cubic graphs. Theoretical Computer Science 237(12), 123–134 (2000)
2. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and hardness of approximation problems. In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, pp. 14–23 (1992)
3. Ausiello, G., Protasi, M., Marchetti-Spaccamela, A., Gambosi, G., Crescenzi, P., Kann, V.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, New York (1999)
4. Baldacci, R., Maniezzo, V., Mingozzi, A.: An exact method for the car pooling problem based on lagrangean column generation. Operations Research 52(3), 422–439 (2004)
5. Bern, M., Plassmann, P.: The Steiner problem with edge lengths 1 and 2. Information Processing Letters 32(4), 171–176 (1989)
6. Charikar, M., Chekuri, C., Cheung, T.-Y., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation algorithms for directed Steiner problems. Journal of Algorithms 33(1), 73–91 (1999)
7. Chvatal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)

8. Crescenzi, P.: A short guide to approximation preserving reductions. In: Proceedings of the 12th Annual IEEE Conference on Computational Complexity, CCC 1997, pp. 262–273. IEEE Computer Society (1997)
9. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks 1(3), 195–207 (1971)
10. Helmert, M.: On the complexity of planning in transportation domains. In: Proceedings of ECP 2001, pp. 349–360 (2001)
11. Helmert, M., Mattmüller, R., Röger, G.: Approximation properties of planning benchmarks. In: Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence, pp. 585–589. IOS Press (2006)
12. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43(3), 425–440 (1991)
13. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems (part I). Journal für Betriebswirtschaft 58(1), 21–51 (2008)
14. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems (part II). Journal für Betriebswirtschaft 58(2), 81–117 (2008)

# Predicting Road Accidents Based on Current and Historical Spatio-temporal Traffic Flow Data

Rupa Jagannathan[1], Sanja Petrovic[1], Gavin Powell[2], and Matthew Roberts[2]

[1] Division of Operations Management and Information Systems, Nottingham University Business School, Nottingham, UK
{rupa.jagannathan,sanja.petrovic}@nottingham.ac.uk
[2] Innovation Works, EADS, UK
{gavin.Powell,matthew.roberts}@eads.com

**Abstract.** This paper presents research work towards a novel decision support system that predicts in real time when current traffic flow conditions, measured by induction loop sensors, could cause road accidents. If flow conditions that make an accident more likely can be reliably predicted in real time, it would be possible to use this information to take preventive measures, such as changing variable speed limits before an accident happens. The system uses case-based reasoning, an artificial intelligence methodology, which predicts the outcome of current traffic flow conditions based on historical flow data cases that led to accidents. This study focusses on investigating if case-based reasoning using spatio-temporal flow data is a viable method to differentiate between accidents and non-accidents by evaluating the capability of the retrieval mechanism, the first stage in a case-based reasoning system, to retrieve a traffic flow case from the case base with the same outcome as the target case. Preliminary results from experiments using real-world spatio-temporal traffic flow data and accident data are promising.

**Keywords:** Traffic flow, road accidents, spatio-temporal data, case-based reasoning.

## 1 Introduction

The aim of this research work is to predict when current traffic conditions make the occurrence of road accidents more likely. If traffic conditions can be analysed in real-time and assessed for their safety with respect to road accidents, preventive measures can be taken, such as changing variable speed limits, to avoid accidents. According to the Department of Transport, out of 151474 road accidents in the UK in 2011, 75708 accidents happened on major roads, i.e. motorways and 'A' roads [1]. Apart from the human loss, the total national financial cost of road accidents has been estimated to amount up to 2.5% of the gross national product [2].

There are many factors that can influence the risk of road accidents such as human error, weather and visibility conditions, vehicle conditions, etc. Another major factor is traffic flow. Traffic flow refers to interactions between vehicles, drivers, and

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 83–97, 2013.

infrastructure and includes information such as the speed of vehicles and the number of vehicles in a given amount of time on a given length of road. Traffic flow parameters are measured using induction loop sensors that are embedded at regular distances along many major roads.

In this research work, we describe a case-based reasoning (CBR) system that analyses current traffic flow data obtained from road sensors and makes a prediction regarding accidents by comparing current flow data with historical flow data that preceded an accident in the past. CBR is an artificial intelligence methodology that solves problems based on similar problems in the past [3]. The advantage of CBR is that it does not depend on patterns that are analysed or defined a priori and can be used to infer solutions, when problem domains are not well understood or are difficult to describe using mathematical formulations or rules. Since CBR does not depend on well-defined rules, interactive and derived factors are inherently taken into account. Traffic flow patterns, such as congestion or traffic waves, which make accidents more likely, are indirectly considered in the flow data contained in the archived accident cases. Further, as the sensors record flow data at regular intervals of time and distance, the data is spatio-temporal in nature. So far, most traffic flow forecasting methods described in the literature considered only sensor data obtained from a single sensor over an interval of time (temporal data) [4-7].

The novelty of the research work presented lies in the application of CBR to identify potentially hazardous traffic flow conditions based on past similar traffic flow conditions considering not only temporal but both spatial and temporal flow data.

The first step in determining the feasibility of using CBR to predict when traffic flow conditions might cause an accident is to assess if a CBR system is capable of differentiating between traffic flow data that was followed by an accident and generic traffic flow data that was not followed by an accident. The experiments presented in this paper focus on this aspect.

The paper is organised as follows. A brief overview of the background to the problem domain and relevant literature is given in Section 2. The setup of the system to investigate the ability of CBR to differentiate between accident and non-accident traffic flow data is presented in Section 3. Section 4 describes experiments and presents preliminary results. The salient points of our research so far are summarized in Section 5, along with future research directions.

## 2 Background and Related Work

This section briefly introduces the fundamental background knowledge required for the comprehension of the research work. Previous and related relevant work from the literature is included. The induction loop sensors that measure traffic flow are described followed by a brief introduction to case-based reasoning.

### 2.1 Traffic Flow Data and Measurement

Traffic flow data is measured by induction loop sensors embedded in roads at regular intervals, which form part of a distributed network called the Motorway Incident

Detection and Automatic Signalling (MIDAS) system, which displays real-time traffic information, designed to set variable message signs and advisory speed limits with little human intervention. The sensors have been installed on major motorways and some 'A' roads in the UK by the Highways Agency to monitor traffic conditions and minimise congestion. The sensors are placed every 500m along a road and measure in intervals of 60 seconds the number of vehicles on each lane (also known as flow), their average speed, average headway (distance between vehicles with respect to time) and the occupancy or the amount of time the vehicles spent on the sensing loop. This project considers parts of the M4, A470 and M48 in the South of Wales.

Intuitively, traffic flow variables such as the number of vehicles on the road, their speed and the distance between vehicles influence the risk of an accident happening. Garber and Ehrhart [8] have found a relationship between the accident rate on two-lane highways and the speed, flow and lane width. Golob et al. [4] investigated the relationship between traffic flow, weather and lighting conditions and the accident rate on urban freeways in California and have shown that in congested traffic flow, both the risk of accident and its cost are correlated to the traffic speed and density.

Research has been carried out to use the traffic flow conditions prior to an accident to predict the probability or risk of an accident happening in real-time. Golob et al. [5] have identified temporal flow and speed patterns that are indicative of conditions that could result in an accident and can be used to predict accidents in real time. Cheol et al. [9] used the mean and standard deviation of flow attribute data over 5 minutes before an accident to predict in real-time when current traffic conditions become hazardous using a Bayesian modelling approach. A study by Lee and Abdel-Aty [7] also used 5 min of traffic flow data before an accident to identify the traffic conditions contributing to crashes on freeway ramps in Florida. However, most work done so far focussed on temporal data rather than spatio-temporal data, which represents an added level of complexity. Spatio-temporal data, in which data vary in both space and time, from sensors have been used in a variety of applications including scientific domains, including weather prediction and climate modeling, earth systems science, biomedicine, and materials science [10, 11]. According to Treiber et al. [12] highway traffic, cannot be adequately understood by time-series data alone but should be described instead by considering data both with respect to time and space. In our work, we use both temporal and spatial flow data to predict accidents based on historical spatio-temporal flow data using case-based reasoning.

### 2.2 Case-Based Reasoning

In case-based reasoning (CBR), problems are solved based on the solutions of similar past problems [3]. The case base consists of past cases, which are stored along with their solution. Given a new case or target case, the CBR system calculates the similarity between the new case and each case in the case base and then retrieves the most similar case. The similarity measure, which calculates the similarity between the target case and the archived cases, is fundamental to the working of a CBR system. The popular *K*-nearest neighbour method (KNN) [13] matches each attribute in the target case to its corresponding attribute in the archive case. Attributes relevant to the

problem are commonly identified and weighted either using domain experts or automated feature selection algorithms [14-16].

Spatio-temporal data has not been widely used in CBR so far. Likhachev and Arkin [17] developed a spatio-temporal case-based reasoning system for behavioural selection of robots in run-time. In this work, the similarity between cases is computed by calculating the distance between the traffic flow attributes of two cases at every sensor over a fixed time interval and along the given distance.

## 3 The Case-Based Reasoning System

This section discusses the input data and the architecture of the CBR system.

### 3.1 Data

The input data to the CBR system was supplied by the Welsh Transport Technology Consultancy (WTTC), which manages various aspects of traffic, including the MIDAS sensors in Wales.

**MIDAS Traffic Flow Data.** The MIDAS flow data consist of the measurements made by each sensor. The schematic diagram in Fig. 1 shows the arrangement of sensors in time and space prior to an accident, where *n* and *m are* the spatial and temporal indices of each sensor, respectively. The rows represent the length of distance and the columns represent the time interval under consideration. For instance, the top right most sensor $d_n t_m$ measures traffic flow data just prior to the time when an accident occurred and is spatially the nearest sensor to the accident location. The MIDAS traffic data is stored in MySQL and accessed using MySQL Workbench (version 5.2.36). MySQL is an open source relational database management system.

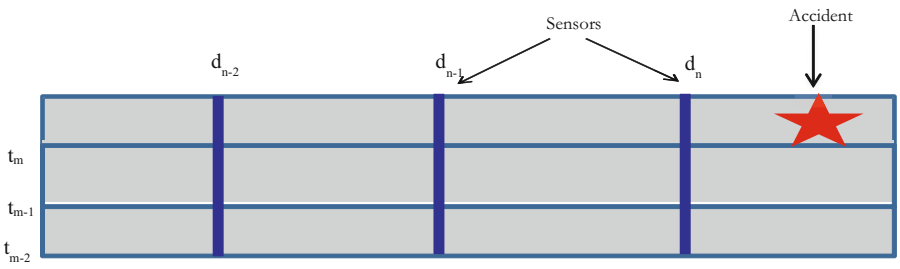



**Fig. 1.** Schematic diagram showing spatio-temporal sensor layout

Besides information that specifies the location of the sensor and the time of each measurement, the MIDAS sensors measure the following parameters:

- Sensor Identification Code: This code uniquely identifies a sensor on a lane.
- Average Speed: Arithmetic mean of vehicle speeds in a lane in a 60s time interval.

- Flow Category 1-4: Type of vehicle based on their length. Categories 1 and 2 represent light vehicles, while categories 3 and 4 represent heavy good vehicles.
- Average Headway: Arithmetic mean of the distance between vehicles measured with a precision of 0.1s.
- Occupancy: Ratio of the total detection time to the maximum detection time, expressed as a percentage, where the detection time refers to the amount of time in which the sensor detects the presence of a vehicle.

**Accident Data.** The WTTC also keeps a record of incidents such as accidents, congestion, road works, etc. We use the accident dataset, which contains information that specifies the location, time and type of every accident.

### 3.2 Architecture of the Experimental Setup

The main requirement of an automated traffic accident forecasting system is to identify when flow data indicates a likelihood of causing an accident. In order to study the feasibility of using CBR for this purpose, the capability of the CBR system to differentiate between historical flow data that has led to an accident and historical flow data that was not followed by an accident is investigated.

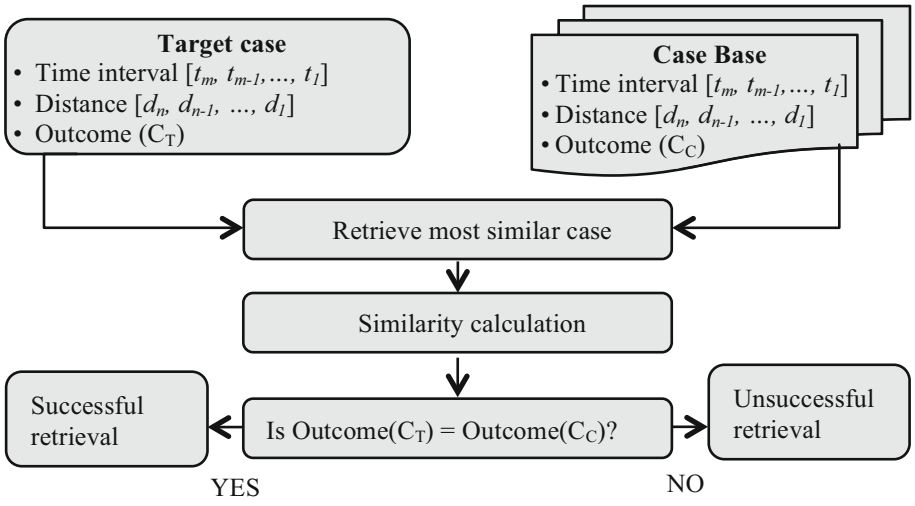



**Fig. 2.** Schematic diagram showing steps involved to evaluate the differentiation capability of the CBR system

Fig. 2 shows a schematic overview of the CBR setup to determine if the system is capable of differentiating between accidents and non-accidents. The current flow data to be analysed for its accident risk constitutes the target case. The flow data is considered over a pre-specified interval of time [$t_m$, $t_1$] and distance [$d_n$,$d_1$], where $m$ and $n$ refer to the point or index of a sensor with respect to time or distance, respectively.

$t_m$ and $d_n$ represent the sensor indices just prior to the point at which the accident risk is to be assessed, whereas $t_1$ and $d_1$ represent the first sensor from which onward sensor measurements are considered with respect to time and distance, respectively. The case base consists of historical accident cases and non-accident cases along with their corresponding flow data. The flow data in the target case is compared to the flow data corresponding to each accident case and their similarity is calculated. The most similar case is retrieved. If the outcome of the retrieved case, Outcome ($C_C$), i.e. either accident or non-accident case, is the same as the outcome of the target case, Outcome ($C_T$), the retrieval is deemed as successful. This procedure is repeated for all accident and non-accident cases as target cases. The average success rate gives an indication of the capability of the system to differentiate between accident and non-accident cases.

### 3.3 Case Attributes

The knowledge contained in a CBR system is represented in the form of cases. The case attributes describe a case using information that is relevant to the solution of the CBR system. In the CBR system under development the cases comprise the traffic flow data over a certain period of time and a certain distance.

The target case contains the current traffic flow data or the flow data for which we want to determine the accident risk. The cases in the case base represent historical accidents and the traffic flow conditions in which they occurred or non-accident cases. The solution of a case is the classification, i.e., has an accident occurred or not. The case attributes are stored in the case base in the form of key value pairs. The attributes can be divided into four groups:

1. Nominal attributes

These attributes take categorical values. They are determined for the entire spatio-temporal period of flow in consideration. They include:

- Time day category: The time of day attempts to capture the traffic state and therefore congestion that can be expected. This attribute can take six nominal values:

— '1' – represents the morning rush hour on a weekday and has been empirically set between 6:00h and 10:00h from Monday to Friday
— '2' – represents normal traffic flow between 10:00h and 16:00h on weekdays from Monday to Friday.
— '3' – represents the evening rush hour traffic state between 16:00h and 19:00h from Monday to Friday
— '4' – represents night time traffic between 19:00h and 6:00h.
— '5' – represents weekend (Saturday/Sunday) day time traffic between 6:00h and 19:00h.
— '6' – represents weekend night time traffic between 19:00h and 6:00h.

- Time-interval/Distance, *T/L*: refers to the period of time *T* or distance *L* over which flow data is considered. It is currently an empirically set design choice.

It ensures that the time period or distance considered are equal in both the target case and the cases in the case base in order to facilitate similarity calculation.

2. Single point attributes

The single point attributes represent the flow data measured by a single sensor. These attributes are calculated for each sensor within time interval $T$ and over a length of distance $L$. In the target case, the starting sensor is at a location and time for which the accident risk is to be analysed, the current sensor. For the cases in the case base, usually the starting sensor immediately preceding the location (or time) of an accident is chosen. The single point attributes, derived from the sensor measurements, include:

- Average number of vehicles (flow) over all lanes, $f$
- Average velocity of vehicles over all lanes, $s$
- Average headway between vehicles over all lanes, $h$
- Average occupancy measured by the sensor, $o$
- Standard deviation in average number of vehicles per lane (flow) between lanes, $f_d$
- Standard deviation in average velocity of vehicles between lanes, $s_d$
- Standard deviation in average headway of vehicles between all lanes, $h_d$
- Standard deviation in occupancy of vehicles between all lanes, $o_d$

3. Temporal attributes

These attributes describe the variation in the attributes of traffic flow data over the specified time period $T$ [5]. The temporal variations are determined by calculating the ratio of the difference between the 90th and 50th percentile of the traffic flow attribute to the maximum value of the attribute found in all lanes. They include the variation in average speed and flow over all lanes. The variation of the other two attributes (headway and occupancy) was not considered to be relevant.

- Variation in average speed over all lanes $V_{T,s}$

$$V_{T,s} = \frac{s_{90} - s_{50}}{s_{max}} \tag{1}$$

where $s_{90}$, $s_{50}$ and $s_{max}$ are the 90$^{th}$ percentile, 50$^{th}$ percentile and maximum average speed found over all lanes, respectively, during the time period $T$.

- Variation in average flow over all lanes

$$V_{T,f} = \frac{f_{90} - f_{50}}{f_{max}} \tag{2}$$

where $f_{90}$, $f_{50}$ and $f_{max}$ are the 90$^{th}$ percentile, 50$^{th}$ percentile and maximum average number of vehicles found over all lanes, respectively, during $T$.

4. Spatial attributes

These attributes describe the variation in the attributes of traffic flow data over distance *L.* The spatial variations are determined by calculating the net increase or decrease between the starting point of the stretch of road considered and the end (or current) point of the traffic flow attributes. They include:

- Variation in average velocity over all lanes $V_{L,s}$

$$V_{L,s} = s_{L,Start} - s_{L,End} \tag{3}$$

where $s_{L,Start}$ and $s_{L,End}$ are the average speed of vehicles over all lanes found at the starting and end point, respectively, of the distance $L$ considered.

- Variation in average flow over all lanes $V_{L,f}$

$$V_{L,f} = f_{L,Start} - f_{L,End} \tag{4}$$

where $f_{L,Start}$ and $f_{L,End}$ are the average number of vehicles over all lanes found at the starting and end point respectively of the distance $L$ considered.

### 3.4 Retrieval and Similarity Measure

The retrieval mechanism constitutes the inference engine of the CBR system. Given a target case, the system identifies the most similar case in the case base to the target case and retrieves it. The main component of the retrieval mechanism is the similarity measure, which calculates the similarity between the target case and the cases in the case base. Since CBR systems are based on the notion of similar cases having similar solutions, the definition of similarity is crucial. The CBR system depends on a good retrieval engine that is capable of retrieving cases whose solution is relevant to the target case.

The similarity in our CBR system is computed using the *K*-nearest neighbour algorithm (KNN), where *K* refers to the number of neighbours. The KNN algorithm, traditionally used in classification and pattern recognition problems to assign objects to classes, has been widely used in CBR as well, owing to its ease of implementation and the fact that it does not make any assumptions about the distribution of the underlying data. In KNN classification, an object is classified by assigning it to the known class of its nearest examples (or neighbours) in the solution space. The solution space in CBR can be viewed as a collection of clusters, where each cluster contains similar solutions [18]. The nearest neighbour is found using a distance metric defined for each type of attribute. Currently, we consider four combinations of attribute similarities in the similarity measure

- Time – Day Category Similarity, $S_{TD}$:

If the target case and a case from the case base fall into the same time-day category, the similarity $S_{TD}$ is '1', else it is '0'.

- Similarity based on spatio-temporal single point attributes, $S_{SP}$

The spatio-temporal similarity is calculated based on all single point attribute similarity values, i.e., the average of the flow ($f$), speed ($s$), headway ($h$) and occupancy ($o$) on all lanes and the standard deviation of the flow ($f_d$), speed ($s_d$), headway ($h_d$) and occupancy ($o_d$) between the lanes. These eight attributes are determined at each point along the distance considered and the given time interval as shown in Fig. 3. The attribute values are normalised between the interval [0,1] based on the range of values of each attribute found in the available flow data. The distance between corresponding points between the target case and the case from the case base is given by:

$$\boldsymbol{D_{SP}} = \frac{1}{8}\left(\sum_{l}\left|\boldsymbol{x_{T,l}} - \boldsymbol{x_{C,l}}\right|^{2}\right)^{1/2} \tag{5}$$

where $x_{T,l}$ is the normalised value of attribute $l$= $k$, $s$, $h$, $o$, $k_d$, $s_d$, $h_d$, $o_d$ in the target case and $x_{C,l}$ is the normalised value of attribute $l$ in the case from the case base.

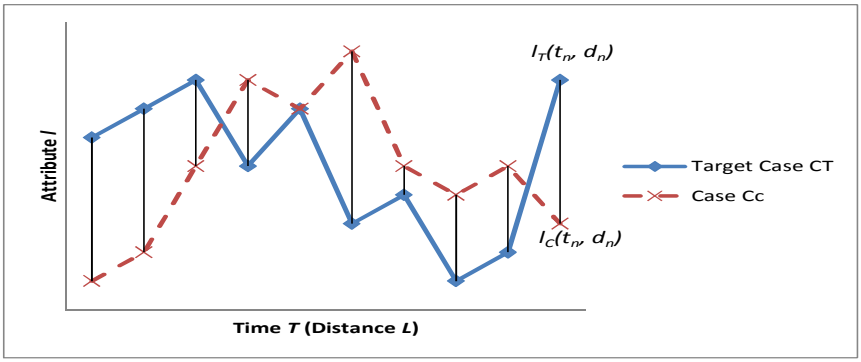



**Fig. 3.** Graph showing values $L_T(t_n,d_n)$ and $l_C(t_n,d_n)$ of attribute $l$ either over time $T$ or distance $L$ for target case $C_T$ and a case from the case base $C_C$. The lines connecting corresponding points in the attribute value curves represent the difference in the attribute values with respect to time $T$ or distance $L$.

The aggregate distance $D_{avg}$ is obtained by taking the mean of the difference between individual point values of the attributes of the target case and a case from the case base in space and time. The similarity $S_{SP}$ is given by

$$S_{SP} = 1 - D_{avg} \tag{6}$$

- Similarity based on variation of flow over distance or time interval, $S_{var}$

This value refers to the difference between the variation of attribute values flow ($f$) and speed ($s$) over a distance $L$ or a time interval $T$ as given in expressions (1) and (2). The difference in net variation between two cases over a time interval constitutes the temporal difference $D_{temp}$ and the difference in net change between two cases over a distance constitutes the spatial difference $D_{spatial}$. The similarity $S_{var}$ is given by

$$S_{var} = 1 - \left[ \left( \frac{\left( \sum_l \left| x_{l,T,temp} - x_{l,C,temp} \right|^2 \right)^{1/2}}{norm_{l,temp}} + \frac{\left( \sum_l \left| x_{l,T,spatial} - x_{l,C,spatial} \right|^2 \right)^{1/2}}{norm_{l,spatial}} \right) \right] \tag{7}$$

where $x_{l,T,temp}$ is the attribute net temporal variation of the target case $T$ with respect to attribute $l$, $l$=$k,s$.and $x_{l,C,temp}$ is the attribute net temporal variation of case $C$, $x_{l,T,spatial}$ is the attribute net spatial variation of the target case $T$ with respect to attribute $l$ and $x_{l,C,spatial}$ is attribute net spatial variation of case $C$. The normalisation factor is given by $norm_{l,,temp}$ and $norm_{l,spatial}$.

- Aggregate similarity

The aggregate similarity $S_{All}$ is given by the average of the sum of the individual similarity values between the target case and a case from the case base.

$$S_{All} = \frac{(S_{TD} + S_{SP} + S_{var})}{3} \tag{8}$$

## 4 Validation of the CBR Methodology

The aim of this study is to investigate the feasibility of using CBR to predict the likelihood of an accident happening under certain traffic conditions. In order to be able to do this, we need to ascertain that the flow data contains certain properties that are indicative of accidents. In other words, there have to be flow data properties that allow us to differentiate between accident flow data and non-accident flow data. The basic premise of CBR is that similar cases have similar solutions. In our problem, we can interpret the premise as similar cases having similar outcomes in terms of accidents. The question is if given a target case of flow data attributes, does the most similar case to the target case retrieved from the case base, have an outcome that can be expected for the target case as well? Currently, the only input data available to the system is the record of accidents and the traffic flow data for both accidents and non-accidents. However, it is still uncertain if by evaluating exclusively the flow data online using CBR it is possible to predict accidents.

The success rate of a CBR system can be quantified in terms of the number of successful retrievals from a sample of test cases (i.e., cases in which the solution to be determined is known). Retrieval is deemed successful if the solution or outcome of the retrieved case, with the highest similarity to the target case, is considered correct. As we know the outcome of the test cases, i.e., which cases constitute accident cases and which cases constitute non-accident cases, a successful retrieval is Outcome ($C_T$) = Outcome ($C_C$), i.e.:

- The target case is an accident case. The retrieved case is an accident case.
- The target case is a non-accident case. The retrieved case is a non-accident case.

### 4.1 Experimental Setup

The available test data comprises the traffic flow data and accident data for accidents. The case base contains 71 historical flow data cases, for which the outcome is known, i.e., the flow data in a case was either followed by an accident or it was not. Creating accident cases was relatively straight forward by using the incident data obtained from the WTTC for the year 2010. Currently, there are 39 accident cases, with sufficient information to correlate them to corresponding flow data. The non-accident cases had to be artificially generated by selecting for each non accident data from the existing flow data over the same time $T$ and distance $L$. To take into consideration differences arising from road locations or timing, we have chosen the non-accident data at the same locations and same date and time at which the accidents considered occurred but in a different year. In other words, currently the non-accident data consists of the locations and relevant flow data taken from the accident data (2010) but in a different year (2008). The time $T$ and distance $L$ prior to an accident (or non-accident) are empirically chosen to be $T$ = 10 minutes and $L$ = 1km.

For assessing the prediction capability of each of the possible models, leave-one-out cross validation (LOOCV) technique was chosen. In LOOCV, one case of the test set (the target case) is chosen in each iteration, and the remaining cases of the test set with the exception of the chosen case is used as the case base.

For each target case, the retrieval mechanism is run, the similarity between the target case and the cases in the case base is computed and the most similar case (K=1) or the three most similar cases (K=3) are retrieved. For each retrieval, we determine if the most similar retrieved case (or cases) is of the same type (accident or non-accident) as the target case. When $K$=3, i.e., the three most similar cases to the target case are retrieved, the mode of the three retrieved case solutions is used. For instance, if for a given target case at least two cases out of the three retrieved ones are of the type 'accident', then the prediction for the target case is 'accident'. Each case in the case base is consecutively made the target case with the remaining cases constituting the case base. A retrieval is considered to be successful if the outcome of the retrieved case or cases, is the same as the known outcome of the target case, i.e. either "accident" or "non-accident".

The average success rate of the retrieval mechanism is computed using

$$Sucess\ Rate = \frac{r_c * 100}{n} \tag{9}$$

where $r_c$ is the number of correct retrievals for the target cases and $n$ is the total number of cases in the case base.

### 4.2 Results

The experiment was carried out first by retrieving the top most similar case to the target case (K=1) and then by retrieving the three most similar cases to the target case

(K=3). When K=3, the mode of the outcomes of the three retrieved cases is considered. This was repeated for each similarity combination, $S_{TD}$, $S_{SP}$ and $S_{var}$. The number of correct retrievals and the success rate with respect to the different similarity combinations, $S_{TD}$, $S_{SP}$ and $S_{var}$ are shown in Table 1 for both situations, i.e., when the target cases consist of known accident cases and when the target cases consist of known non-accident cases, generated as explained above. The case base consists of both accident and non-accident cases.

**Table 1.** Average number of correct retrievals and success rate of accident and non-accident cases

| | $S_{TD}$ | | $S_{SP}$ | | $S_{var}$ | | $S_{All}$ | |
|---|---|---|---|---|---|---|---|---|
| | K=1 | K=3 | K=1 | K=3 | K=1 | K=3 | K=1 | K=3 |
| Target cases = Accident cases | | | | | | | | |
| Number of correct retrievals | 0 | 0 | 25 | 30 | 21 | 24 | 28 | 27 |
| Success rate | 0 | 0 | 64% | 77% | 54% | 62% | 72% | 69% |
| Target cases = Non-accident cases | | | | | | | | |
| Number of correct retrievals | 0 | 0 | 25 | 27 | 26 | 26 | 20 | 23 |
| Success rate | 0 | 0 | 78% | 84% | 81% | 81% | 63% | 72% |
| **Total success rate** | **0** | **0** | **71%** | **81%** | **68%** | **71%** | **68%** | **71%** |

As can be seen from the table, the best retrieval success rate is obtained when the spatio-temporal, single-point similarity combination $S_{SP}$ is used with K=3. As expected, computing the similarity based on spatio-temporal attribute differences has the strongest prediction power for both accident and non-accident cases as target cases. Counter intuitively, it was found that the time-day category attribute is not capable of differentiating between accidents and non-accidents and on its own is not sufficient to retrieve a case with the same outcome as the test cases. It is possible that the time and day information is already included in the flow data. For instance, during rush hour we assume that the flow of vehicles is large and the speed is small. Also, it is difficult to define accurately the time-day categories, which are currently arbitrarily defined. The success rate improves in general when three cases are retrieved and the mode of the retrieved cases' outcomes is used as the solution. By retrieving more than one case, we can reduce the risk of retrieving an uncharacteristic case that happens to have a large similarity with the target case. The retrieval success rates obtained with $S_{SP}$, in particular, look promising and can be considered to be a good starting point for further research.

## 5 Conclusion and Future Work

The relatively high success rate of 81% shows that the CBR system under development is capable of differentiating between accident and non-accident cases based on

spatio-temporal flow data attributes. In order to confirm the results obtained with this data set, the next step of this research will evaluate the system by using a larger number of both accident and non-accident cases, and also by varying the percentage of accident/non accident cases in the case base. Currently, the case base consists of 39 accident and 32 non-accident cases, so for each case, the probability that retrieval is successful is 54% for accident cases and 44% for non-accident cases. Increasing the percentage of non-accident cases in the case base might reduce the success rate, however considering that currently the success rate for both accident and non-accident retrievals is comparable, we expect that the system performance is relatively stable to changes in the contents of the case base. However, this will be confirmed using experiments. Currently, most attributes describe the traffic flow. However, it is likely that accidents occur also due to other conditions, such as the weather, the light conditions or visibility and human behaviour. We will use another accident data set consisting of police accident records, which contains a wealth of additional information, including weather conditions, types of accidents and the severity of an accident that could be important in determining accidents [19]. It is difficult to directly measure the contribution of individual human error resulting from their state of mind to accidents though without doubt they play a large role. However, human error is expressed partially in their driving and therefore in the flow data, for instance, speeding or insufficient distance between vehicles. The similarity between cases with respect to the normalised spatio-temporal flow attributes is computed using the Euclidian distance. We are currently working on implementing a more accurate similarity measure based on the Mahalanobis distance, which takes into account the correlation between attributes. Another important consideration is case attribute selection and weighting. Not all attributes are significant with respect to the desired solution of a target case and the contribution of each attribute to the similarity between cases can differ, too. A possible method of attribute weighting would be to use local weights, which vary based on the attribute values in the target case and take into account the relation between attributes [20]. Currently, the spatio-temporal attributes are computed by matching corresponding points in time and space. However, often spatio-temporal patterns can be similar even if one pattern is stretched, contracted or of a different length than the other pattern. This is possible using dynamic time warping [21]. It would also be interesting to evaluate how the accident prediction system using CBR and spatio-temporal data compares to other accident prediction systems in the literature [5, 9].

The next stage in the system design involves finding the threshold above which cases similar to the target case are deemed to be good predictors of the outcome of the target case. Given a target case, if the similarity between the retrieved case and the target case is larger than the threshold, the CBR system will predict that the outcome of the target case is similar to the outcome of the retrieved case. The advantage of CBR is that it inherently detects traffic flow patterns that are indicative of accidents. However, the accident cases in the case base (and their similarity) can be studied to identify a set of conditions that might lead to accidents. These could be used by traffic controllers or designers a priori to reduce the likelihood of accidents. The aim of the final system is to automate this process so that flow data is analysed continuously

online and feedback about the accident risk is fed back to the controller in real time so that preventive action can be taken. Therefore, finally, an interface will be created to feedback the prediction of the outcome of current traffic flow data in real time to an agency, such as the Welsh Traffic Technology Consultancy or to an automated system to accordingly take anticipative actions, such as setting variable speed limits. An interesting extension of this system would be to predict traffic flow based on actions such as changing speed limits and then re-analysing the risk of an accident occurring. This can be done again using CBR by comparing the changes in flow data due to anticipative actions to similar changes in the past and their outcomes.

**Acknowledgements.** This work was funded by an EPSRC Knowledge Transfer Secondment programme allocated to the University of Nottingham. The work was done in collaboration with Innovation Works, EADS**.** Valuable guidance and traffic data was provided by the Welsh Traffic Technology Consultancy (WTTC), Wales.

## References

1. https://www.gov.uk/government/statistical-data-sets/ras10-reported-road-accidents
2. Elvik, R.: How much do road accidents cost the national economy? Accident Analysis & Prevention 32, 849–851 (2000)
3. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, San Francisco (1993)
4. Golob, T.F., Recker, W.W.: Relationships among urban freeway accidents, traffic flow, weather, and lighting conditions. Journal of Transportation Engineering 129, 342–353 (2003)
5. Golob, T.F., Recker, W.W., Alvarez, V.M.: Freeway safety as a function of traffic flow. Accident Analysis & Prevention 36, 933–946 (2004)
6. Taehyung, K., Hyoungsoo, K., Lovell, D.J.: Traffic flow forecasting: overcoming memoryless property in nearest neighbor non-parametric regression. In: Proceedings of the Intelligent Transportation Systems, pp. 965–969. IEEE (2005)
7. Lee, C., Abdel-Aty, M.A.: Two-level nested Logit model to identify traffic flow parameters affecting crash occurrence on freeway ramps. Transportation Research Record 2083, 145–152 (2008)
8. Garber, N., Ehrhart, A.: Effect of speed, flow, and geometric characteristics on crash frequency for two-lane highways. Transportation Research Record: Journal of the Transportation Research Board 1717, 76–83 (2000)
9. Cheol, O., Jun-Seok, O., Ritchie, S.G.: Real-time hazardous traffic condition warning system: framework and evaluation. IEEE Transactions on Intelligent Transportation Systems 6, 265–272 (2005)
10. Saltz, J.H., Teodoro, G., Pan, T., Cooper, L.A.D., Kong, J., Klasky, S., Kurc, T.M.: Feature-based analysis of large-scale spatio-temporal sensor data on hybrid architectures. International Journal of High Performance Computing Applications (2013)
11. George, B., Kim, S.: Spatio-temporal networks: An introduction. In: Spatio-temporal Networks, pp. 1–6. Springer, New York (2013)
12. Treiber, M., Kesting, A.: Validation of traffic flow models with respect to the spatiotemporal evolution of congested traffic patterns. Transportation Research Part C: Emerging Technologies 21, 31–41 (2012)

13. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7, 39–59 (1994)
14. Beddoe, G.R., Petrovic, S.: Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering. European Journal of Operational Research 175, 649–671 (2006)
15. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
16. Aha, D., Bankert, R.: A comparative evaluation of sequential feature selection algorithms. In: Fisher, D., Lenz, H.-J. (eds.) Learning from Data, vol. 112, pp. 199–206. Springer, New York (1996)
17. Likhachev, M., Arkin, R.C.: Spatio-temporal case-based reasoning for behavioral selection. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, vol. 1622, pp. 1627–1634 (2001)
18. Blanzieri, E., Ricci, F.: Probability based metrics for nearest neighbor classification and case-based reasoning. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 14–28. Springer, Heidelberg (1999)
19. http://data.gov.uk/dataset/road-accidents-safety-data
20. Jagannathan, R., Petrovic, S.: A local rule-based attribute weighting scheme for a case-based reasoning system for radiotherapy treatment planning. In: Agudo, B.D., Watson, I. (eds.) ICCBR 2012. LNCS, vol. 7466, pp. 167–181. Springer, Heidelberg (2012)
21. Wöllmer, M., Al-Hames, M., Eyben, F., Schuller, B., Rigoll, G.: A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams. Neurocomputing 73, 366–380 (2009)

# An Integrated Simulation and Business Intelligence Framework for Designing and Planning Demand Responsive Transport Systems

José Telhada, Ana Cecília Dias, Paulo Sampaio, Guilherme Pereira, and Maria Sameiro Carvalho

Centro Algoritmi, Escola de Engenharia, Universidade do Minho, Campus de Gualtar, 4710-057 Braga, Portugal
{jose.telhada,ana.dias,paulo.sampaio,guilherme.pereira, maria.carvalho}@algoritmi.uminho.pt

**Abstract.** Transportation planners, analysts and decision-makers usually make use of simulation tools in order to accurately predict the performance of new practices and policies before their implementation. Additionally, most of them, recently introduced new auxiliary Business Intelligence decision support tools in order to transform their available huge amount of real operating data into timely and accurate information for their decisions. However, although both of those types of automated decision support are valuable, very few attempts were already made by researchers to integrate them into a unified tool.

This paper proposes a general conceptual framework for a strategic and tactical decision support system that integrates both of these technologies, simulation and Business Intelligence, in order to enhance the overall practical interest for its usage by taking advantage of the foreseen synergy of the integration. This approach is then extended and illustrated to the case of demand responsive transport (DRT) systems. DRT systems provide transport on demand for users, using flexible schedules and routes to satisfy their travel needs.

The paper discusses the potential interest of the proposed integration, identifying a set of questions that can then be answered with effectiveness and more efficiently than before. In particular, such set comprises a list of strategic and tactical issues that are of crucial importance to the design and planning of sustainable DRT systems, before its implementation, thus overcoming the current lack of analytical tools to this end.

**Keywords:** Operations research, What-if simulation, Business intelligence, Integrated decision support system, Demand responsive transport (DRT).

## 1 Introduction

The design and planning of transportation systems are usually very difficult tasks due to the intrinsic high complexity of the interactions between the numerous stakeholders and physical elements involved. This is the main reason why simulation tools are widely used in such tasks, since they allow mimicking the functioning of systems,

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 98–112, 2013.

testing and evaluating the performance of different plans, prior to their actual execution. Nowadays, evaluation of different alternative solutions is worthy to identify the option(s) that are economic, social and environmental sustainable. In most cases, traditional analytical methods are unable to fully address these issues.

Business intelligence (BI) tools comprise a set of analytical, statistical and artificial intelligence techniques along with sophisticated graphical technologies enabling decision makers to transform their available (huge amount of) data into timely and accurate information for decision making. Therefore, simulation and BI tools can be viewed as complementary tools for supporting decision making. However, only a few attempts to make use of effective operations research (OR) technologies (e.g. simulation) along with BI capabilities of historical data analysis has been reported in the literature. For example, Sabbour et al. [1] use visual interactive simulations coordinated with BI to support decisions at the strategic level. They also present the essential top-level managerial requirements that would transform strategic decision simulation into an integral component of BI systems, in the field of supply chain management. This paper intends to do basically the same, i.e., to propose a conceptual framework for a decision support system (tool) integrating both technologies, simulation and BI, in a similar fashion as [1], but focused on a specific field of transportation (Demand Responsive Transport - DRT).[1] That is, this paper proposes an integrated decision support system (IDSS) targeted to support decision makers in designing and implementing a DRT system, by investigating the impacts of alternative specifications of such system on its viability and sustainability for a given area.

A literature review is provided in Section 2 on actual decision support tools, along with a systematization of the performance measures that are relevant to transportation systems, and therefore to be used as the output of the proposed advanced BI tool. Since BI is a relatively new concept for the OR community, Section 2 also introduces such concept, reports some examples of tools that are commercially available, and generally describes their main functionalities. The focus of this work is on the proposal of a comprehensive framework for a what-if simulation based BI decision support system, and this is the matter of Section 3, where particular emphasis is done on a specific proposal to tackle DRT design and planning modelling tasks. Section 4 discusses the relevance of this proposal, and Section 5 illustrates some forms of visualization to enhance the effective and efficient use of the data and information available. Finally, Section 6 summarizes the main conclusions of this work.

## 2 Literature Review

Existing simulation tools aim at strategic or tactical decisions, and rarely are used for operational decision making. In this sense, the key idea of simulation is to execute a model repeatedly, allowing the estimation of statistical measures, such as averages, standard deviations and their confidence intervals. As a result, analysts will foresee

[1] Basically, both approaches were developed independently, since the authors of this paper became aware about the referred work only in the final stage of completing this paper. Moreover, our paper is an extension of our own recent work [2-3].

the long-run (in most cases, the steady-state) behaviour of the system and its related set of performance measures, thus allowing to perform its evaluation, e.g. comparatively against a "competing" scenario for the best in the pool for sustainability.

Transportation systems are, in general, intrinsically complex in terms of designing, planning and analysis. For this reason, perhaps the majority of transportation planners and analysts (e.g., traffic and mobility management bodies and agents, consultants) already use simulation tools. There is a significant number of commercial solutions available, such as EMME2, AIMSUN, DYNASMART, VISUM, DRACULA, etc.

The main problem with such tools is that they are intended to be used by specialists since their outputs are usually very technical and do not integrate some other relevant data analyses (e.g., financial data). Therefore, further analyses are needed in order to reach to very specific and succinct relevant information that can deploy the decision making. In other words, these tools are devoid of such benefits (process wizards, clicks and drag-and-drops, and relevant-only information) that characterizes BI tools. Additionally, Intelligent Transportation Systems (ITS) integrate a broad range of IS/IT elements for gathering on-site data and transfer it to the central database(s) of the enterprises or organizations. ITS also integrate the required hardware and software to analyse and manage the information extracted from data. This means that usually a huge amount of (operational) data is available. However, only a small fraction of this amount is adequately translated into asset information that can be used by managers.

On the other hand, BI tools, as is, are aimed at enabling analysis of historical data only. Therefore, they are not capable of giving accurate anticipations of future trends as new scenarios of functioning are concerned. And, this is the usual shortcoming that decision makers are faced to when using Decision Support Systems (DSS) purely based on BI tools. Therefore, it is important to incorporate reliable predictive systems capable of evaluating beforehand the impact of alternative or small changes in strategic or tactical plans. This can be done by integrating what-if analysis, by means of simulation, whose goal is to inspect the behaviour of a complex system under a given scenario.

In recent years, what-if analysis has been incorporated in BI tools. For example: SAP and SAS dedicated modules, MicroStrategy, Pentaho, QlikView, Powersim Studio, etc. Also, Microsoft has been integrating this technology in spreadsheets and OLAP tools. However, these analyses rely on the application of forecasting models and data-mining techniques on static historical data, thus heavily limiting the capability of performing the evaluation of most scenarios, namely all those that are somewhat different from the existing one.

In order to overcome this shortcoming, we then propose, in Section 3, an integrated approach that fundamentally consists on setting up an iterative (and user interactive) loop between a compatible simulation model or tool and a BI tool to support decisions at the strategic and tactical levels.

The herein proposed approach is particularly welcome for the designing and planning of DRT systems. Its potential interest relies on identifying a set of questions that can then be answered with effectiveness and more efficiently than before. In particular, such set comprises a list of strategic and tactical issues that are of crucial

importance to the design and planning of DRT systems, before their implementation, thus overcoming the current lack of analytical tools to this end.

DRT systems provide transport on demand from users, using flexible schedules and routes to satisfy their travel needs. A DRT system operationally receives trip requests either for an immediate service or as an advanced reservation and organizes routes and schedules (e.g. using OR methodologies) to accommodate trip requests aiming to respond in real time to user's mobility needs. At the design phase of the DRT system, no real information exists yet on its functioning, so it is considered as a strategic and tactical decision process.

On the other hand, even after implementation, there is a lack of research work into evaluation methods and definition of real time evaluation and monitoring systems [4].

Nowadays, a particular focus is given to three fundamental broader categories of performance measures (or indicators) – economic, social and environmental. This categorization is obviously related to recent concerns about the sustainability of existing and new projects (e.g., for DRT, [5-6]).

The absence of measurement limits organizations ability to evaluate the changes and therefore precludes systematic improvement. Thus, good performance measures are a necessity for any progressive organization to recognize successful strategies and discard the unsuccessful.

### 2.1 Performance Measurement in Transportation Systems

The evaluation of the systems performance and all related decision making processes are based on the knowledge and analysis of quantitative and qualitative measures, usually named performance indicators. In this section, it is reported, in a broader perspective, the set of these indicators that may be relevant in transportation systems in general, and DRT in particular.

Performance measurement is a way of monitoring progress towards a result or goal. It is also a process of gathering information to make well-informed decisions.

Traditionally, performance measures have been largely technical in nature. However, today transportation executives and managers must address an increasingly complicated and wide-ranging set of issues regarding the best solutions on balance to transportation problems, the cost-effectiveness of proposed projects and estimated impacts of those projects.

Based on the literature related to performance measurement systems for transportation, there is a large number of measures within the following categories: preservation of assets, mobility and accessibility, operations and maintenance, and safety.

The Transportation Research Board had developed a framework [7] that transportation organizations should use in order to:

- identify performance measures that are most useful for asset management: assessing existing performance measures that are in place, identifying gaps, and considering new measures to fill gaps;
- integrate these performance measures effectively within the organization: engaging stakeholders to ensure buy-in, designing families of measures that can

be used at different organizational levels and for different types of decisions, ensuring consistency across measures, identifying needed improvements to data collection and analytic tools, designing communication devices, and documenting measurement and reporting procedures; and

- set of performance targets: establishing both long-term goals and short-to medium-term targets for performance measures.

Rather than a single prescriptive set of measures that might provide a good textbook example (but surely would not be suitable for all companies), the framework is in the form of guidance on the preferred types and forms of measures and the process by which companies should identify and implement them. The framework was developed with the recognition that each company will have a different set of circumstances and needs to consider in selecting and implementing performance measures [7].

One could not define the best set of performance measures for transportation companies, because each company has their own characteristics. However, the TRB suggests a set of "good measures" that, if implemented properly, will produce good results. Detailed information can be consulted from the referenced work [7].

In general, several measures or indicators can be used to evaluate a DRT service. Examples of those are [8-11]: the service reliability (customers' satisfaction, willingness to pay, non-accomplished valid requests, general mobility improvement), core background information, service passenger restrictions (ex, only disabled and elderly people, or other mobility-constrained persons), trip purpose, coverage of service (which days it works, where it works), easiness of reservations, pre-booking antecedence and time windows for reservations, passenger convenience (the time spent in transit becomes less satisfactory as the service area increases), need for intermodal transfers, driver satisfaction, walking time to stops, waiting time for delayed vehicles.

### 2.2 The Actual Business Intelligence Concept

Business Intelligence solutions create learning organizations by enabling companies to follow a virtuous cycle of collecting and analyzing information, devising and acting on plans, and reviewing and refining the results. To support this cycle and gain the insights that BI delivers, organizations need to implement a BI system comprised of data warehousing and analytical environments.

According to Eckerson [12], smart companies recognize that the systems that support BI solutions are very different from other systems in the company. Well-designed BI systems are adaptive by nature and they continually change to answer new and different business questions. The best way to adapt effectively is to start small and grow organically. Each new increment refines and extends the solution, adjusting to user feedback and new requirements. Additionally, the BI solutions combine data with analytical tools in order to provide information that will support top management strategic decisions [13].

The main goal of a BI solution is to collect data, store the data collected and analyse the data gathered in order to produce knowledge [13-14]. Figure 1 shows the main components of a BI tool. As part of its data warehousing environment, it takes

as input historical data (e.g. operational details) that is then treated in three different phases globally designated by ETL, which means: data extraction, transformation and cleaning, and transfer and load. As a result, all the relevant information is stored into internal a general-purpose database and then into multi-dimensional (specific-data) cubes, allowing a fast execution of dashboards, analysis and reports which is (ideally) triggered "on-the-fly" by a simple mouse click – this is the analytical environment part of the BI, which is powered by statistical, OR (e.g. forecasting) and artificial intelligence (e.g. data-mining) techniques.

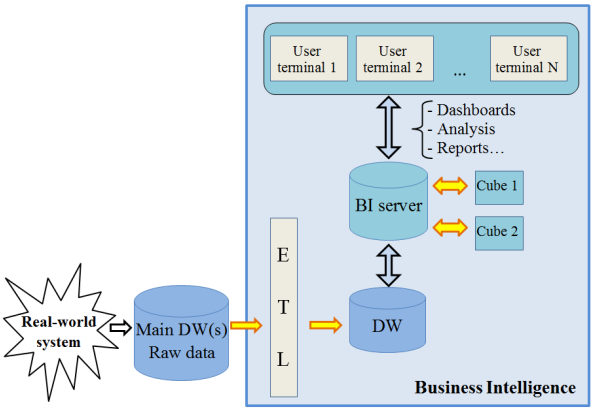



**Fig. 1.** Business intelligence components

The main features of a BI solution is therefore the creation of reports, the exploration and analysis of data and the creation of dashboards, which provide information on the overall system performance in a simplified form. The use of BI solutions are intended to provide managers and agents responsible for decision making in an organization, timely and accurate critical information about the (overview) performance of the system, so that they can make better decisions to improve such performance.

Organizations that have deployed BI solutions cite many tangible and intangible benefits. Although it is difficult to associate a concrete return of investment (ROI) resulting from these benefits, most enlightened executives place huge value on having a "single version of the truth", better information for strategic and tactical decision making and more efficient processes [12]. The most common benefits cited in the literature are the followings: time savings, "single version of truth", better strategic and plans, better tactics and decisions, more efficient processes, cost savings, greater customer and supplier satisfaction, greater employee satisfaction, return of investment, new revenues, total cost of ownership, and shareholder value.

## 3 Proposed What-If Simulation and BI Conceptual Framework

The cornerstone module of this new integrated model or tool will be the "so-called BI what-if component". The idea for this component is to provide it with properly designed wizards in order to create (new) scenarios that can be simultaneously

understood by managers and by the simulator component. As part of the scenario, the simulation model is inputted with all relevant information gathered by the traditional components of the BI (that, in turn, are based on historical data or previously simulated data), e.g. probability distributions of possible events. The approach fundamentally consists on setting up an interactive loop between a compatible simulation model or tool and a BI tool to support decisions at the strategic and tactical levels.

Figure 2 illustrates the proposed framework, an extension of the previously discussed BI decision support system (Section 2, Figure 1).

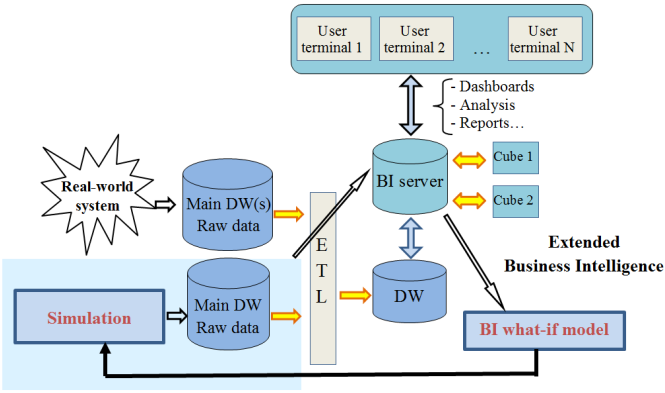



**Fig. 2.** Integrated general BI and simulation-based decision support system

In this proposed approach, the BI environment triggers the simulation runs that are necessary to estimate the performance of new scenarios. The simulation model mimics the functioning of the (new) system, and populates its proper data warehouse. This data is extracted by the BI server (if it is correctly "cleaned" already) or by the ETL component, following then the normal analytical processes of the BI. (Note, however, that it will be important to being able to distinguish between real and simulated data and derived information at any moment and at any BI component.)

## DRT Framework

In the design phase of a DRT, an integrated decision support system (IDSS) is proposed next, aiming to support decision makers in designing and implementing by firstly investigating the impacts of alternative potential specifications of the DRT system on its viability and sustainability for a given territorial area. This framework combines supply and demand data with analytical and simulation tools in order to provide information that will support top management strategic and tactical decisions.

Since the outcome of the evaluation is highly dependent on the DRT specification (in terms of operational parameters), this framework must comprise an iterative approach that consists on defining an initial DRT specification, estimating their impacts in terms of performance indicators, redefining the specification and re-estimating the

new impacts, until a suitable solution is found, in terms of technical and economic viability and sustainability.

Figure 3 illustrates a proposed integrated decision support system (IDSS) for a DRT system, pointing out its main components and sub-components, their relationships and the different levels of decision: strategic, tactical and operational (ODSS).

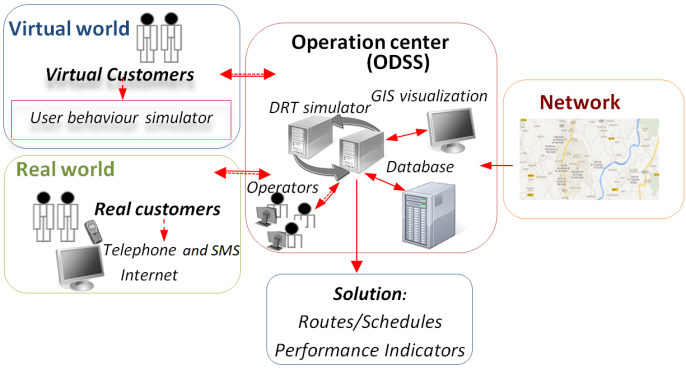



**Fig. 3.** IDSS for different levels of decision in DRT systems

The ODSS component represents the heart of the physical operations centre, receiving trip calls, processing them in terms of rational/optimized service (route) plans and scheduled services, giving advanced information to customers, monitoring performance and recording detailed historic data.

At the design phase of the DRT system, no real information exists yet on its functioning, so it is considered as a strategic and tactical decision process. In this case, the simulator component must be used in order to emulate what could happen at a real-world scenario, in order to allow the evaluation of different options (essentially, system objectives and rules) by the analyst component. The simulator will take implicitly account of their necessary functions of the ODSS, communicating directly with the remaining operational sub-components.

Trip requests are added to the system using a trip request generator which reproduces user's mobility needs for a given period (when using this approach in the design stage) or using a web application designed for this purpose.

Routes and schedules are solved by applying alternative algorithms, automatically selected according to the patterns of trips. For example, in the case where groups of persons are at the same stop (work-, market- and school-to-home trips), a Vehicle Routing Problem (VRP) savings-like heuristic can be applied, whereas in the general case where vehicles have to pick up and drop off different customers at/to different stops across the territory, a dial-a-ride problem should apply. Mean times are taken from origin-destination trip times previously stored in the system database and obtained by invoking Google Maps internet services (shortest route between two points).

The simulation component comprises two main models: (1) a demand-side model implemented as a travel request generator and users' behaviour simulator, and (2) a supply-side simulator that simulates the functioning DRT services, including the decisions made by the ODSS and the vehicles operations.

Both of these models are based on a micro-simulation event-driven approach. The main demand-side events are: trip reservation, trip cancelation, user arrival to stops (origin and destination), user no-show and non-reserved shows at stops. The main supply-side events are: trip planning events (such as hourly and daily time-scheduled planning of advanced reservations, real-time acceptances or rejections and re-planning), vehicle departure and arrival from/to stations, stops and strategic waiting and interface points.

Travel requests are generated based on socio-economic characteristics of the resident population (from Census), from domiciliary questionnaires and local authorities interviews, as well as acquired knowledge about the main attracting poles for visiting (workplaces, schools, hospitals, markets, general services, transfer stops to inter-urban transport services, etc.).

The analyst component is used to properly analyse and evaluate DRT options. It is powered by a BI type framework that starts to extract relevant data from the historic main database, transform such data and load it into a proper database system for multi-specialized analyses. It comprises different methodologies: simple and advanced statistical reporting and inference techniques, data mining and operational research inference and prospective models.

Thus, it is important to incorporate reliable predictive systems capable of evaluating beforehand the impact of alternative or small changes in strategic or tactical plans. This can be done by integrating simulation tools whose goal is to inspect the behaviour of a complex system under different scenarios. This referred framework is used to determine the system performance to evaluate the alternative specifications for a certain area. It allows choosing the better specifications and the better working rules, as illustrated in Figure 4. This figure shows how the analytic component of the system generates new alternatives to be simulated and how their performance is compared with the previous solutions already tried.

An iterative approach is used between the simulator and the analyst components. At each iteration, a solution option is simulated and evaluated by performing an assessment process. And, for each iteration (option) a large set of simulation runs (days) are realized and their operational details are stored into the main database of the operational decision support system (ODSS). The number of days is set in order to infer statistics (e.g., mean values) within a given high level of accuracy.

The result of the assessment process will provide guidelines and required feedback to adjust system resources and operating parameters, as illustrated by feedback arrows in Figure 3. Additional analysis is performed to assess solution viability and sustainability, encompassing several dimensions such as: social, economic and environmental.

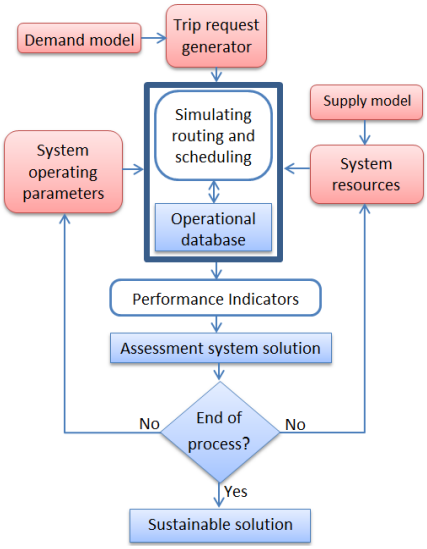



**Fig. 4.** Conceptual framework of the DSS

These impacts are increasingly important aspects of decision in transport investments. To take care of this, the analyst component incorporates, and automatically computes, a set of key performance indicators (KPIs) hierarchized by their measuring aim (e.g., evaluate client satisfaction, financial and organizational performance).

The IDSS can integrate several types of other advanced modern technologies, such as knowledge-based modules, and internet and web capabilities (e.g., free map services, map visualisation and on-line trip requests. According to Liu et al. [15], an IDSS which combines the use of models, analytical techniques, data access and retrieval functions, by using advanced differentiated technologies (this is the case), will have the advantage (in relation to traditional "static" DSS) in providing consistent, coordinated, active and global support for the various managers/analysts on their different levels of decision-making processes.

## 4 Discussion

In general, the amount of data available (e.g. detailed traffic operations events and conditions) is huge, and only a relatively small fraction of that is adequately translated into information that is used by managers. In most cases, this is restricted to the information that is easily obtained by applying traditional query database operations and graphical displays, including standard statistical elements (e.g. historical means and deviations of traffic volumes) in general-purposed tables, graphs and maps. But, in fact, much more information can be potentially gathered from real data: (1) further performance measures can still be obtained by traditional forms; (2) deeper information, somehow hidden in the databases, can be highlighted by applying more advanced analytical methods, such as data mining, optimization and

simulation, as well as advanced forecasting (e.g. auto-correlative, multi-regressive) and statistical techniques; and, very important, (3) most of all this information can be better highlighted by using alternative, more adequate forms of visualization such as well-designed graphs and maps; also, some information (e.g. spatial-temporal trends) can be exclusively evidenced by specific forms of visualization. It is widely recognized that it is easier to detect a pattern from a "picture" than from a textual or numeric explanation.

From the above, it is obvious that one of the most important problems of ITS is the analysis and management of information. This problem is becoming further relevant due to permanent increase on the availability of data and their inexpensive storage and processing power as a result of the wide implementation of modern IS/IT systems.

On the other hand, ITS planners and decision-makers need to foresee the performance of systems (existing or desired to be implemented), and therefore they need to use suitable simulation tools. The integrated framework helps DRT managers (TDC coordinators, system designers and analysts) at their different levels of decision.

At the strategic level, planners are supposed to set up the objective, aim, and the broader strategy and policies of the system. For example, in a recent work on the design and planning of demand-responsive transportation system, Dias et al. [5] report the following set of decisions at this level: How the system should be operating in the long-term in order to be viable and sustainable in the three basic terms: economic, social and environmental? What are the main objectives of its existence? Which type of services must it offer? At what level(s) of flexibility? At what price levels (whether taking account or not potential subsidisation)?

At the tactical level, decisions aim to devise ways to implement the transport system according to the strategy previously defined. The analyst component of the model monitors and analyses current performance of the system, tries to identify hidden patterns of operational data, and continually tries to devise better solutions to tackling operational problems; some solutions are automatically devised and incorporated into the ODSS component (e.g., a recurrent set of day to day fixed travel patterns are identified and a shortcut route planning procedure automatically generates a fixed service plan); however, the most part of solutions requires the investigation and validation of human analysts before their practical implementation. This planning stage is crucial for the design of the transport scheme and several authors had identified the most critical decisions. For example, for the case of demand-responsive transport [16-17]):

- Which exact level(s) of flexibility should be adopted, in terms of routes, timetables, frequencies, time-windows (e.g., arrivals at stops)?
- Which pre-booking mechanisms and rules should be adopted?
- Which resources will be used (fleet, drivers, informatics, operations centre and staff)?
- Which fare structure should be implemented?
- Which level of integration with public transport network (schedule buses/train network, etc.)?

Additionally, the ODSS component drives the DRT control and managing centre, receiving travel requests and determining the most appropriate service plans (which vehicle? which route? which schedule?).

The proposed extended and integrated BI solution can implement most of the performance measures reported before (Section 2), and easily translate them into the form of traditional tables, graphs and reports. This includes making use of simple dashboards. Most of these elements can be easily customized by end users, according to their specific needs and options.

Data visualization can be seen as the process of translating information into a visual form (usually graphs and maps), enabling users (scientists, engineers and managers) to perceive the features of the data available, and, from that point, to help them to make more effective decisions at different levels: strategic, tactical or (even) operational. For example, deciding on the design of service operations strategies or, simply deciding on the realization of a functional regulatory (e.g. corrective) measure to ensure the normal functioning of the system or to minimize the impact of an anomalous event (incident) that has occurred.

In the next section, some illustrative results of a given iteration of the above process is reported and discussed. It shows the software viability in theoretical terms (demand is defined randomly).

## 5 Illustrative Examples

In the following paragraphs, examples show the importance of adequate visualization patterns according to specific nature of data recorded and information needed.

The examples are related to the design phase of a particular DRT transportation system, before its implementation. No real information exists yet on its functioning, so it is considered as a purely strategic and tactical decision process. In this case, the simulation model component must be used in order to emulate what could happen at any tested real-world scenario, in order to allow the evaluation of different options (essentially, system objectives and rules).

In order to detect peak periods and understand correlation to service delays may be worth using simple line charts of average traffic volumes and delays estimated in the system, as a function of day-time. This should be done by taking into account the average figures for relevant zones or intersections that affects each transport service. Such information (patterns) can be also incorporated into adequate predictive models of time arrivals at stops (e.g. for public panel information). Also, panels or dashboards are very interesting and useful to the analyst, allowing an overview of the system and discovery interaction between its various elements (tables, charts, combo box, etc.).

In order to perceive and monitor service levels in different directions from a given urban center (e.g. incoming PT services bringing people to work at the morning peak), may be worth using a rose-like diagram (Figure 5). This can be done per service or group of services in each direction (e.g., north, west, south and east bands).

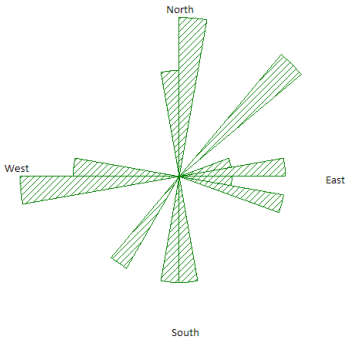



**Fig. 5.** Rose-like graph of frequencies for 12 outbound service calls at the evening peak: each route is represented by a triangle illustrating its direction and relative frequency

In order to infer the patterns of transit volumes in each direction per time of day (on average), may be worth using clustering analysis, e.g. putting in evidence similar directional service patterns among different time intervals. In these cases, the use of colored "graph-image" may be useful (Figure 6).

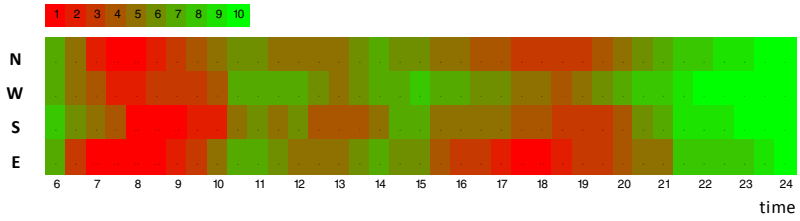



**Fig. 6.** Average inbound and outbound public transit volume by direction and time of day (30 min clusters)

The ODSS uses GIS technology by integrating the Google Maps service that allows the representation of geo-referenced information on a map. So, the graphical visualization of a particular day run, for a particular vehicle, is displayed in Figure 7. Green points represent pick up points and red points represent drop off points (or both, if there is anyone to pick up there).

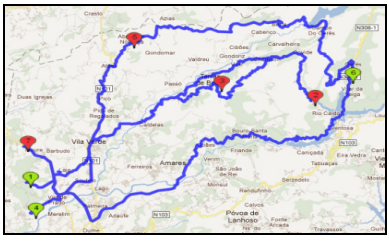


**Fig. 7.** Routes of a vehicle during a particular day

This technology integration would be also interesting for customers, as they can see their planned trips online (Web-Portal), for example for a given week. The simulation model plus to produce solutions, also print reports on systems performance to allow analysts and decision-makers to assess the DRT specification being simulated (or in use, if the system was implemented). Appropriated sets of KPIs can be obtained from a series of simulation runs (working-day unchanged pattern) for a particular DRT specification, i.e. set of fixed rules and parameters.

## 6 Conclusions

This paper presents a systematization of performance measures to be used for the evaluation of the performance of transportation systems. The model adopted is supported in the framework developed by the Transportation Research Board for transportation companies. These performance measures are supposed to be incorporated into simulation and BI transportation models, by properly designing their computation model. (Most of the measures, such as averages and standard deviations of a given relevant variable, are easy to implement, but others may not.)

One of the most important problems of ITS is the analysis and management of information gathered either from historical data or simulated data. The simulation-based BI solution herein proposed is funded on a framework that extends commercial available software by explicitly incorporating a "what-if analysis" BI component that easily creates new scenarios (of the functioning of the system) and automatically transmits all needed information (data) to the simulation model as its input.

Real and/or simulated data can be extracted by BI system, and then processed and translated into the form of traditional tables, graphs and reports, and allowing easy customization by end users. Thus, this work promoted the use of general proposed BI tools that, along with the integration of advanced analytical techniques and prospective what-if simulation, support the inference of both strategic and operational performance measures helping decision makers in the transportation sector in monitoring and managing their working system, in designing new strategic and tactical improvements, and even in designing new sustainable systems by accurately predicting their performance and related economic, social and environmental impacts.

The general framework herein proposed was extended and adapted to the particular problem of designing and planning DRT systems, thus overcoming a current lack of analytical tools to this end. The paper shows the potential interest of the proposed what-if simulation and BI integration, identifying a set of questions that can be answered with effectiveness and more efficiently before implementation. The approach is based on a iterative procedure (between BI analysis and simulation run sets) thus producing and comparing alternative DRT parameterization for different scenarios, and proactively foreseeing the correspondent effects on performance. It may contribute to implement the most probable effective and sustainable DRT for a given area.

**Acknowledgements.** The authors gratefully acknowledge the financial support provided by the European Union through FEDER-"Programa Operacional Factores de Competitividade" (POFC) and by the Portuguese Government through FCT-"Fundação para a Ciência e a Tecnologia" under projects FCOMP-01-0124-FEDER-016099 and FCOMP-01-0124-FEDER-022674.

## References

1. Sabbour, S., Lasi, H., Tessin, P.: Business intelligence and strategic decision simulation. World Academy of Science, Engineering and Technology 61, 1130–1137 (2012)
2. Dias, A., Carvalho, M.S., Telhada, J.: Simulation approach for an integrated decision support system for demand responsive transport planning and operation. In: Proc. of the 10th Annual Ind. Simulation Conference, Brno Univ. of Technology, Czech Republic (2012)
3. Telhada, J., Sampaio, P., Pereira, G., Carvalho, M.: Integrating simulation and business intelligence analyses for tactical and strategic decisions in transportation systems. In: Congreso Latino-Iberoamericano de Investigación Operativa, Rio de Janeiro (2012)
4. Battellino, H.: Transport for the transport disadvantaged: A review of service delivery models in New South Wales. Transport Policy 16, 123–129 (2009)
5. Dias, A., Carvalho, M.S., Telhada, J.: Economic evaluation of a demand responsive transport in rural area. In: Proceedings of the First International Conference on Project Economic Evaluation, Guimarães, Portugal, pp. 269–277 (2011)
6. Oliveira, J., Afonso, P., Telhada, J.: An activity-based costing model for strategic decisions in transportation on demand projects. In: Proceedings of the First International Conference on Project Economic Evaluation, Guimarães, Portugal, pp. 127–134 (2011)
7. TRB, Performance Measures and Targets for Transportation Asset Management. NCHRP Report 551. Transportation Research Board, USA (2006)
8. Mageean, J., Nelson, J.: The evaluation of demand responsive transport services in Europe. Journal of Transport Geography 11, 255–270 (2003)
9. Brake, J., Nelson, J.: A case study of flexible solutions to transport demand in a deregulated environment. Journal of Transport Geography 15, 262–273 (2007)
10. Fernández, J., Cea, J., Malbran, R.: Demand responsive urban public transport system design: Methodology and application. Transportation Research A 42, 951–972 (2008)
11. Quadrifoglio, L., Li, X.: A methodology to derive the critical demand density for designing and operating feeder transit services. Transportation Research B 43, 922–935 (2009)
12. Eckerson, W.: Smart Companies in the 21st Century: The Secrets of Creating Successful Business Intelligence Solutions. The Data Warehousing Institute, USA (2003)
13. Cody, W.F., Kreulen, J.T.: The integration of business intelligence and knowledge management. IBM Systems Journal 41, 697–713 (2002)
14. Santos, M.Y., Ramos, I.: Business Intelligence: Tecnologias da Informação na Gestão do Conhecimento. FCA-Editora de Informática, Portugal (2006)
15. Liu, S., Duffy, A., Whitfield, R., Boyle, I.: Integration of decision support systems to improve decision support performance. Knowledge Inf. Systems 22, 261–286 (2010)
16. Giannopoulos, G.: The application of information and communication technologies in transport. European Journal of Operational Research 152, 302–320 (2004)
17. Mulley, C., Nelson, J.: Flexible transport services: a new market opportunity for public transport. Research in Transportation Economics 25, 39–45 (2009)

# A Model for the Coordination of 20-foot and 40-foot Container Movements in the Hinterland of a Container Terminal

Jörn Schönberger, Tobias Buer, and Herbert Kopfer

Chair of Logistics, University of Bremen,
Wilhelm-Herbst-Str. 5, 28359 Bremen, Germany
{jsb,tobias.buer,kopfer}@uni-bremen.de
http://www.logistik.uni-bremen.de

**Abstract.** Considered is a carrier that requires decision support to organize an efficient transport of loaded and empty containers in the hinterland of a sea port. Loaded containers are handled as pickup-and-delivery requests, however, requests for empty containers are incomplete because either the pickup location of a container or the delivery location of a container is a priori unknown. The problem is modelled as a generalization of the pickup-and-delivery problem (PDP) with less-than-truckload (LTL) requests. Practically speaking, by using LTL request we are able to consider 20-foot and 40-foot containers simultaneously. This is closer to reality than most previous models discussed in the literature which use full truckload requests, i.e., only containers of homogeneous size are possible. Three types of decisions are involved in the proposed model: a determination of pickup or delivery locations for the incomplete requests, a routing of vehicles, and a routing of empty containers. The presented model is validated by means of a numerical example computed by a MIP solver.

**Keywords:** hinterland container transport, container and request matching, incomplete requests, pickup-and-delivery.

## 1 Introduction

Drayage operations involve the transport of containers between a container terminal (sea port or railway yard) and some customer locations in the hinterland of the terminal. Here, the focus is on road-based transport of the containers by means of trucks. The transport requests in drayage operations include movements of loaded as well as empty containers. Usually, containers are transported from customers to terminals or vice versa. The requests are initiated and specified by the customers. Drayage operations are of high importance for the design of efficient intermodal container traffic, because they account for about 40 percent of the total transport costs.

We consider daily drayage operations from the point of view of a trucking company (decision maker) which is responsible to organize the container movements.

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 113–127, 2013.

As new container transport requests arrive over the day the decision maker should apply online planning tools which provide a frequent and reactive plan revision of the needed hinterland container movements. In order to improve planning, the purpose of this paper is to present a decision model for a snapshot of this online planning problem. The model assumes a given set of transport requests together with the current positions of the vehicles and containers. The proposed model is an extension of the well-known less-than-truckload pickup-and-delivery problem (PDP). Unlike previous models for drayage transport operations, the model at hand is able to handle 20-foot and 40-foot containers simultaneously. The model therefore contributes to reduce the gap between real-world drayage planning and mathematical planning models used as core components of decision support systems.

The remaining paper is organized as follows. Section 2 describes the container pickup-and-delivery problem (CPDP) and Section 3 reviews related literature. A model for the CPDP with less-than-truckload requests is presented in Section 4 and is validated by means of a worked numerical example in Section 5. Section 6 concludes the paper.

## 2 The Container Pickup-and-delivery Problem

The presented problem is denoted as container pickup-and-delivery problem (CPDP). The CPDP has to be solved by a trucking company that organizes the transport of containers in the hinterland of a port or a rail yard. Fig. 1 shows an exemplary layout of such a hinterland network. There is at least one (e.g. seaside) container terminal, the set of terminal nodes is denoted as $\mathcal{N}^{TERMINAL}$. Through a terminal, containers enter or leave the hinterland. Furthermore, there is a set $\mathcal{N}^{CUSTOMER}$ of customer locations. A customer may receive a loaded container from a terminal (*import request*) or ship a loaded container to a terminal (*export request*). At the customer locations there might be an imbalance of inbound and outbound containers. Therefore, some customers have a surplus of *empty* containers while others are in need of empty containers. A customer with a need for an empty container for his or her goods initiates a *provide request*. Hence, the delivery location of a provide request is known but the empty container may be picked up from any location with a surplus of empty containers, e.g., from a container depot or from another customer. On the other hand, a customer with a surplus of empty containers initiates a *storage request* which requires a pick up of an empty container at the customer location. However, the empty container may be delivered to a container depot or to another customer in need of an empty container. Storage and provide requests are also denoted as *incomplete requests*, because either the pickup or the delivery location is initially unknown.

We take the perspective of a trucking company that is responsible for performing all required road-based container movements in the hinterland system necessary to fulfill the customer-specified requests. The considered trucking company maintains a fleet $\mathcal{V}$ of homogenous vehicles that are used to fulfill the container

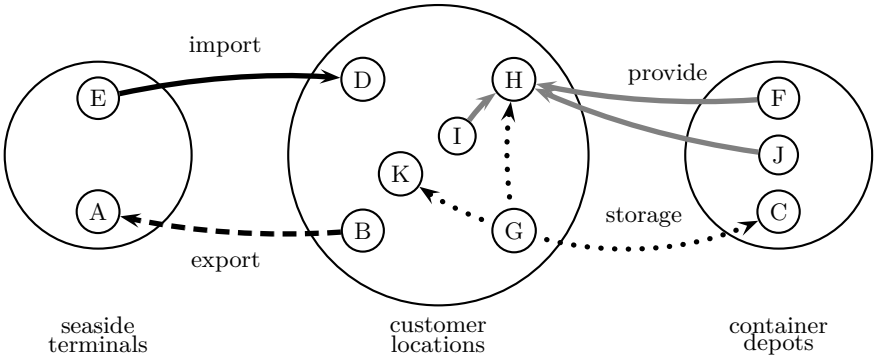



**Fig. 1.** Container flow in the hinterland of a seaport

transport requests. The vehicles are located at a single vehicle depot. We assume that it is possible to transport at most two containers – either two 20-foot containers or one 40-foot container – at the same time with a single vehicle. We consider a vehicle as a single entity, i.e., there are no trucks with trailers. However, we assume that a container can be unloaded from a truck at each node of the hinterland network.

There are four types of transport requests (cf. Fig. 1) which are characterized from the point of view of a customer: (i) export requests, (ii) import requests, (iii) storage requests, and (iv) provide requests. Each request $r$ is determined by a triple $(r^+, r^-, c(r))$. This means, that request $r$ requires the transport of a container of size $c(r)$ from the pickup location $r^+$ to the delivery location $r^-$. The container size $c(r))$ is measured in twenty-foot equivalent units (TEU) and only $c(r) = 1$ (a 20-foot container) or $c(r) = 2$ (a 40-foot container) are allowed values.

Import and export requests are regular PDP requests, i.e., both the pickup location as well as the delivery location are given. In an import request, a loaded container of known size has to be delivered from a terminal to a customer location, i.e., $(r^+, r^-, c(r)) \in \mathcal{N}^{TERMINAL} \times \mathcal{N}^{CUSTOMER} \times \{1; 2\}$. An export request, on the other hand, requires the transportation of a loaded container of known size from a customer location to a terminal, i.e., $(r^+, r^-, c(r)) \in \mathcal{N}^{CUSTOMER} \times \mathcal{N}^{TERMINAL} \times \{1; 2\}$. We distinguish these two request types for two reasons. On the one hand, this fact may be used in providing tighter model formulations in the future or it may be exploited by a heuristic. This distinction is also used in some related approaches and therefore provides some consistency with the literature, e.g. [14], [6], [12].

By means of a *provide request*, a given customer requires an *empty* container. That is, the delivery location $r^-$ of a provide request $r$ is given. However, the used pickup location is part of the decision problem. We assume, it is always possible to pickup an empty container from a container depot. However, it is also possible to pickup a container from a customer with a surplus of empty containers. Node $H$ in Fig. 1 represents the delivery location of a provide request, i.e., $H$ requires

an empty container. A potential pickup location $r^+$ for the empty container may be the container depot (nodes $F$ or $J$), or another customer (node $I$).

By means of a *storage request*, a customer requires the collection of an empty container at a location $r^+ \in \mathcal{N}^{CUSTOMER}$. The collected empty container must be forwarded either to a customer in need for an empty container or to a container depot where the container is stored. Node $G$ in Fig. 1 offers an empty container. The container may be delivered to another customer (e.g. nodes $H$ or $K$), or to a container depot (node $C$). Note, we assume that it is always possible to store a container in a container depot.

The trucking company has to care for three types of decisions. First, store and provide requests have to be completed by selecting an appropriate pickup or delivery location, respectively. Second, for each request a container has to be selected. Third, containers have to be assigned to vehicles and routes for the vehicles have to generated. The decisions are interdependent and therefore have to be taken into account simultaneously.

Due to the online characteristic of the transportation planning situation it is difficult to state an obvious objective function. However, for the snapshot planning model considered here, the minimization of the transportation effort appears to be a reasonable goal. Therefore and for simplicity, we assume the decision maker wants to minimize the total travel distance of all trucks.

## 3 Literature Review

A comparison in a nutshell of the literature related to this work is provided by Table 1. The second column states whether the discussed models use full-truckload (FT) or less-than-truckload (LTL) requests. For the scenario at hand, FT requests basically mean that only homogenous container sizes (either 40-foot or 20-foot) are supported. LTL requests support 20-foot and 40-foot containers simultaneously. The third column states whether incomplete requests, i.e., provide and store requests with unknown pickup and delivery locations, respectively, are supported. Column four (five) states whether one or multiple container depots (terminals) are used. The rightmost column indicates, whether time-windows have to be considered for the requests.

As far as we known the approach of [11] is the only approach that considers 20-foot and 40-foot containers simultaneously. However, the problem is modeled as a variant of a vehicle routing problem with backhauls and excludes the possibility that storage and provide requests are performed in the same tour, i.e., empty containers may only be transported from and to the terminal but not to customer locations which limits the potential to reduce transportation costs.

The paper [5] was among the first to consider drayage scenarios. The proposed model considers container transport between customers and a terminal and models this situation as a multiple traveling salesman problem with time windows. Incomplete store and provide requests are not supported, however.

Likewise, [4] study the vehicle routing problem with full container load with a priori known pickup and delivery locations of all requests. A container is

**Table 1.** Comparison of different models from the literature

| | truck-load | incomplete requests | depots | terminals | TW |
|---|---|---|---|---|---|
| Jula et al. 2005 [5] | FT | no | 1 | m | no |
| Imai et al. 2007 [4] | FT | no | 1 | 1 | no |
| Zhang et al. 2010 [14] | FT | yes | m | m | yes |
| Braekers et al. 2011 [1] | FT | yes | 1 | m | yes |
| Zhang et al. 2011 [13] | FT | yes | 1 | 1 | yes |
| Vidovic et al. 2011 [11] | LTL | no | 1 | 1 | no |
| Reinhardt et al. 2012 [7] | FT | no | 1 | m | yes |
| Braekers et al. 2013 [2] | FT | yes | 1 | m | yes |
| Wang and Yun 2013 [12] | FT | yes | 1 | m | yes |
| Nossack and Pesch 2013 [6] | FT | yes | m | m | yes |
| **CPDP, this paper** | LTL | yes | 1 | 1 | no |

FT = full truckload, LTL = less-than-truckload, m = multiple, TW = time windows

permanently assigned to a vehicle. That is, a vehicle which fulfills an import request first and then an export request uses for both requests the same container. This is not required in our formulation. Furthermore, an inhomogeneous fleet regarding travel cost and maximum travel time, but not with respect to the supported container sizes, is considered. In [3] the vehicle routing with full containers (VRPFC) is extended by time windows and solved as a VRPTW. The paper [7] extends the model proposed in [3] by multiple container terminals. The approach is tested by means of real world data.

[13] model the problem with flexible tasks. A flexible task is a request with an a priori unknown delivery or pickup location which is equal to our approach. Furthermore, they consider a dynamic planning situation which allows that requests may be added during the day.

In contrast to the previously mentioned approaches, empty containers are explicitly considered as transportable resources in Zhang et al. [14]. The model is denoted as Inland Container Transportation Problem (ICT) and takes into account multiple terminals and multiple depots. The ICT is also studied in [9] where a solution approach based on tabu search is proposed. In [10] a variant of the ICT model is used to evaluate the potential of collaborations between the several carriers in the hinterland. [12] extend the ICT by considering train transportation in addition to truck transportation. The model of [6] also builds upon the model of Zhang et al. [14]. Each truck serves exactly one container (i.e. only 40-foot containers are regarded). Consolidation of two 20-foot containers on a truck does not take place.

In a recent paper [2] also considers the situation where pickup or delivery locations of empty containers are undefined. An empty container allocation problem is proposed which is based on a transportation problem before solving the routing problem. However, only homogenous container sizes are considered. The goal is to minimize the travel distance of empty containers.

All in all, the CPDP presented here distinguishes itself from previous models by explicitly taking into account 20-foot and 40-foot containers simultaneously and supporting store and provide requests with unknown pickup or delivery locations. However, in contrast to many models from the literature time windows are not yet considered.

## 4 Mixed-integer Linear Program

The parameters of CPDP and the decision variables are introduced in Section 4.1. Section 4.2 presents the constraints and the objective function.

### 4.1 Parameters and Decision Variables

The basic idea of our decision model is to integrate the three decision tasks request completion, container flow determination and vehicle routes into one mixed-integer linear program. We deploy a first set of decision variables which represents the request completion decision problem. Another set of decision variables controls the container flow and a third decision variable set is dedicated to the compilation of executable vehicle routes. Two additional decision variable families are introduced in order to control the assignment of completed requests to containers and for conducting the matching of the flows of containers and the vehicle routes. All five decision variable collections are declared in Tab. 2.

**Table 2.** Decision Variables

| | |
|---|---|
| **completion of requests** | |
| $z_{ik}^{+} \in \{0;1\}$ | equals 1 if and only if node $i$ is the pickup node of container $k$ |
| $z_{ik}^{-} \in \{0;1\}$ | equals 1 if and only if node $i$ is the delivery node of container $k$ |
| **assignment of requests to containers** | |
| $y_{rk} \in \{0;1\}$ | equals 1 if and only if request $r$ is assigned to container $k$ |
| **determination of the container flow** | |
| $x_{ijk}^{CON} \in \{0;1\}$ | equals 1 if and only if container $k$ flows along arc $(i,j)$ |
| $t_{ik}^{CON} \in \mathbb{R}^{\geq 0}$ | visiting time of container $k$ at node $i$ |
| **assignment of containers to vehicles** | |
| $w_{kf} \in \{0;1\}$ | equals 1 if and only if container $k$ is assigned to vehicle $f$ |
| **determination of the vehicle routes** | |
| $x_{ijf}^{VEH} \in \{0;1\}$ | equals 1 if and only if vehicle $f$ flows along arc $(i,j)$ |
| $t_{if}^{VEH} \in \mathbb{R}^{\geq 0}$ | visiting time of vehicle $f$ at node $i$ |
| $l_{if} \in \mathbb{N}$ | outbound payload of vehicle $f$ leaving node $i$ |
| $\delta_{if} \in \mathbb{Z}$ | payload variation of vehicle $f$ at node $i$ |

Although we consider only one seaside / railroad terminal, we introduce a terminal node for each single container in the system, e.g. if the number of containers in the system is 5 then nodes 1 to 5 are nodes that belong to the terminal and node $i$ can be used exclusively by container $i$. We use this node-duplication strategy in order to make the control of the in- and outbound flow of containers into the terminal easier. We apply the same concept for only the container depot, e.g. there are $N^{CONTAINER}$ nodes that form the container depot but each node can be visited or left by exactly one container. Since a vehicle is allowed to visit the terminal (or the container depot) several times it is not necessary that we store different arrival times of a vehicle at the depot. Instead we need to store only one arrival time of a vehicle at a node. We are going to use the recursive arrival time update [8] for short cycle prevention in the vehicle flows. Therefore, we need a common outbound node and a common inbound node for each vehicle depot.

The set of all involved nodes is $\mathcal{N}$ and $\mathcal{N}^{VDEPOT}$ is the set of all nodes belonging to the truck depot. Set $\mathcal{V}$ is the set of vehicles and set $\mathcal{C}$ is the set of all containers. The set of all requests is $REQS$. It is partitioned into four subsets, i.e., the set $REQS^{EXP}$ of export requests, the set $REQS^{IMP}$ of import requests, the set $REQS^{STOR}$ of storage requests, and the set $REQS^{PROV}$ of provide requests.

The binary parameter $AVAIL(k, i)$ equals 1 if and only if container $k$ is available at node $i$. Similarly, the binary parameter $REQUI(k, i)$ equals 1 if and only if container $k$ is allowed to be delivered to node $i$. The length of arc $(i, j)$ is $d(i, j)$. A route of vehicle $f$ starts at node $START(f)$ and ends in node $STOP(f)$. If the pickup (delivery) node of request $r$ is specified by the customer then this node is referred to as $REQSTART(r)$ ($REQSTOP(r)$). The current position of container $k$ is $CONT\_LOC(k)$ and the binary parameter $FIXED(k, r)$ equals 1 if and only if request $r$ is already matched with container $k$ (this is true for all import, export and storage requests). The size of container $k$ is given by $C(k)$. A 20-foot container has size $C(k) = 1$ and a 40-foot container has size $C(k) = 2$. Finally, $\mathbf{M}$ is a sufficiently large integer number.

We construct a complete directed graph from the node set $\mathcal{N}$, which means that the arc set of the graph equals $\mathcal{N} \times \mathcal{N}$. The arcs are weighted by their travel distance $d(i, j)$. We assume that all vehicles travel at unit speed so that the absolute value of $d(i, j)$ equals the absolute value of the time needed to travel along arc $(i, j)$.

### 4.2 Constraints and Objective Function

In the following, we present the constraints required to ensure the feasibility of the generated vehicle routes. The selection of loading and unloading locations for all containers is controlled by constraints (1)–(16). The assignment of requests to containers is determined by constraints (17)–(20). Container flows are controlled by the constraints (21)–(26), and the generation of vehicle routes is subject to constraints (27)–(47). The matching of container flows and vehicle routes is achieved by constraints (48)–(54).

**Selection of Locations for Loading and Unloading**

$$z^{+}_{CONT\_LOC(k)k} \geq \sum_{r in REQS} y_{rk} \quad \forall k \in \mathcal{C} \tag{1}$$

$$\sum_{i \in \mathcal{N}} z^{+}_{ik} \geq y_{rk} \quad \forall k \in \mathcal{C}, r \in REQS \tag{2}$$

$$\sum_{i \in \mathcal{N}} z^{-}_{ik} \geq y_{rk} \quad \forall k \in \mathcal{C}, r \in REQS \tag{3}$$

$$\sum_{k \in \mathcal{C}} z^{+}_{ik} \leq 1 \quad \forall i \in \mathcal{N} \tag{4}$$

$$\sum_{k \in \mathcal{C}} z^{-}_{ik} \leq 1 \quad \forall i \in \mathcal{N} \tag{5}$$

The start node for each container is known (1). If request $r$ is assigned to container $k$ then a start node must be selected for container $k$ (2). Similarly, a stop node is selected for container $k$ assigned to request $r$ (3). A node can become the starting point of at most one container (4). Similarly, it is not allowed that two or more containers are delivered to a node (5).

$$z^{+}_{REQSTART(r)k} \geq y_{rk} \quad \forall r \in REQS^{EXP}, k \in \mathcal{C} \tag{6}$$

$$z^{-}_{REQSTOP(r)k} \geq y_{rk} \quad \forall r \in REQS^{EXP}, k \in \mathcal{C} \tag{7}$$

$$z^{+}_{REQSTART(r)k} \geq y_{rk} \quad \forall r \in REQS^{IMP}, k \in \mathcal{C} \tag{8}$$

$$z^{-}_{REQSTOP(r)k} \geq y_{rk} \quad \forall r \in REQS^{IMP}, k \in \mathcal{C} \tag{9}$$

$$z^{+}_{REQSTART(r)k} \geq y_{rk} \quad \forall r \in REQS^{STOR}, k \in \mathcal{C} \tag{10}$$

$$z^{-}_{REQSTOP(r)k} \geq y_{rk} \quad \forall r \in REQS^{PROV}, k \in \mathcal{C} \tag{11}$$

The constraint (6) ((7)) assigns the pickup (delivery) node of the export request $r$ to container $k$. For each import request, the two constraints (8) and (9) determine the corresponding container pickup and the container delivery location. The start node for each container being assigned to a storage request is known (10) but for each provision request only the destination node is known (11).

$$\sum_{i \in \mathcal{N} | REQUI(k,i) > 0} z^{-}_{ik} \geq y_{rk} \quad \forall r \in REQS^{STOR}, k \in \mathcal{C} \tag{12}$$

$$\sum_{i \in \mathcal{N} | AVAIL(k,i) > 0} z^{+}_{ik} \geq y_{rk} \quad \forall r \in REQS^{PROV}, k \in \mathcal{C} \tag{13}$$

$$z^{-}_{ik} \leq REQUI(k,i) \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{14}$$

$$z^{+}_{ik} \leq AVAIL(k,i) \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{15}$$

$$z^{+}_{ik} + z^{-}_{ik} \leq 1 \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{16}$$

Constraint (12) ensures that only a node that needs a container becomes the terminating node of a storage request. Constraint (13) ensures that only a node

that stores a container becomes the start node of a provision request. No node receives more containers than requested (14) and no node provides more containers than available at this node (15). The starting and the terminating node assigned to a container are unequal (16).

**Request to Container Assignment**

$$\sum_{k \in \mathcal{C}} y_{rk} = 1 \quad \forall r \in REQS \tag{17}$$

$$\sum_{r \in REQS} y_{rk} \leq 1 \quad \forall k \in \mathcal{C} \tag{18}$$

$$y_{rk} = FIXED(k, r) \quad \forall r \in REQ^{EXP}, k \in \mathcal{C} \tag{19}$$

$$y_{rk} = FIXED(k, r) \quad \forall r \in REQ^{IMP}, k \in \mathcal{C} \tag{20}$$

Each request is assigned to exactly one container (17) and each container receives at most one request (18). The two constraints (19) and (20) ensure consistency of fully known import and export requests with the corresponding used containers.

**Container Flows**

$$x_{iik}^{CON} = 0 \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{21}$$

$$\sum_{j \in \mathcal{N} | j \neq i} x_{ijk}^{CON} \geq z_{ik}^{+} \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{22}$$

$$\sum_{j \in \mathcal{N} | j \neq i} x_{jik}^{CON} \geq z_{ik}^{-} \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{23}$$

$$z_{jk}^{+} + \sum_{iin\mathcal{N}} x_{ijk}^{CON} = z_{jk}^{-} + \sum_{i \in \mathcal{N}} x_{jik}^{CON} \quad \forall j \in \mathcal{N}, k \in \mathcal{C} \tag{24}$$

Self-referencing flows of containers are not allowed (21). If node $i$ has been selected as starting node of the flow of container $k$ then container $k$ leaves node $i$ (22). In case that $i$ has been selected to be the terminal node of the flow of container $k$ then constraint (23) ensures that $k$ travels to $i$. Constraint (24) preserves the balance of inbound and outbound flow of container $k$ at node $i$. If $i$ is the selected start node of the flow of container $k$ then there is no inbound flow of $k$ into $i$. Similarly, if $i$ is selected to be the terminus node of container $k$ then there is no outbound flow of $k$ from $i$.

$$t_{ik}^{CON} \geq 0 \quad \forall i \in \mathcal{N}, k \in \mathcal{C} \tag{25}$$

$$t_{ik}^{CON} + d(i, j) \cdot x_{ijk}^{CON} \leq t_{jk}^{CON} + (1 - x_{ijk}^{CON}) \cdot \mathbf{M} \quad \forall i, j \in \mathcal{N}, k \in \mathcal{C} \tag{26}$$

**Vehicle Routing.** In order to avoid short cycles of the flowing containers, we recursively calculate the arrival times of container $k$ at the nodes along the determined flow path (25)-(26) as proposed by [8].

$$x_{ijf}^{VEH} = 0 \quad \forall f \in \mathcal{V}, i \in \mathcal{N} \tag{27}$$

$$\sum_{j \in \mathcal{N}} x_{START(f),j,f}^{VEH} = 1 \quad \forall f \in \mathcal{V} \tag{28}$$

$$\sum_{j \in \mathcal{N}} x_{j,STOP(f),f}^{VEH} = 1 \quad \forall f \in \mathcal{V} \tag{29}$$

$$\sum_{j \in \mathcal{N}} x_{jif}^{VEH} = \sum_{j \in \mathcal{N}} x_{ijf}^{VEH} \quad \forall f \in \mathcal{V}, i \in \mathcal{N} \setminus \mathcal{N}^{VDEPOT} \tag{30}$$

Self-referencing loops are forbidden by (27). Each vehicle leaves its home position (28) and terminates in its designated terminus node (29). Constraint (30) ensures the vehicle flow preservation at nodes away from the vehicle depot.

$$t_{if}^{VEH} \geq 0 \quad \forall i \in \mathcal{N}, f \in \mathcal{V} \tag{31}$$

$$t_{if}^{VEH} + d(i,j) \cdot x_{ijf}^{VEH} \tag{32}$$

$$\leq t_{jf}^{VEH} + (1 - x_{ijf}^{VEH}) \cdot \mathbf{M} \quad \forall i,j \in \mathcal{N}, f \in \mathcal{V} \tag{33}$$

$$t_{if}^{VEH} \leq t_{jf}^{VEH} + (1 - w_{kf}) \cdot \mathbf{M} \tag{34}$$

$$+(1 - z_{ik}^{+}) \cdot \mathbf{M} + (1 - z_{jk}^{-}) \cdot \mathbf{M} \quad \forall i,j \in \mathcal{N}, f \in \mathcal{V}, k \in \mathcal{C} \tag{35}$$

We use the arrival time - based prevention of short cycles of the vehicle routes (31)-(33). Constraint (35) guarantees that vehicle $f$ visits the pickup location $i$ of container $k$ (equiv. to $z_{ik}^{+} = 1$) earlier than the associated delivery location $j$ of $k$ (equiv. to $z_{jk}^{-} = 1$) if vehicle $f$ has been selected to move container $k$ (equiv. to $w_{kf} = 1$).

$$t_{if}^{VEH} = 0 \quad \forall i \in \mathcal{N}, f \in \mathcal{V} \tag{36}$$

$$t_{STOP(f)f}^{VEH} \leq T_{max} \quad \forall f \in \mathcal{V} \tag{37}$$

All vehicles depart from their depot at time 0 (36) and return there not later than $T_{max}$ (37).

$$l_{START(f)f} = 0 \quad \forall f \in \mathcal{V} \tag{38}$$

$$l_{STOP(f)f} = 0 \quad \forall f \in \mathcal{V} \tag{39}$$

$$\delta_{START(f)f} = 0 \quad \forall f \in \mathcal{V} \tag{40}$$

$$\delta_{STOP(f)f} = 0 \quad \forall f \in \mathcal{V} \tag{41}$$

Each vehicle leaves its starting node empty (38) and heads emptied towards its terminus node (39). No load variation is effective at these two locations (40),(41).

$$\delta_{if} + (1 - w_{kf}) \cdot \mathbf{M} + (1 - z_{ik}^{+}) \cdot \mathbf{M} \geq C(k) \quad \forall i \in \mathcal{N}, k \in \mathcal{C}, f \in \mathcal{V} \tag{42}$$

$$\delta_{if} \leq C(k) + (1 - z_{ik}^{+}) \cdot \mathbf{M} \quad \forall f \in \mathcal{V}, i \in \mathcal{N}, k \in \mathcal{C} \tag{43}$$

$$\delta_{if} + (1 - w_{kf}) \cdot \mathbf{M} + (1 - z_{ik}^{-}) \cdot \mathbf{M} \leq -C(k) \quad \forall i \in \mathcal{N}, k \in \mathcal{C}, f \in \mathcal{V} \tag{44}$$

$$\delta_{if} \geq -C(k) - (1 - z_{ik}^{-}) \cdot \mathbf{M} \quad \forall f \in \mathcal{V}, i \in \mathcal{N}, k \in \mathcal{C} \tag{45}$$

We set, by constraints (42) and (43), the payload variation $\delta_{if}$ of vehicle $f$ at node $i$ to $C(k)$ if container $k$ has been assigned to vehicle $f$ ($w_{kf} = 1$) and if $i$ is the starting (pickup) node associated with container $k$ ($z^+_{ik} = 1$). Similarly, we set, by constraints (44) and (45), the payload variation $\delta_{if}$ of vehicle $f$ at node $i$ to $-C(k)$ if container $k$ has been assigned to vehicle $f$ ($w_{kf} = 1$) and if $i$ is the terminus (delivery) node associated with container $k$ ($z^-_{ik} = 1$).

$$l_{if} + \delta_{jf} \leq l_{jf} + (1 - x^{VEH}_{ijf}) \cdot \mathbf{M} \quad \forall i, j \in \mathcal{N}, f \in \mathcal{V} \tag{46}$$

$$l_{jf} \leq 2 + (1 - \sum_{i \in \mathcal{N}} x^{VEH}_{jif}) \cdot \mathbf{M} \quad \forall j \in \mathcal{N}, f \in \mathcal{V} \tag{47}$$

The payload variation along the route of vehicle $f$ is updated by $\delta_{if}$ recursively (46) so that the payload is increased by at least 1 (TEU) if $f$ visits a node with a container pickup but the payload is reduced by at least 1 (TEU) if $f$ visits a container delivery node. It is not allowed that a vehicle carries a payload of more than 2 TEU at any position of its route (47).

**Matching of Container Flows and Vehicle Routes**

$$\sum_{f \in \mathcal{V}} w_{cf} \leq 1 \ \ \forall c \in \mathcal{C} \tag{48}$$

$$\sum_{f \in \mathcal{V}} w_{cf} \geq \sum_{r \in REQS} y_{rc} \ \ \forall c \in \mathcal{C} \tag{49}$$

Each container is assigned to at most one vehicle for being moved (48). If container $k$ is assigned to a request $r$ then it is necessary that $k$ is assigned to exactly one vehicle $f$ (49).

$$x^{CON}_{ijc} \leq x^{VEH}_{ijf} + (1 - w_{cf}) \cdot \mathbf{M} \ \ \forall i, j \in \mathcal{N}, c \in \mathcal{C}, f \in \mathcal{V} \tag{50}$$

In the situation where container $k$ moves along arc $(i, j)$ and if container $k$ has been assigned to vehicle $f$ it is necessary that vehicle f also travels along arc $(i, j)$ (50).

$$2 \cdot \left( \sum_{k \in \mathcal{C}} w_{kf} \right) \geq -1 + \sum_{i,j \in \mathcal{N}} x^{VEH}_{ijf} \ \ \forall f \in \mathcal{V} \tag{51}$$

With the goal to avoid the insertion of dummy nodes into the route of vehicle $f$ with the goal to artificially reduce the payload of $v$, we restrict the number of visited nodes to the least possible number (51), e.g. if $m$ requests are assigned to $f$ then the least necessary number of visited nodes is $2m + 1$ (the starting node is not visited but only left by $f$).

$$\sum_{r \in REQS} y_{rk} = \sum_{f \in \mathcal{V}} w_{kf} \ \ \forall k \in \mathcal{C} \tag{52}$$

In order to prevent the incorporation of nodes associated with containers that have not been selected for incorporation into the routes of the vehicles constraint (52) is added to the model.

$$\sum_{r \in REQS} y_{rk} = \sum_{i \in \mathcal{N}} z_{ik}^{-} \quad \forall k \in \mathcal{C} \tag{53}$$

$$\sum_{r \in REQS} y_{rk} = \sum_{i \in \mathcal{N}} z_{ik}^{+} \quad \forall k \in \mathcal{C} \tag{54}$$

Finally, constraints (53) and (54) are used to prevent the visitation of containers for which a pickup and a delivery node are specified but which is not assigned to any request.

**Objective Function**

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{f \in \mathcal{V}} d(i,j) \cdot x_{ijf}^{VEH} \rightarrow \min \tag{55}$$

The trucking company aims at minimizing the overall travel distance of their vehicles (55).

## 5 Numerical Example

The presented decision model is very difficult to solve. Using appropriate parameter instantiations, well-studied NP-hard decision problems like the pickup and delivery problem are identified as special cases of the presented model. We cannot expect to solve this model to optimality within an acceptable time. Initial computational experiments with the commercial solver CPLEX showed that even quite small instances of the integrated request completion, container flow and vehicle routing problem cannot be solved within reasonable computation time independently from the applied objective function. Often, the solver even failed to identify a feasible solution.

Being aware of the challenging difficulty of the CPDP, we focus on the assessment of the suitability of the presented constraint collection for a rather small example scenario. We instruct CPLEX to identify any feasible solution and terminate the search then. We use the identified solution to verify the feasibility with respect to the aforementioned problem constraints. In order to get further insights into the performance of the model it is necessary to develop specific (meta-)heuristics which is, however, beyond the scope of the present paper.

The example shown in Fig. 2 comprises a hinterland system with eight 20-foot containers. There is a (seaside or railway) container terminal comprising the nodes 1 to 8. Six customer locations are represented by nodes 9 to 14. Two trucks of capacity 2 TEU are available at a vehicle depot with outbound node 15 and inbound node 16. One hinterland container depot (nodes 17 to 24) can be used to store empty containers or to pickup empty containers needed for the fulfillment of a provide request. Nodes with a surplus of containers are shaded

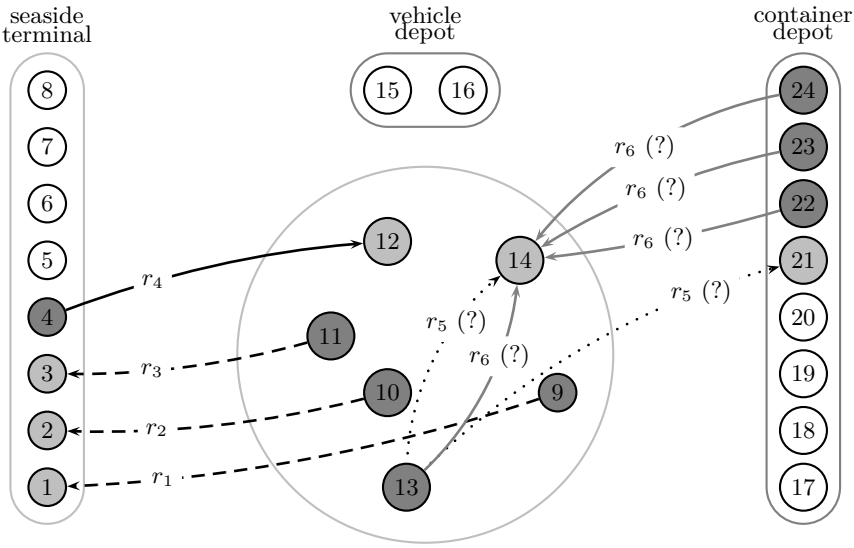



**Fig. 2.** Example scenario with 6 requests, 8 containers, and 5 vehicles. Request $r_5$ and $r_6$ are a priori incomplete.

dark gray while nodes with a need for a container are shaded light gray. The remaining white nodes neither require nor need a container.

The trucking company has to fulfill six requests ($r_1$ to $r_6$). Export requests $r_1, r_2, r_3$ and import request $r_4$ are complete requests. Request $r_5$ is an incomplete *storage request*, i.e., customer node 13 offers an empty container and we have to decide whether this empty container shall be delivered to the container depot or to another customer that requires an empty container. Request $r_6$ is an incomplete *provide request*, i.e., customer node 14 requires an empty container and we have to decide whether it is provided from the container depot or from another customer. The maximal length $T_{max}$ of a vehicle route is set to 600 time units.

The model of this small scenario comprises already 29000 constraints, 6545 binary decision variables, 96 integer decision variables, and 241 continuous decision variables. Fig. 3 shows the resulting feasible vehicles routes. The first route (black line) consists of the node sequence $15, 11, 3, 9, 1, 16$ and the second route (dotted line) consists of the node sequence $15, 22, 10, 14, 2, 4, 13, 21, 12, 16$. *First*, a container has been selected for each request. Here, request $r_i$ has been assigned to container $i$ ($i = 1, \ldots, 6$). The storage request $r_5$ has been completed by determining node 21 as sink of the flow of container 5. The provide request $r_6$ has been completed by choosing node 22 as source, i.e., container 6 is selected as start node of request $r_6$. *Second*, the flow through the network has been determined for each container. *Third*, feasible routes are setup for the two trucks by assigning containers to vehicles.

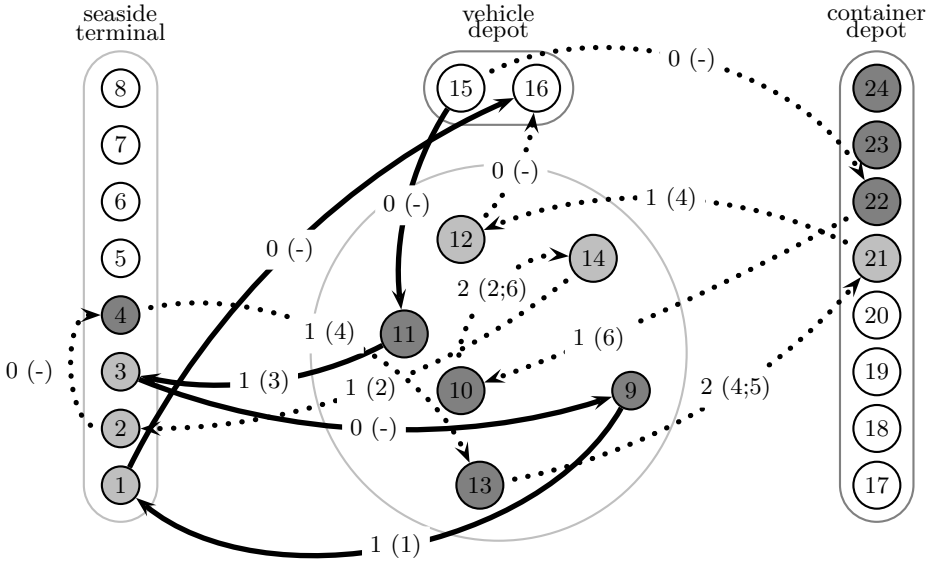



**Fig. 3.** Example scenario: routes of vehicle 1 (continuous) and vehicle 2 (dotted). The arc labels represent the number of TEU currently loaded by the vehicle and the numbers in brackets refer to the IDs of the currently loaded containers.

## 6 Summary and Outlook

Efficient drayage operations in the hinterland of a seaport or railroad terminal are important for worldwide container traffic. We introduced a mixed integer linear model which includes two additional decisions in extension to traditional vehicle routing with pickup-and-delivery decisions. On the one hand, for an empty container at a customer location a storage location has to be determined. On the other hand, a provide location for an empty container has to be determined for those customers that demand an empty container. The proposed model for the container pickup-and-delivery problem (CPDP) deals with these additional decisions simultaneously and integrated with a less-than-truckload PDP model. For the first time, transport of 20-foot and 40-foot containers in drayage operations can be managed by means of a single model. Future research will focus on the development of heuristic solution approaches for the CPDP.

## References

1. Braekers, K., Caris, A., Janssens, G.K.: A deterministic annealing algorithm for a bi-objective full truckload vehicle routing problem in drayage operations. Procedia - Social and Behavioral Sciences 20, 344–353 (2011)
2. Braekers, K., Caris, A., Janssens, G.: Integrated planning of loaded and empty container movements. OR Spectrum 35(2), 457–478 (2013)

3. Caris, A., Janssens, G.: A local search heuristic for the pre- and end-haulage of intermodal container terminals. Computers & Operations Research 36(10), 2763–2772 (2009)
4. Imai, A., Nishimura, E., Current, J.: A lagrangian relaxation-based heuristic for the vehicle routing with full container load. European Journal of Operational Research 176(1), 87–105 (2007)
5. Jula, H., Dessouky, M., Ioannou, P., Chassiakos, A.: Container movement by trucks in metropolitan networks: modeling and optimization. Transportation Research Part E: Logistics and Transportation Review 41(3), 235–259 (2005)
6. Nossack, J., Pesch, E.: A truck scheduling problem arising in intermodal container transportation. European Journal of Operational Research (2013), doi:10.1016/j.ejor.2013.04.042
7. Reinhardt, L., Spoorendonk, S., Pisinger, D.: Solving vehicle routing with full container load and time windows. In: Hu, H., Shi, X., Stahlbock, R., Voß, S. (eds.) ICCL 2012. LNCS, vol. 7555, pp. 120–128. Springer, Heidelberg (2012)
8. Sigurd, M., Pisinger, D., Sig, M.: The pickup and delivery problem with time windows and precedences. Technical Report 00/08, DIKU, University of Copenhagen (2008)
9. Sterzik, S., Kopfer, H.: A tabu search heuristic for the inland container transportation problem. Computers & Operations Research 40(4), 953–962 (2013)
10. Sterzik, S., Kopfer, H., Yun, W.Y.: Reducing hinterland transportation costs through container sharing. Flexible Services and Manufacturing Journal (2012), doi:10.1007/s10696-012-9167-y
11. Vidovic, M., Radivojevic, G., Rakovic, B.: Vehicle routing in containers pickup up and delivery processes. Procedia - Social and Behavioral Sciences 20, 335–343 (2011)
12. Wang, W.F., Yun, W.Y.: Scheduling for inland container truck and train transportation. International Journal of Production Economics 143(2), 349–356 (2013)
13. Zhang, G., Smilowitz, K., Erera, A.: Dynamic planning for urban drayage operations. Transportation Research Part E: Logistics and Transportation Review 47(5), 764–777 (2011)
14. Zhang, R., Yun, W., Kopfer, H.: Heuristic-based truck scheduling for inland container transportation. OR Spectrum 32, 787–808 (2010)

# Dynamic Collaborative Transportation Planning: A Rolling Horizon Planning Approach

Xin Wang and Herbert Kopfer

Chair of Logistics, University of Bremen,
Wilhelm-Herbst-Str. 5, 28359 Bremen, Germany
{xin.wang,kopfer}@uni-bremen.de
http://www.logistik.uni-bremen.de

**Abstract.** Small and mid-sized freight carriers are suggested to cooperate with fellow companies and to perform collaborative transportation planning (CTP) enabling an exchange of customer requests for higher operational efficiencies. In order to realize the synergy embedded in CTP, appropriate request exchange mechanisms have to be developed. Compared to their static counterparts, little research has been conducted on dynamic CTP problems (DCTPP). In this paper, the DCTPP of a coalition of freight carriers offering full truckload transport services is presented. A rolling horizon planning approach is proposed to solve this problem. Computational studies are carried out to identify the potential of cost-savings through CTP and to analyze the influence of the degree of dynamism of instances on the planning results.

**Keywords:** Dynamic pickup and delivery problem, Dynamic collaborative transportation planning, Rolling horizon planning, Horizontal carrier collaboration, Distributed decision making, Request exchange.

## 1 Introduction

Nowadays, freight carriers are confronted with increasing pressures to improve the operational efficiency of transportation fulfillment. Compared with large forwarding companies which can consolidate requests to a high extend by exploiting the economy of scale, it is difficult for small and mid-sized carriers (SMC) to reach the same efficiency. Exploiting synergies by performing collaborative planning in a horizontal coalition may be considered as a promising remedy.

Through exchanging customer transportation requests with other coalition members, freight carriers may reduce their operational costs up to 30 percent [7]. In such horizontal coalitions of carriers, transportation planning is not performed by each participant separately but in a concerted fashion, which is referred to as collaborative transportation planning (CTP) [31]. According to [27], CTP can be understood here as a joint decision making process for aligning plans of individual members with the aim of achieving coordination in light of information asymmetry. This means that private information and autonomy of coalition members will be preserved while the quality of the plans of all members is intended to be improved. In CTP, all partners generate plans only for

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 128–142, 2013.

themselves and try to harmonize their plans with those of other coalition partners by applying appropriate request exchange mechanisms. The specific goal of CTP is to find a reallocation of requests among the coalition members with total costs smaller than the sum of the forwarders' individual fulfillment costs, i.e., without cooperation with other members. The obtained cost-savings present the joint benefits of the coalition that cannot be achieved individually. These joint benefits are then to be shared by the members in such a way that all freight carriers will benefit from the cooperation.

CTP has been investigated by many researchers in the last decade. However, most of their research on CTP considers only the static planning scenario, where all information is available a priori at the time CTP is performed. On the contrary, little attention has been paid to dynamic CTP problems (DCTPP). In this paper, a DCTPP with deterministic information in which full truckload (FTL) pickup and delivery transportation requests are to be served is investigated. In order to solve the DCTPP, the route-based request exchange mechanism proposed in [31] for the static CTP problem is embedded into a rolling horizon planning framework. Using computer simulation, the potential of cost-savings through CTP and the influence of varying degrees of dynamism of instances on the planning results are analyzed.

This paper is organized as follows. In Section 2, we give a brief literature review on related topics. In Section 3, the DCTPP is formally described. In Section 4, the rolling horizon planning solution framework for the DCTPP is proposed. Computational study and a comprehensive discussion on the results are shown in Section 5. Section 6 concludes this paper.

## 2 Literature Review

The DCTPP extends the static CTP problem over a long planning horizon with gradually revealed request information. The two closely related topics of this problem are the dynamic and deterministic vehicle routing and the CTP. In this section, literature reviews on these two related topics are given.

### 2.1 Dynamic and Deterministic Vehicle Routing

In contrast to static routing problems where all input data are known a priori at the time when the routes are constructed, some input data are revealed or updated during the period of time in which operations take place in a dynamic routing problem [2]. Moreover, a routing problem can be either deterministic or stochastic according to the information quality which reflects possible uncertainty on the available data [20]. The reader is referred to [20] for a recent review on dynamic routing problems. Specifically, [2] review the literature on dynamic pickup and delivery problems (PDP). In this section, we focus on some major contributions to dynamic and deterministic routing problems.

The solution approaches for dynamic routing problems can be classified according to [20] into two categories. The first one comprehends periodic reoptimization approaches. After having constructed a set of initial routes, these

approaches periodically solve static problems corresponding to the current states triggered by specific incidences. The first type of such triggers is an update of input data, which is practically the release of a new customer request. This strategy is used in [21,32,33]. In [21] a dynamic programming approach is proposed for the dynamic dial-a-ride problem. Rolling horizon approaches are used to solve the real-time FTL PDP with time windows [32] and its extension by the possibility of rejecting requests and soft time windows [33].

Alternatively, a re-optimization can be trigged whenever a predefined interval is reached. A branch-and-price algorithm is applied in [23] to solve the static PDP with time windows for each single re-optimization. Similarly, in [4] the column generation scheme is used in a dynamic approach which solves each time the vehicle routing problem (VRP) with time windows [6]. An ant colony system is developed in [18] for a dynamic VRP in which encrypted information about characteristics of good solutions are conserved in a pheromone matrix and passed on to the next static problem after a predefined duration.

The second category of solution approaches is referred to as continuous re-optimization. The general idea is to run the optimization routines and maintain information on good solutions in an adaptive memory. Whenever an event occurs, e.g., a new request is known or a vehicle has finished its current job, a decision procedure stops the optimization routine and updates the routes. After that, the optimization routine is restarted with the new generated solutions. Different from periodic re-optimization approaches, a driver does not know the next customer to serve until he has finished his current job. Diverse optimization routines are used in these approaches. In [13] a parallel tabu search (TS) proposed in [28] is applied. Another TS is proposed in [12] while the concept of ejection chains [14] is used to construct a powerful neighborhood structure.

In addition to work on developing efficient solution approaches for dynamic routing problems, the influence of different waiting strategies on the quality of solutions to dynamic routing problems is studied in [17]. In [29] it is analyzed how valuable it is for carriers to have the load information in advance.

### 2.2 Collaborative Transportation Planning

Some general opportunities and impediments of horizontal cooperation in logistics are revealed in [8]. A review of the literature on horizontal cooperation in logistics can be found in [9]. Through performing CTP with fellow carriers in a horizontal coalition, SMCs can reduce their operational costs to a great extent. Several studies on identifying this cost-saving potential show that CTP can account to 5-30 percent cost reduction compared to the case without request exchange [7,10,16]. However, appropriate request exchange mechanisms must be developed to exploit the cost-saving potential embedded in CTP. Such mechanisms have to be (1) simple and implementable, (2) effective in terms of generating high joint benefits [19], and (3) able to deal with distributed information and decision-making competencies [31].

Some approaches have been proposed in literature to tackle this challenging task for static CTP problems. In most approaches auction mechanisms are used.

A request exchange mechanism using a combinatorial auction (CA) [1] under the assumption that the fulfillment costs for any bundle of requests can be exactly evaluated is proposed in [15]. In [25] a one-round auction and an iterative auction are proposed for a scenario with several warehouses supplying single commodity goods to customers, while the VRP with time windows has to be solved for each warehouse. In these auctions, only requests located between neighboring profit centers are exchanged. In [3] the CTP is solved for the traveling salesman problem with pickup and delivery using both a Vickrey auction [30] and a CA. Vehicle capacity is not considered in this approach. In [24] a CA-based approach is proposed for a similar problem, in which also time windows of requests have been considered. Since the introduction of time windows makes it impossible for the coalition to fulfill all the requests, subcontracting is also considered. In [31] a route-based request exchange mechanism is proposed for less-than-truckload requests with time windows and capacity restrictions, while each member has only a limited fixed number of vehicles in his fleet. This approach is similar to the CA-based mechanisms as it also allows the exchange of bundles of requests.

Besides the mechanisms that exchange request bundles, pair-wise exchange based approaches are also proposed to solve the static CTP problems. An incentive compatible approach using cryptographic techniques for swapping pickup and delivery requests among independent carriers is proposed in [5]. A protocol is developed that is secure against a centralized model referred to as the "trusted broker" model, where all parties give their input to the broker and the broker computes and returns the result [5]. Bilateral exchange mechanisms enabling FTL forwarders exchanging lanes to be served are proposed in [19]. Four settings concerning information transparency and transfer payments between the two sides of an exchange are studied.

From a methodological point of view, these approaches are developed based on two different strategies. The first one is used in [3,5,19,25]. It depends on the calculation of the marginal costs for single requests or request bundles. The basic idea is to choose those requests with the highest marginal costs and to offer them for exchange. If an exchange resulting in a better solution for all parties is found, it will be accepted and the exchange process continues. Otherwise the process ends. The approaches in [24] and [31] follow the Dantzig-Wolfe decomposition principle [11]. They decompose the CTP problems into several sub-problems reflecting the routing decisions of single carriers and a coordinating problem in form of a CA and a set partitioning problem (SPP) or a set covering problem (SCP). Using this decomposition scheme, each member in the coalition decides only for his part without regarding the feasibility of any other part [11] and without having to expose private information.

In contrast to static CTP problems, little research was conducted on DCTPP. To the best of our knowledge, only in [26] a variant of DCTPP of a coalition of SMC fulfilling FTL pickup and delivery requests is studied. Whenever a member acquires a customer request, he launches an auction for the assignment of this request and acts as an auctioneer. Other coalition members acting as bidders calculate the marginal costs of inserting this request into their existing routes.

The request will be transferred to the bidder with the lowest bid price if this price is lower than the auctioneer's own marginal costs.

## 3 Problem Definition

Suppose a horizontal carrier coalition with $m$ independent members. Each member $i$, $i = 1, \ldots, m$, acquires FTL requests from customers during the entire time horizon $[0, \infty]$. The set of requests of carrier $i$ is denoted as $R_i$. Let $R = \cup_{i=1}^{m} R_i$ denote the set of all requests to be served. A request $r \in R$ must be transported from its pickup location to the corresponding delivery location. At each location $u$, the operation (pickup or delivery) must be started in a customer defined time window $[a_u, b_u]$. The service time at $u$ is given by $s_u$. In a static PDP, all request information is available at the time of planning. In a dynamic PDP, however, requests may be released while the routes are executed. The time when a request $r$ is released to a carrier is called the release time $t_r^{rls}$. Any request $r$ in carrier $i$'s request portfolio $R_i$ can be executed by a vehicle $k$ of $i$'s own fleet $K_i$ which is called *self-fulfillment* or by *subcontracting* it to a common carrier outside the coalition. In case of subcontracting, a certain price $\gamma_r$ must be paid. At the beginning of the whole time horizon ($t_0 = 0$), the vehicles in $K_i$ are staying at different locations. Let $K = \cup_{i=1}^{m} K_i$ denote the entire fleet of the coalition. All vehicles in $K$ are homogeneous so that every request in $R$ can be fulfilled by any vehicle in $K$. However, all carriers may use their own scheme to calculate route costs for their own fleet, i.e., the vehicles may have different variable cost $\beta_k$.

In case of *isolated planning* (IP), in which no request is exchanged among the member carriers, the dynamic transportation planning problem consists of constructing request fulfillment plans for each carrier $i$ such that all requests in portfolio $R_i$ are fulfilled either by self-fulfillment or subcontracting. The total costs of IP include the route costs for all vehicles owned by partners and the charges paid to common carriers for subcontracting. In the *collaborative planning* scenario, carriers have a third request fulfillment option besides self-fulfillment and subcontracting which is to exchange requests with other partners. Each exchange results an internal payment among the participating members of the coalition. The expenses of acquired requests through exchange will be covered by such internal payments so that no member will get worse off. Associated with the payments, the responsibility of the fulfillment of the transferred requests is also shifted. A member who gets some requests from others must guarantee the fulfillment of these requests. However, they can choose either to fulfill them using their own vehicles or by subcontracting. In the latter case, they have to pay the freight charges to the common carriers on their own.

## 4 Solution Approach

To solve the DCTPP, the route-based request exchange mechanism proposed for the static CTP in [31] is integrated into a rolling horizon planning framework.

### 4.1 Rolling Horizon Planning Framework

Figure 1 illustratively shows the rolling horizon planning framework. The entire time horizon is divided into a series of planning periods. All planning periods have the same length $\tau$. We use index $p$ to denote the planning periods, $p = 0, 1, 2, \ldots, \infty$. At the beginning time point $t_0 = 0$, an initial plan $\Pi_1$ for the first planning horizon including $H$ following planning periods, i.e., planning periods $p = 1, 2, \ldots, H$, is constructed. The plan for the first planning period is fixed. The plan for the next planning periods $p = 2, \ldots, H$, however, will be actualized in the forthcoming plans as a result of the dynamically released new requests during the execution of planned routes. After that, at the end of each planning period $p-1$, i.e., at the planning time $t_{p-1} = (p-1)\tau$, $p = 2, 3, \ldots, \infty$, a new plan $\Pi_p$ for the next planning horizon ranging from planning periods $p$ to $p + H - 1$ will be made based on the updated status. Again, the partial plan constructed for planning period $p$ will be fixed while the rest part will be kept as changeable. Figure 1 shows the case when $H$ is set to 3.

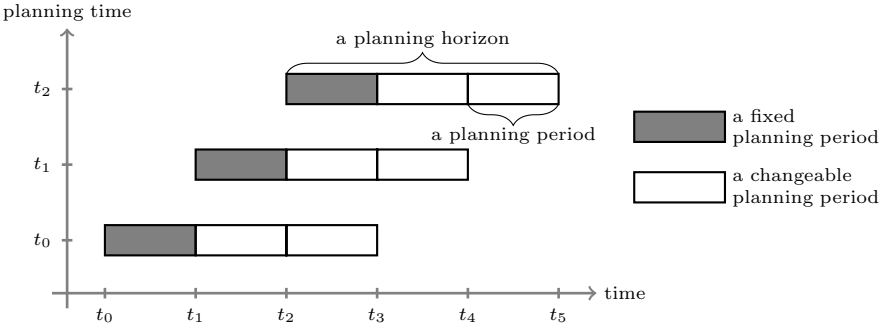



**Fig. 1.** Framework of rolling horizon planning

At each planning time $t_p$, $p = 0, 1, 2, \ldots, \infty$, each vehicle can be assigned some new customer requests after they have finished all the requests that have been planned fixed in its route in the previous planning periods. In case that a vehicle has already finished serving all assigned requests, it waits at the location of the last customer until new orders are assigned.

Let $R_i^{p,a}$ denote the set of all active requests of carrier $i$ at the planning time $t_{p-1} = (p-1)\tau$, $p = 1, 2, \ldots, \infty$. An active request at a specific time point is a released yet not executed request that has not been fixed planned in previous planning periods. The requests in $R_i^{p,a}$ can be further differentiated into two subsets according to their urgency. The most urgent requests are labeled as *due requests* that must be fixed planned for the next planning period $p$, while the remaining part of requests are called *non-due requests* that will be planned in the afterwards planning periods $p + 1$ to $p + H - 1$. We use $R_i^{p,d}$ and $R_i^{p,n}$ to denote the sets of due requests and non-due requests, respectively, and we have

$R_i^{p,a} = R_i^{p,d} \cup R^{p,n}$. Then, the sets of all active, due, and non-due requests of the entire coalition can be defined as $R^{p,a} = \sum_{i=1}^{m} R_i^{p,a}$, $R^{p,d} = \sum_{i=1}^{m} R_i^{p,d}$, and $R^{p,n} = \sum_{i=1}^{m} R_i^{p,n}$, respectively.

Based on the differentiation of requests according to their due time, two planning strategies concerning the information about future requests are defined. Carriers may plan in either a myopic or a forward-looking way by considering only requests in $R_i^{p,d}$, $(H = 1)$, or all requests known in $R_i^{p,a}$, $(H > 1)$, respectively. In a *myopic planning* (MY), only due requests are considered in each planning. In case of *forward-looking planning* (FL), all requests known at the planning time are taken into account in the planning. Considering the fact that the plan for the non-due requests will be actualized in the next planning periods and thus these requests are not equally important at that moment of planning as the due requests, the importance of these two types of requests must be accordingly differentiated. This can be done by multiplying the outsourcing prices for the requests with different weights according to their degrees of urgency. It is worth mentioning that by setting the weight of requests that need to be served after the next $H$ planning periods to zero, the term $R_i^{p,a}$ can still be used for the rolling horizon planning framework with a constant $H$. Thus, we also use $R_i^{p,a}$ to denote all known requests in the next planning horizon at planning time $t_{p-1} = (p-1)\tau$. In the collaborative planning scenario, these two strategies can be formulated by using the corresponding notations $R^{p,d}$ and $R^{p,a}$.

### 4.2 Identification of Requests for Exchange

The first issue in applying the rolling horizon planning framework is to determine the set of requests to be considered in each static planning and the set of requests to be exchanged. This question can be easily answered in case of MY that deals only with those requests in $R^{p,d}$. All requests in $R^{p,d}$ are considered in the planning and all of them are to be fixed assigned among the coalition members through request exchange. For the FL however, all known requests $R^{p,a}$ in the next $H$ planning periods are considered in constructing the plan. However, only the most urgent requests, i.e., the requests in $R^{p,d}$, for which the plan must be fixed, will be exchanged. Actually, MY can be realized by setting the weight for all requests in $R^{p,n}$ to zero and thus be regarded as a special case of FL in a broad sense.

It is important to differentiate the requests to be considered and those to be exchanged in case of FL. The reason is that each reallocation of requests is associated with transfer payments among the members which are determined based on the costs of routes. These costs are supposed to be as accurate as possible. In a dynamic environment, these costs can only be precisely determined for the partial routes serving the most urgent requests since they will not be changed during their execution. Another important reason is that, although request information in advance should be considered in planning, the plan for requests that are not urgent should not be fixed as soon as the plan is made [29].

Obviously, at planning time $t = p\tau$, all requests that must be picked up in the next planning period $p+1$, i.e., the service at their pickup locations must be

started before the end of the planning period $p+1$, are due requests. Additionally, in order to improve the continuity of the plan, requests whose pickups must be served soon after the end of the planning period $p+1$ are also considered as due requests in our approach.

### 4.3 Route-Based Request Exchange

In order to solve the DCTPP, the route-based request exchange mechanism in [31] is adapted and used to solve the static problem periodically within the rolling horizon framework. Figure 2 shows an overview of the route-based request exchange mechanism.

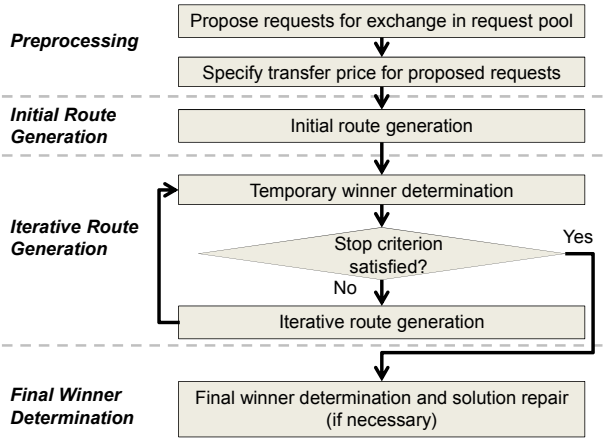



**Fig. 2.** Overview of the route-based request exchange mechanism [31]

In *preprocessing*, all partners propose all their requests for exchange, i.e., $R_i^{p,d}$ in case of MY, and $R_i^{p,d}$ in case of FL, into a common request pool for exchange. After the requests have been proposed, each carrier $i$ solves for himself a routing problem using his own fleet only for their own requests. Through introducing a weight $w_r^p$ for each request $r \in R_i^{p,a}$, the objective function of this routing problem can be formulated as:

$$\min \sum_{k \in K_i^p} \sum_{(u,v) \in A_i^p} \beta_k d_{uv} x_{uvk} + \sum_{r \in R_i^{p,a}} \gamma_r w_r^p z_r \tag{1}$$

$K_i^p \subseteq K_i$ is the vehicle set available in the current planning. $A_i^p$ is the set of arcs defined by carrier $i$'s own requests and vehicles and $d_{uv}$ is the distance of the arc $(u, v)$. The decision variable $x_{uvk} \in \{0, 1\}$ indicates if an arc $(u, v)$ is used in vehicle $k$'s route and the other binary variable $z_r$ indicates if a request is outsourced to a common carrier. In case of MY, all requests in $R_i^{p,n}$ have $w_r^p = 0$ and all due requests have a weight of 1. In case of FL, all due requests also have

the weight of 1 while other requests have a weight less than 1. Next, each carrier declares the total costs for his own request portfolio $R_i^{p,d}$ as a transfer price, which is the maximum payment for the fulfillment of these requests without cooperation. In case of FL, a route may have both due and non-due requests. Only the first part containing the due requests are considered, i.e., the partial route costs until the delivery location of the last due request in a route are used. Denote this transfer price as $C_i^{p,d}$. The total transfer prices $TC_{IP}^{p,d} = \sum_{i=1}^{m} C_i^{p,d}$ represent the total costs for this single planning period $p$ without cooperation. This piece of information is used for the acceptance of CTP solutions. A solution of CTP will only be accepted, when it is better than the solution of IP, i.e., $TC_{CTP}^{p,d} < TC_{IP}^{p,d}$.

The next step is the *initial route generation*. Each carrier $i$ solves a routing problem for his own available vehicles $K_i^p$ and generates a set of routes fulfilling selected requests from the request pool $R^{p,a}$. The objective function of this routing problem is the same as (1) except the sets $A_i^p$ and $R_i^{p,a}$ are replaced by $A^p$ and $R^{p,a}$, respectively. Through solving this problem in a heuristic manner, a set of different solutions can be obtained. The first part of the routes in these solutions containing only the due requests are reported as candidate routes to an "agent" of the coalition. The costs of the partial routes will be declared as the costs of these candidate routes.

After the set of candidate routes has been initialized, the iterative process starts. In each iteration, the agent solves the winner determination problem (WDP) in form of a linear relaxed SPP such that the total costs of the fulfillment of all due requests are minimized. The option that the requests can be outsourced to common carriers is also considered in the WDP such that each due request $r \in R^{p,d}$ is either to be assigned to a winning candidate route or common carriers for the price $\gamma_r$. This price is the same to all coalition members.

Suppose that there are $n^p$ requests in $r \in R^{p,d}$ and $b_i$ candidate routes have been proposed by carrier $i$ in the *initial route generation* step. For each request $r$, a fictive route representing the common carrier option with the route cost of $\gamma_r$ is also added into the set of candidate routes. Thus, $b = \sum_{i=1}^{m} b_i + n^p$ candidate routes are there in total. Let $a_{rj} = 1$ indicate that request $r$ is in route $j$ and $a_{rj} = 0$ otherwise, $j = 1, 2, \ldots, b$. We use $f_{kj} = 1$ to indicate that route $j$ is proposed for vehicle $k$, $k \in K^p$. The cost of a candidate route is denoted by $c_j$. The WDP can be formulated as follows by introducing the binary variable $y_j$, $j = 1, 2, \ldots, b$, to indicate whether a route is chosen as a winning route:

$$\min TC_{CTP}^{p} = \sum_{j=1}^{b} c_j y_j \tag{2}$$

subject to:

$$\sum_{j=1}^{b} a_{rj} y_j = 1 \quad \forall r \in R^{p,d} \tag{3}$$

$$\sum_{j=1}^{b} f_{kj} y_j \leq 1 \quad \forall k \in K^p \tag{4}$$

Constraints (3) make sure that each request is assigned to exactly one winning route and constraints (4) ensure that each vehicle is assigned no more than one route. The agent solves the linear relaxation of this model and gets the dual values related to (3) for requests $\pi_r$ and related to (4) for vehicles $\sigma_k$. These values are sent back to the carriers for the next iteration of route generation.

Using this feedback information, carriers can generate and submit new candidate routes in the *iterative route generation* step by solving another routing problem with the following objective function:

$$\min \sum_{k \in K_i^p} \sum_{(u,v) \in A^p} \beta_k d_{uv} x_{uvk} + \sum_{r \in R^{p,d}} \pi_r z_r + \sum_{r \in R^{p,a}} \gamma_r w_r^p z_r \tag{5}$$

Again, only the first part of the routes in the solutions obtained by using heuristic algorithms are proposed as candidate routes. They are then added into the existing set of candidate routes. Readers are referred to [31] for a more detailed presentation of the derivation of this objective function and more details of the whole request exchange mechanism.

Iterative route generation ends when predefined criteria are satisfied. The whole process ends with the *final winner determination* step, in which the agent solves an SCP-based formulation of the WDP by replacing (3) with

$$\sum_{j=1}^{b} a_{rj} y_j \geq 1 \quad \forall r \in R^{p,d} \tag{6}$$

If some requests belong to more than one winning route in the WDP solution, the agent calls a simple repair routine to obtain a feasible solution for the CTP problem. The result of the WDP will only be accepted if its total costs are less than the total transfer prices reported by all members in the first phase, which indicates a cost reduction for the entire coalition.

## 5 Computational Experiments

In order to validate the functionality of the proposed solution approach for the DCTPP and to evaluate the cost reduction potential embedded in the CTP reachable by using the proposed approach, a computational study based on some new theoretical instances is conducted.

### 5.1 Test Cases

In the first step, ten static instances are generated in total. The procedure of generating static instances begins with generating request information in an iterative fashion. The length per planning period is set as $\tau = 100$. In each iteration $it$, $it = 1, \ldots, 30$, which corresponds to a time interval with $(\tau \cdot it, \tau \cdot (it + 1)]$, about 40 requests for the entire coalition are generated. In order to introduce fluctuation of customer demand, this number is adjusted with an amount up to $\pm 20\%$. These requests are then assigned to three coalition carriers according

to the request weights which are also randomly generated in $[0.7, 1.3]$ for each carrier in each iteration. As a result, each instance consists of three carriers and in average 1186 requests over the entire planning horizon.

Pickup and delivery locations of requests are randomly located within a square of size $200 \times 150$. The distance between two location nodes is the Euclidean distance. For more than 80% of all requests, the distance between pickup and delivery locations is in the range from 20 to 160 and the average value is about 90. The velocity of all vehicles is assumed to be 1 so that the driving time between two nodes equals the distance. The variable cost rate $\beta_k$ for a distance unit is set to 1 for all vehicles $k \in K$. The time windows for a request $r$ generated in iteration $it$ are defined in the following way. Let $r^+$ and $r^-$ represent the pickup and delivery locations of request $r$. For the first iteration ($it = 1$), $a_{r^+}$ is given a random value in $[\tau/3, \tau]$. In the following iterations, $a_{r^+}$ is set to a random value in range $(\tau \cdot it, \tau \cdot (it + 1)]$. $b_{r^+}$ is given by adding $a_{r^+}$ with a time window width, which is determined as $\tau/2 \pm 30\%$. The time window for the delivery location $[a_{r^-}, b_{r^-}]$ is simply defined as $[a_{r^+} + d'_{r^+r^-} + s_{r^+}, b_{r^+} + d'_{r^+r^-} + s_{r^+}]$, while $d'_{r^+r^-}$ is the driving time from $r^+$ to $r^-$ and $s_{r^+}$ is the service time at $r^+$. All operations are assigned the same service time of 10. Since the execution of some requests generated in the last iteration $it = 30$ may be finished later, the entire planning horizon of our instances is $[0, 3300]$.

Since requests are allowed to be transferred to common carriers, the price for outsourcing requests must also be specified. This cost $\gamma_r$ for a request $r$ is calculated as $\gamma_r = \varphi d_r \theta^{d_r}$, where $\varphi$ is a constant cost rate per distance unit and set to 2, and $d_r$ is the adjusted travel distance between pickup and delivery locations and defined as $d_r = \max\{5, d_{r^+r^-}\}$. The motivation to use the adjusted travel distance is that the common carriers charge a fixed minimum fee for each request if the distance to travel lies below a specific level. $\theta$ is a parameter which is set to 0.9986. $\theta^{d_r}$ can be seen as a distance-dependent discount on the cost rate per distance unit. Through introducing $\theta$, the fact that in practice freight rates reduce with increasing transportation length can be captured in our instances.

Each carrier is assigned a vehicle fleet. The number of vehicles is determined as the average request number per planning period with a deviation of up to $\pm 30\%$. Vehicles are located at randomly generated start locations with empty load at the very beginning $t_0$. The average number of vehicles per carrier in an instance is 13.3, while the specific numbers are varying from 9 to 17.

The second step is to assign each request $r$ a release time $t_r^{rls}$ to make a static instance to a dynamic one. $\delta_1\%$ of all requests are given a release time of $(a_{r^+} - 3\tau)$, $\delta_2\%$ of $(a_{r^+} - 2\tau)$, and $\delta_3\%$ of $(a_{r^+} - \tau)$. Negative values are set to zero since we start our simulation at $t_0 = 0$. Through changing the values of $\delta$, the degree of dynamism of a single instance can be varied. Using the ten static instances, three sets of DCTPP instances are generated. They have different degrees of dynamism: high dynamism (HD), middle dynamism (MD), and low dynamism (LD). The parameter triple $(\delta_1, \delta_2, \delta_3)$ is fixed for set HD to $(10, 10, 80)$, for MD to $(10, 80, 10)$, and for LD to $(80, 10, 10)$.

### 5.2 Simulation Study

Both scenarios IP and CTP are simulated in our computational study. An important issue in our simulation is the specification of due requests. In our simulation, a known request is declared in the planning procedure for planning period $p+1$ at $t = p\tau$ as a due request if the service at its pickup location must be started before $(p+1.25)\tau$. For the very last period, all requests are labeled as due requests. We consider both strategies MY and FL as well as each degree of dynamism for the planning scenarios IP and CTP. As a result, four settings are defined: (IP, MY), (IP, FL), (CTP, MY), and (CTP, FL). While using the FL strategy, the potential outsourcing costs are adjusted by a weight smaller than 1. Non-due requests with $(p + 1.25)\tau < b_{r^+} \leq (p + 2)\tau$ are multiplied with a weight factor of 0.75, while the remaining non-due requests with $b_{r^+} > (p+2)\tau$ are multiplied with $0.75^2 \approx 0.56$.

An adaptive large neighborhood search heuristic (ALNS) which has been developed based on the ALNS heuristic proposed for the PDP with time windows in [22] is used to solve the routing problems in the *initial route generation* and *iterative route generation*. A more detailed description of this heuristic can be found in [31]. Both the linear relaxation of the SPP-based and the SCP-based WDP are solved by CPLEX. The time limit for each static CTP is set to 6.5 min. This time limit is meant to synchronize the decentralized process since in most situations, the total computational time used is much less than this limit.

For each instance and simulation setting, we run the simulation three times and the average costs per instance are calculated. These average costs of all ten instances in each set are given in Table 1. For each set, the total costs $TC$, the total route costs $TC_R$, the total costs of subcontracting requests $TC_C$, and the total number of outsourced requests are given.

**Table 1.** Results of simulation

| | HD | | | | MD | | | | LD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $TC$ | $TC_R$ | $TC_C$ | $n_C$ | $TC$ | $TC_R$ | $TC_C$ | $n_C$ | $TC$ | $TC_R$ | $TC_C$ | $n_C$ |
| IP,MY | 156375 | 86613 | 69762 | 541 | 156287 | 86185 | 70102 | 545 | 156262 | 86532 | 69730 | 543 |
| IP,FL | 156290 | 86068 | 70222 | 544 | 155767 | 84613 | 71154 | 554 | 155767 | 84196 | 71571 | 558 |
| CTP,MY | 146197 | 94967 | 51230 | 434 | 146137 | 94806 | 51331 | 436 | 146120 | 95134 | 50986 | 433 |
| CTP,FL | 146134 | 94412 | 51722 | 441 | 145805 | 93193 | 52612 | 455 | 145791 | 92966 | 52825 | 456 |

CTP obviously outperforms IP for both MY and FL and all instances. The cost reductions realized by CTP roughly amount to 6.5%. On the other hand, carriers can reduce costs by planning in a forward-looking way. However, compared with CTP, the benefits of FL are rather ignorable. It can be concluded that it is a much more promising strategy to perform CTP than trying to improve the own planning individually. The reason can be derived from a comparison of $TC_R$, $TC_C$, and $n_C$ between IP and CTP. The costs of routes and the number of requests planned in routes of CTP is significantly higher than for IP, while the total costs $TC$ are smaller. This indicates that the routes constructed in CTP are obviously more efficient and the decisions on outsourcing are made in a better

way. Comparing the performance of CTP for different sets, CTP performs even slightly better for the highly dynamic set HD than for the other two sets MD and LD. This implies that CTP is more valuable for carriers when a quick response is expected to the release of customer requests because the exchange of requests increases the possibility to find a cheaper way to fulfill the requests and this is very important in highly dynamic environments.

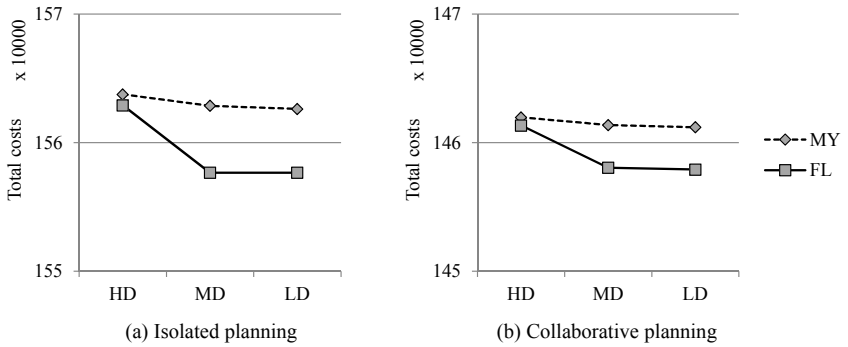



**Fig. 3.** Comparison of myopic planning and forward-looking planning

It can be observed that if carriers plan myopically, the quality of the planning suffers little from the increment of dynamism of the instances. However, the benefit of having request information in advance can be clearly seen when carriers plan in a forward-looking way. For set HD, FL performs almost equal with MY because there is little information about the future known. For sets MD and LD, FL clearly outperforms MY. This implies that carriers are able to reduce costs if they can acquire request information in advance and consider this information in their planning. However, the solution quality cannot be enormously improved by having this information much earlier than necessary. This effect can be seen through the comparison of the results for sets MD and LD in Figure 3. In other words, in a dynamic environment, it is unnecessary for carriers to consider such requests in their current planning which are to be fulfilled in the far future.

## 6 Conclusion

In order to improve their operational efficiency, small and mid-sized freight carriers are suggested to integrate external transportation resources into their operational planning process. Besides the option to outsource requests to common carriers, they can further reduce costs through setting up a horizontal coalition with fellow companies and perform CTP. In this paper, the DCTPP of a coalition of carriers offering exchangeable FTL transport services is introduced. A rolling horizon planning approach is proposed to solve this problem. A computational study is carried out to identify the potential of cost-savings through CTP and

to analyze the performance of the proposed solution approach under varying conditions. For the new instances generated for the computational study, CTP can achieve a cost reduction of about 6.5% for all settings. CTP is especially preferable when the environment is highly dynamic. It is further suggested to use advance information on requests by performing forward-looking planning for better results. However, only requests that are to be fulfilled in the near future should be considered. This can reduce the computational efforts needed for the planning without harming the quality of the planning. In the future, more computational experiments have to be conducted to specify the key factors that affect the performance of the proposed approach for the DCTPP. An example of such factors is the definition of due requests. Further factors that need to be studied include time window width and different cost level of subcontracting.

**Acknowledgments.** This research was supported by the German Research Foundation (DFG) as part of the project "Kooperative Rundreiseplanung bei rollierender Planung".

## References

1. Abrache, J., Crainic, T.G., Gendreau, M., Rekik, M.: Combinatorial auctions. Annals of Operations Research 153, 131–164 (2007)
2. Berbeglia, G., Cordeau, J.-F., Laporte, G.: Dynamic pickup and delivery problems. European Journal of Operational Research 202, 8–15 (2010)
3. Berger, S., Bierwirth, C.: Solutions to the request reassignment in collaborative carrier networks. Transportation Research E 46, 627–638 (2010)
4. Chen, Z.-L., Xu, H.: Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. Transportation Science 40, 74–88 (2006)
5. Clifton, C., Iyer, A., Cho, R., Jiang, W., Kantarciglu, M., Vaidya, J.: An Approach to Securely Identifying Beneficial Collaboration in Decentralized Logistics Systems. Manufacturing and Service Operations Management 10, 108–125 (2008)
6. Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M., Soumis, F.: VRP with time windows. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem, pp. 157–194. SIAM, Philadelphia (2002)
7. Cruijssen, F., Bräysy, O., Dullaert, W., Fleuren, H., Solomon, M.: Joint route planning under varying market conditions. International Journal of Physical Distribution and Logistics Management, 287–304 (2007)
8. Cruijssen, F., Cools, M., Dullaert, W.: Horizontal cooperation in logistics: Opportunities and impediments. Transportation Research E 43, 129–142 (2007)
9. Cruijssen, F., Dullaert, W., Fleuren, H.: Horizontal cooperation in transport and logistics: A literature review. Transportation Journal 46, 22–39 (2007)
10. Cruijssen, F., Salomon, M.: Empirical Study: order sharing between transportation companies may result in cost reductions between 5 to 15 percent. Technical report, CentER Discussion Paper (2004)
11. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Operations Research 8, 101–111 (1960)
12. Gendreau, M., Guertin, F., Potvin, J.-Y., Séguin, R.: Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Transportation Research C 14, 157–174 (2006)

13. Gendreau, M., Guertin, F., Potvin, J.-Y., Taillard, E.: Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science 33, 381–390 (1999)
14. Glover, F.: Ejection chains, reference structures and alternating path methods for traveling salesman problems. Discrete Applied Mathematics 65, 223–253 (1996)
15. Krajewska, M.A., Kopfer, H.: Collaborating freight forwarding enterprises. OR Spectrum 28, 301–317 (2006)
16. Krajewska, M.A., Kopfer, H., Laporte, G., Ropke, S., Zaccour, G.: Horizontal cooperation of freight carriers: request allocation and profit sharing. Journal of the Operational Research Society 59, 1483–1491 (2008)
17. Mitrović-Minić, S., Laporte, G.: Waiting strategies for the dynamic pickup and delivery problem with time windows. Transportation Research B 38, 635–655 (2004)
18. Montemanni, R., Gambardella, L.M., Rizzoli, A.E., Donati, A.V.: Ant colony system for a dynamic vehicle routing problem. Journal of Combinatorial Optimization 10, 327–343 (2005)
19. Özener, O., Ergun, O., Savelsbergh, M.: Lane-exchange mechanisms for truckload carrier collaboration. Transportation Science 45, 1–17 (2011)
20. Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L.: A review of dynamic vehicle routing problems. European Journal of Operational Research 225, 1–11 (2013)
21. Psaraftis, H.N.: A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science 14, 130–154 (1980)
22. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 445–472 (2006)
23. Savelsbergh, M., Sol, M.: DRIVE: Dynamic Routing of Independent Vehicles. Operations Research 46, 474–490 (1998)
24. Schönberger, J.: Operational Freight Carrier Planning, pp. 135–148. Springer, Berlin (2005)
25. Schwind, M., Gujo, O., Vykoukal, J.: A combinatorial intra-enterprise exchange for logistics services. Information Systems and e-Business Management 7, 447–471 (2009)
26. Song, J., Regan, A.: An auction based collaborative carrier network. Technical report. Institute of Transportation Studies, University of California, Irvine (2003)
27. Stadtler, H.: A framework for collaborative planning and state-of-the-art. OR Spectrum 31, 5–30 (2009)
28. Taillard, E., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y.: A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 31, 170–186 (1997)
29. Tjokroamidjojo, D., Kutanoglu, E., Taylor, G.D.: Quantifying the value of advance load information in truckload trucking. Transportation Research E 42, 340–357 (2006)
30. Vickery, W.: Counterspeculation, Auctions, and Competitive Sealed Tenders. Journal of Finance 16, 8–37 (1961)
31. Wang, X., Kopfer, H.: Collaborative transportation planning of less-than-truckload freight: A route-based request exchange mechanism. To appear in: OR Spectrum (2013), doi:10.1007/s00291-013-0331-x
32. Yang, J., Jaillet, P., Mahmassani, H.S.: On-line algorithms for truck fleet assignment and scheduling under real-time information. Transportation Research Record 1667, 107–113 (1998)
33. Yang, J., Jaillet, P., Mahmassani, H.S.: Real-time multivehicle truckload pickup and delivery problems. Transportation Science 38, 135–148 (2004)

# A GRASP for a Multi-depot Multi-commodity Pickup and Delivery Problem with Time Windows and Heterogeneous Fleet in the Bottled Beverage Industry

Roger Z. Ríos-Mercado[1], J. Fabián López-Pérez[2], and Andrés Castrillón-Escobar[1]

[1] Universidad Autónoma de Nuevo León (UANL), Graduate Program in Systems Engineering, AP 111–F, Cd. Universitaria, San Nicolás de los Garza, NL 66450, Mexico
{roger,andres}@yalma.fime.uanl.mx

[2] Universidad Autónoma de Nuevo León (UANL), Graduate Program in Management Science, CEDEEM, San Nicolás de los Garza, NL 66450, Mexico
jesus.lopezpz@uanl.edu.mx

**Abstract.** A pickup and delivery vehicle routing problem from a real-world bottled-beverage distribution company is addressed. The problem consists of deciding how to load every trailer, how to configure the vehicles in terms of the trailers, and how to route the vehicles, so as to minimize routing and fixed costs. The problem includes several features such as time windows on both customers and vehicles, multiple depots, multiple product delivery, split delivery, heterogeneous fleet, and dock capacity, to name a few. To solve this problem a GRASP-based heuristic is proposed. Problem decomposition, vehicle composition and route construction mechanisms, and local search procedures for different types of neighborhoods are developed. The heuristic is empirically assessed over a wide set of instances. Empirical results show that the proposed method obtains solutions of better quality than those reported by the company.

**Keywords:** Vehicle routing, Pickup and delivery, Multi-commodity, Heterogeneous fleet, Metaheuristics, GRASP.

## 1 Introduction

The problem addressed in this paper comes from a bottled beverage distribution company located in the city of Monterrey, Mexico. This company needs to distribute its products across several distribution centers. The distribution centers can also in turn request to withdraw products or relocate products to other centers. Due to its specific features, this problem is classified as a vehicle routing problem with pickup and delivery (PDP) with multiple depots.

The problem consists of transporting products among plants and distribution centers in a network. In each plant, there is a determined number of vehicles that

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 143–157, 2013.

can be used to do the routing. In addition, a decision has to be made so as to haul one or two trailers into a vehicle. This decision has two main effects. As it turns out, both the cost and time of traveling between two given points depend on whether the vehicle is single or double. A single vehicle travels faster and costs less than a double vehicle. However, a double vehicle has more capacity and can carry a larger amount of product between points. There are time windows requirements in both distribution centers and vehicles. The later is due to the fact that a daily maintenance for each vehicle must be performed at specific times. The orders can be split into several deliveries and vehicles can also visit several times a distribution center if necessary. The planning horizon is daily. In addition, there is a dock capacity in each distribution center that must be met. Each vehicle must return to its original depot by the end of the day.

Each trailer has two compartments: one on the top and one on the bottom. For each trailer, the capacity of both top and bottom is known, but not necessarily equal for every trailer. Moreover, some trailers may have a fixed shelf (or division between the top and bottom compartments) which would become an important issue when loading the products. There are many different products handled by the company; however, by and large they can be divided into two main types of products: returnable (type R) and non-returnable (type N). This distinction is important because type R products are heavier. As a direct consequence of this, the following loading rules must be met: (i) if the trailer has a shelf, any product, regardless its type, can be placed in any part of the trailer compartments; (ii) if the trailer does not have a shelf, it is strictly forbidden to place a type R product on top of a type N product due to the weight difference between products. It is assumed that when a product of type R is delivered to a customer a matching amount of empty type R bottles is picked up. The goal is to find the best route configuration that minimizes the total cost associated with the product routing and vehicle usage. In doing so, decisions so as to what type of vehicle is to be used and how each trailer must be loaded must be made simultaneously.

Although the field of vehicle routing, particularly the class of PDPs, has been widely studied (e.g. [3, 6, 8–10]), to the best of our knowledge there is no other work in the literature addressing a PDP with all the features mentioned previously simultaneously present, namely, multiple depots, heterogeneous vehicles, time windows at nodes and vehicles, split deliveries, multiple products, vehicles with compartments, dock capacity. For recent extensive surveys on PDPs the reader is referred to the work of Berbeglia et al. [1], Drexl [4], and Parragh et al. [7].

In this paper, we introduce a mixed-integer linear programming (MILP) model for this problem. Given its inherent computational complexity, we propose a metaheuristic framework that combines decomposition, greedy randomized construction, and local search components. The effectiveness of the heuristic is empirically assessed. It was found that the proposed method found better routings and truck configurations than those reported by the company, resulting in important cost reductions.

## 2 Mathematical Formulation

For modeling the problem we define the following notation.
*Sets*

$T$ Set of trailers.
$T_i$ Subset of trailers that are initially located at node $i$.
$T^e(T^w)$ Subset of trailers with (without) a shelf.
$K$ Set of vehicles.
$K_i$ Subset of vehicles that are initially located at node $i$.
$V$ Plants and distribution centers (nodes).
$E$ Edges.
$P$ Products.
$P_R(P_N)$ Subset of returnable (non-returnable) products.
$H$ Time index (usually hours) set.

*Parameters*

$c_t^{\text{top}}$ Capacity (number of pallets) on top compartment of trailer $t$.
$c_t^{\text{bot}}$ Capacity (number of pallets) on bottom compartment of trailer $t$.
$\tau_t^l$ Time in which trailer $t$ can start its route.
$\tau_t^u$ Time in which trailer $t$ must return to its plant.
$\sigma(t)$ Original location of trailer $t$.
$\sigma(k)$ Original location of vehicle $k$.
$n_{ip}^+$ Amount of product $p$ (number of 12-bottle boxes) that must be picked up from node $i$.
$n_{ip}^-$ Amount of product $p$ that must be delivered to node $i$.
$(a_i, b_i)$ Time window for servicing node $i$.
$o_p$ Number of 12-bottle boxes of product $p$ that fit in a pallet.
$c_{ij}^s$ Travel cost from node $i$ to $j$ using a single vehicle.
$c_{ij}^d$ Travel cost from node $i$ to $j$ using a double vehicle.
$s_{ij}^s$ Travel time from node $i$ to $j$ using a single vehicle.
$s_{ij}^d$ Travel time from node $i$ to$j$ using a double vehicle.
$S$ Customer service time at each node.
$f_{ih}$ Available dock capacity in node $i$ at time $h$.
$C$ Fixed cost for vehicle use.

*Binary variables*

$w_{kt}$ = 1, if trailer $t$ is assigned to vehicle $k$; 0, otherwise
$x_{ijk}$ = 1, if vehicle $k$ travels directly from node $i$ to $j$; 0, otherwise
$y_{ikh}$ = 1, if vehicle $k$ arrives at node $i$ at time $h$; 0, otherwise
$z_k$ = 1, if vehicle $k$ is used; 0, otherwise

*Integer variables*

$v_k^{\text{top}}$ Number of pallets on top of vehicle $k$.
$v_k^{\text{bel}}$ Number of pallets on bottom of vehicle $k$.

$v_k^{\text{type}}$ Configuration of vehicle $k$ (=-1/0/1 if unassigned/single/double).
$g_{ikp}^-$ Amount of product $p$ (number of 12-bottle boxes) that vehicle $k$ will deliver at node $i$.
$g_{ikp}^+$ Amount of product $p$ that vehicle $k$ will pickup at node $i$.
$q_{ikp}$ Amount of product $p$ in vehicle $k$ after servicing node $i$.

Note: At the start of a route, $g^-_{\sigma(k)kp} = 0$ and $q_{\sigma(k)kp} = g^+_{\sigma(k)kp}$, for $k \in K, p \in P$.

*Real variables*

$(l_k, u_k)$ Starting and finishing time of vehicle $k$.
$s_{ik}$ Starting service time of vehicle $k$ at node $i$.
$c_{ik}$ Accumulated cost of vehicle $k$ up to node $i$.
$OC_k$ Total travel cost for vehicle $k$.
$OT_k$ Total travel time for vehicle $k$.

*Model*

$$\text{Minimize} \quad \sum_{k\in K} OC_k + \sum_{k\in K} Cz_k \tag{1}$$

$$\text{subject to} \quad \sum_{t\in T_{\sigma(k)}} w_{kt} \le 2 \qquad k \in K \tag{2}$$

$$\sum_{k\in K_{\sigma(t)}} w_{kt} \le 1 \qquad t \in T \tag{3}$$

$$v_k^{\text{type}} = \sum_{t\in T_{\sigma(k)}} w_{kt} - 1 \qquad k \in K \tag{4}$$

$$u_k \le \tau_t^u + M_T(1 - w_{kt}) \qquad k \in K, t \in T_{\sigma(k)} \tag{5}$$

$$l_k \ge \tau_t^l w_{kt} \qquad k \in K, t \in T_{\sigma(k)} \tag{6}$$

$$v_k^{\text{top}} \le \sum_{t\in T_{\sigma(k)}} c_t^{\text{top}} w_{kt} \qquad k \in K \tag{7}$$

$$v_k^{\text{bel}} \le \sum_{t\in T_{\sigma(k)}} c_t^{\text{bel}} w_{kt} \qquad k \in K \tag{8}$$

$$x_{ijk} \le z_k \qquad i, j \in V, k \in K \tag{9}$$

$$\sum_{t\in T_{\sigma(k)}} w_{kt} \ge z_k \qquad k \in K \tag{10}$$

$$\sum_{i:(i,j)\in E} x_{ijk} - \sum_{i:(j,i)\in E} x_{jik} = 0 \qquad j \in V, k \in K \tag{11}$$

$$\sum_{i:(i,j)\in E} x_{ijk} \le 1 \qquad j \in V, k \in K \tag{12}$$

$$a_i \sum_{j:(j,i)\in E} x_{jik} \le s_{ik} \le b_i \sum_{j:(j,i)\in E} x_{jik} \qquad i \in V, k \in K \tag{13}$$

$$a_{\sigma(k)} z_k \le OT_k \le b_{\sigma(k)} z_k \qquad k \in K \tag{14}$$

$$l_k - M_T(1 - z_k) \le s_{\sigma(k)k} \le u_k + M_T(1 - z_k) \qquad k \in K \tag{15}$$

$$l_k - M_T(1 - z_k) \leq OT_k \leq u_k + M_T(1 - z_k) \quad k \in K \tag{16}$$

$$\sum_{h \in H} h y_{ikh} \leq \frac{s_{ik}}{60} \leq \sum_{h \in H} (h+1) y_{ikh} \quad k \in K, i \in V \setminus \{\sigma(k)\} \tag{17}$$

$$\sum_{h \in H} h y_{\sigma(k)kh} \leq \frac{OT_k}{60} \leq \sum_{h \in H} (h+1) y_{\sigma(k)kh} \quad k \in K \tag{18}$$

$$\sum_{h \in H} y_{ikh} \leq 1 \quad i \in V, k \in K \tag{19}$$

$$\sum_{k \in K} y_{ikh} \leq f_{ih} \quad i \in V, h \in H \tag{20}$$

$$s_{ik} + S + s^s_{ij}\left(1 - v^{\text{type}}_k\right) + s^d_{ij} v^{\text{type}}_k - M_T(1 - x_{ijk}) \leq s_{jk} \quad k \in K, (i,j) \in E | j \neq \sigma(k) \tag{21}$$

$$s_{ik} + S + s^s_{i\sigma(k)}\left(1 - v^{\text{type}}_k\right) + s^d_{i\sigma(k)} v^{\text{type}}_k - M_T(1 - x_{i\sigma(k)k}) \leq OT_k \quad i \in V, k \in K \tag{22}$$

$$\sum_{k \in K} g^+_{ikp} \leq n^+_{ip} \quad i \in V, p \in P \tag{23}$$

$$\sum_{k \in K} g^-_{ikp} \geq n^-_{ip} \quad i \in V, p \in P \tag{24}$$

$$g^+_{ikp} \leq n^+_{ip} \sum_{j:(j,i) \in E} x_{jik} \quad i \in V, k \in K, p \in P \tag{25}$$

$$g^-_{ikp} \leq n^-_{ip} \sum_{j:(j,i) \in E} x_{jik} \quad i \in V, k \in K, p \in P \tag{26}$$

$$q_{ikp} + g^+_{jkp} - g^-_{jkp} - M_L(1 - x_{ijk}) \leq q_{jkp} \quad i \in V, k \in K, p \in P, j \in V \setminus \{\sigma(k)\} \tag{27}$$

$$q_{ikp} + g^+_{jkp} - g^-_{jkp} + M_L(1 - x_{ijk}) \geq q_{jkp} \quad i \in V, k \in K, p \in P, j \in V \setminus \{\sigma(k)\} \tag{28}$$

$$\sum_{p \in P} \frac{q_{ikp}}{o_p} \leq v^{\text{top}}_k + v^{\text{bel}}_k \quad i \in V, k \in K \tag{29}$$

$$\sum_{p \in P_N} \frac{q_{ikp}}{o_p} \leq v^{\text{bel}}_k + \sum_{t \in T_{\sigma(k)} \cap T^e} c^{\text{top}}_t w_{kt} \quad i \in V, k \in K \tag{30}$$

$$c_{ik} + c^s_{ij}\left(1 - v^{\text{type}}_k\right) + c^d_{ij} v^{\text{type}}_k - M_C(1 - x_{ijk}) \leq c_{jk} \quad i \in V, k \in K, j \in V \setminus \{\sigma(k)\} \tag{31}$$

$$c_{ik} + c^s_{i\sigma(k)}\left(1 - v^{\text{type}}_k\right) + c^d_{i\sigma(k)} v^{\text{type}}_k - M_C(1 - x_{i\sigma(k)k}) \leq c_{jk} \quad i \in V, k \in K \tag{32}$$

$$w_{kt} = 0 \quad k, t | \sigma(k) \neq \sigma(t) \tag{33}$$

$$g^-_{\sigma(k)kp} = 0 \quad k \in K, p \in P \tag{34}$$

$$q_{\sigma(k)kp} = g^{+}_{\sigma(k)kp} \qquad k \in K, p \in P \tag{35}$$

$$w_{kt}, y_{ikh}, z_k \in \{0,1\} \qquad i \in V, t \in T, k \in K, h \in H \tag{36}$$

$$x_{ijk} \in \{0,1\} \qquad (i,j) \in E, k \in K \tag{37}$$

$$v^{\text{type}}_k \in \{-1,0,1\} \qquad k \in K \tag{38}$$

$$v^{\text{top}}_k, v^{\text{bel}}_k, g^{-}_{ikp}, g^{+}_{ikp}, q_{ikp} \in Z^{+} \qquad i \in V, k \in K, p \in P \tag{39}$$

$$l_k, u_k, s_{ik}, c_{ik} \geq 0 \qquad i \in V, k \in K \tag{40}$$

Here $M_L$, $M_T$, and $M_C$ are large enough big-$M$ constants to make the corresponding constraints redundant. The objective is to minimize the sum of vehicle routing cost and vehicle usage cost (1). Constraints (2) limit the number of trailers that may be assigned to a vehicle. Constraints (3) ensure that a single trailer can be assigned to at most one vehicle. Constraints (4) are used to identify the type of vehicle configuration. Constraints (5)-(6) set the vehicle time windows. Constraints (7) and (8) set the vehicle capacity on top and bottom compartments, respectively. Constraints (9) and (10) set the relationship between the $z_k$ and the $x_{ijk}$ and $w_{kt}$ variables, respectively. Flow balance is assured by (11). Constraints (12) ensure that a vehicle may visit a node at most once. Constraints (13)-(14) and (15)-(16) guarantee the time windows for both customers and vehicles, respectively. Constraints (17)-(19) set the correct relationship between the binary time index variables $y_{ijk}$ and the time variables $s_{ik}$ and $OT_k$. The dock capacity constraints are given by (20). Constraints (21) and (22) are used to ensure that the time variables are consistent with travel and service times. Product availability and demand are set by (23) and (24), respectively. Constraints (25) and (26) ensure that a vehicle can load or unload product at node $i$ only if it visits that node. Constraints (27)-(28) establish the connection on vehicle load between nodes $i$ and $j$ when vehicle $k$ indeed travels directly from $i$ to $j$. Vehicle capacity is set by (29) and (30), the latter considering that type N products can be placed in the bottom compartment of any vehicle or in the top compartment of vehicles with a shelf. Constraints (31)-(32) ensure the consistency of the cost variables. Conditions (33) make it impossible to assign trailers to vehicles in different starting locations. Constraints (34) and (35) establish initial delivery and pickup conditions for a vehicle at the start of its route. The nature of the decision variables are set in (36)-(40).

The problem is NP-hard [2]. The largest instance we could solve exactly after several hours of computing time by CPLEX branch-and-bound algorithm was in the order of 6 nodes, 7 trailers, and 5 products. Given real-world instances are significantly larger, we develop a heuristic framework for this problem.

## 3 Proposed Heuristic

GRASP [5] is a multi-start metaheuristic that has been widely used for finding good quality solutions to many hard combinatorial optimization problems. It relies on greedy randomized constructions and local search.

The proposed method is depicted in Procedure 1. The problem involves many decisions at different levels. Thus, to make the problem more tractable a decomposition approach is taken. The main idea is first to estimate point-to-point requests (done in Step 5), and then, based on this information, to apply the iterative GRASP phases of construction (Step 7) and local search (Step 8). Each of these components is described next.

**Procedure 1.** GRASP( $\Delta\beta$, $\alpha$, *limit_iter* )

**Input:** $\Delta\beta$, step parameter for cost matrix; $\alpha$, RCL quality parameter; *limit_iter*, number of iterations
**Output:** $X_{\text{best}}$, best solution found
1: $X_{\text{best}} \leftarrow \emptyset$; $f(X_{\text{best}}) \leftarrow +\infty$
2: *find_shortest_paths*$(G, C^s, C^d, S^s, S^d)$
3: $\beta \leftarrow 0$
4: **while** ( $\beta \leq 1$ ) **do**
5: $R \leftarrow$ *solve_TP*$(\beta)$
6: **for** ( *iter* = 1 to *limit_iter* ) **do**
7: $X \leftarrow constructSolution(\alpha, R)$
8: $X \leftarrow localSearch(X)$
9: **if** ( $X$ is better than $X_{\text{best}}$ ) **then**
10: $X_{\text{best}} \leftarrow X$
11: **end if**
12: **end for**
13: $\beta \leftarrow \beta + \Delta\beta$
14: **end while**
15: **return** $X_{\text{best}}$

*Preprocessing: Construction of the optimal cost and time matrices:* Initially we have an undirected graph where the edges handle two types of costs ($c^s_{ij}$ and $c^d_{ij}$) and two types of times ($s^s_{ij}$ and $s^d_{ij}$), for single and double vehicles. In this preprocessing phase (Step 2 of Procedure 1), we find optimal shortest paths between all pairs of nodes for each of the four matrices by applying the well-known Floyd-Warshall algorithm. Let $c^s_{[ij]}$ and $c^d_{[ij]}$ be the cheapest cost of traveling from node $i$ to node $j$ using a single and double vehicle, respectively. Similarly, let $s^s_{[ij]}$ and $s^d_{[ij]}$ represent the shortest time of traveling from $i$ to $j$ for a single and a double vehicle, respectively. This information on optimal paths is used in other components of the algorithm and needs to be computed only once.

*Decomposition: Point-to-point request generation:* As mentioned before, to make the problem more tractable we first attempt to estimate point-to-point requests. Each request or order is identified by a vector $(i, j, p, r_{ijp})$, where $r_{ijp}$ is the amount of product $p$ (measured in number of 12-bottle boxes) to be picked up at $i$ and delivered to $j$. To compute these requests, we solve a transportation problem, where we take as input the information on pickup and delivery quantities at every node (given by parameters $n^+_{ip}$ and $n^-_{ip}$, respectively), and the "cost" between nodes $i$ and $j$. Now, we must bear in mind that this cost depends on whether

single or double vehicles are used which is unknown at this point. However, we can construct a cost function as a convex combination of the optimal costs for single and double vehicles parameterized by a weight $\beta \in [0,1]$. The output, for a fixed value of $\beta$ is a set of requests $R$.

Let $y_{ij}$ a binary-decision variable equal to 1 if at least one request from $i$ to $j$ exists and 0 otherwise. Then, the transportation problem TP($\beta$) given below is solved in Step 5 of Procedure 1. Here, $M = \max_{i \in V, p \in P}\{n_{ip}^+\}$ is a large enough constant to make the constraint (44) redundant when $y_{ij} = 1$. The problem is not quite a classical transportation problem. Here we have a fixed cost rather than a variable cost. Also, we have several capacity/demand constraints for the different products. However, time is not an issue since this problem is easy to solve even for large instances. Note that different values of $\beta$ give rise to different cost structure among nodes, and therefore, we proceed to generate different TPs for a variety of values of $\beta$. As can be seen in Procedure 1 we use a step size $\Delta\beta$ that allows us to discretize the range for $\beta$ and try out different matrices and have therefore more diversity.

*Model TP($\beta$)*

$$\text{Minimize} \quad f(r,y) = \sum_{i,j \in V} \left(\beta c^s_{[ij]} + (1-\beta) c^d_{[ij]}\right) y_{ij} \tag{41}$$

$$\text{subject to} \quad \sum_{j \in V} r_{ijp} \le n^+_{ip} \qquad i \in V, p \in P \tag{42}$$

$$\sum_{i \in V} r_{ijp} \ge n^-_{jp} \qquad j \in V, p \in P \tag{43}$$

$$\sum_{p \in P} r_{ijp} \le M y_{ij} \qquad i,j \in V \tag{44}$$

$$r_{ijp} \ge 0 \qquad i,j \in V, p \in P \tag{45}$$

$$y_{ij} \in \{0,1\} \qquad i,j \in V \tag{46}$$

*Construction phase:* Given a set of requests $R$, this phase is where these requests are assigned to vehicles, and, as a consequence, where routes are determined for these vehicles. The construction procedure is depicted in Procedure 2.

The procedure starts by initializing the set of vehicle routes $X_\pi = \{\pi_k | k \in K\}$ as the empty set (Step 1) and by assigning each trailer $t \in T$ to a vehicle $k \in K_{\sigma(t)}$ (Step 2), that is, initally we have single vehicles. In a later stage we consider merging two single vehicles into one double vehicle. Then a set $R'$ containing all possible point-to-point pairs that need to be served and a set $\bar{P}$ containing all feasible routes are formed. Here, the latter is defined as $\bar{P} = \{(i,j,k) : (i,j) \in R' \wedge (i,j,k) \text{ is a feasible route}\}$, where the term feasibility means that vehicle $k$ can deliver the products from node $i$ to $j$ within the vehicle and node time windows. Let $P_k$ be the set of all feasible routes associated with vehicle $k$. Then, the actual route construction takes place within the *while* loop (Steps 5-11). Following the GRASP philosophy, a greedy function that measures the cost of assigning pair $(i,j)$ to vehicle $k$ is computed as follows:

**Procedure 2.** constructSolution( $\alpha$, $R$ )

```
1: X_π ← ∅
2: K ← T
3: R' = {⟨i, j⟩ : (i, j, p, r_ijp) ∈ R}
4: P̄ ← getFeasiblePaths(K, R')
5: while ( R ≠ ∅ ) do
6:       φ^min ← min{φ(i, j, k) : (i, j, k) ∈ P̄}
7:       φ^max ← max{φ(i, j, k) : (i, j, k) ∈ P̄}
8:       RCL = {(i, j, k) ∈ P̄ : φ(i, j, k) ≤ φ^min + α(φ^max − φ^min)}
9:       (i, j, k) ← chosen randomly from RCL
10:      (X_k, R) ← assignProducts(i, j, k, R, X)
11: end while
12: returnToDepot(X_Π)
13: return X
```

$$\varphi^{\delta}(i,j,k) = c^{\delta}_{[u(k)i]} + c^{\delta}_{[ij]} + c^{\delta}_{j\sigma(k)]}$$

where $\delta = s(d)$ if $k$ is a single (double) vehicle. This function estimates the cost of traveling from the vehicle $k$ current location $u(k)$ to node $i$ then to $j$ and then back to its depot $\sigma(k)$. Then, a restricted candidate list (RCL) is built (Step 9) with those elements whose greedy function value falls within $\alpha$ % of the best possible value. An element from RCL is randomly chosen. Step 10 performs the assignment of route $(i, j)$ to vehicle $k$, and figures out the amount of product to be loaded by first assigning the non-returnable items without exceeding the non-returnable capacity, and then filling the vehicle with the returnable products, performing the necessary time updating for vehicle $k$ and remaining orders to be served $R$. By proceeding this way, we guarantee that the product type priority rule (i.e., not placing type N products on top of type R in vehicles with no shelf) is met. Finally, in Step 12, we make sure that every vehicle returns from its current location to its depot.

*Improvement Phase:* A solution delivered by the construction phase might not necessarily satisfy the dock capacity constraints (20). Therefore, the goal of the local search is to attempt to improve the quality of the solution or to repair infeasibility if needed. The proposed improvement phase (depicted in Procedure 3) includes three different methods, which are described next.

**Procedure 3.** localSearch()

```
1: mergeVehicles()
2: transferShipment()
3: repairDocks()
4: return
```

**Method 1: Merging single vehicles.** The idea behind this method is to identify those pairs of single vehicles that use the same route and merge them

into a double vehicle (see Figure 1). This step is done only if feasibility with respect to the time window constraint is kept and the resulting cost of the merge is less than the sum of the cost of both single vehicles.

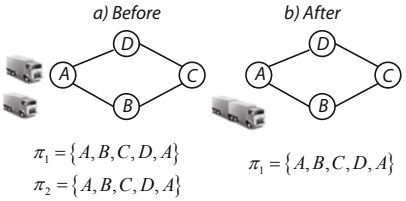



**Fig. 1.** Merging two single vehicles into one double vehicle

**Method 2: Transfering loads between vehicles.** The idea behind it is to analyze vehicles (either single or double) emanating from the same plant and try to combine both routes into a single route by one of the vehicles. This move is allowed only if the vehicles to be combined are of the same type and the resulting route is feasible with respect to the time window constraints. The method is depicted in Procedure 4, where $u$ and $v$ denote the vehicles to be combined with corresponding routes $\pi_u = \{u_1, u_2, \ldots, u_{\bar{u}}\}$ and $\pi_v = \{v_1, v_2, \ldots, v_{\bar{v}}\}$, respectively. Here, the neighborhood of possible moves is $N_2$ (Step 1).

Note in Steps 5-9, when combining these routes $\pi_u$ and $\pi_v$, if vehicle $u$ has still some load that must be delivered back to its depot $\sigma(u)$ or if vehicle $v$ must pickup some load in $\sigma(k)$ then the resulting combined route would have to go through the depot resulting in $\pi_{u'} = \{u_1, \ldots, u_{\bar{u}}, v_2, \ldots, v_{\bar{v}}\}$; otherwise, the combined vehicle may skip the depot to have $\pi_{u'}$ as in Step 8.

**Procedure 4.** transferShipment()

1: $N_2 = \{(k_1, k_2) \in K \times K : \sigma(k_1) = \sigma(k_2) \wedge v_{k_1}^{\text{type}} = v_{k_2}^{\text{type}}\}$
2: **while** ( $|N_2| > 0$ ) **do**
3: $(u, v) \leftarrow$ chose one element of $N_2$; $N_2 \leftarrow N_2 \setminus \{(u, v)\}$
4: $\pi_u = \langle u_1, \ldots, u_{\bar{u}} \rangle$; $\pi_v = \langle v_1, \ldots, v_{\bar{v}} \rangle$
5: **if** $\left( \sum_{p \in P} g^-_{\sigma(u)up} > 0 \vee \sum_{p \in P} g^+_{\sigma(v)vp} > 0 \right)$ **then**
6: $\pi_{u'} = \langle u_1, \ldots, u_{\bar{u}}, v_2, \ldots, v_{\bar{v}} \rangle$
7: **else**
8: $\pi_{u'} = \langle u_1, \ldots, u_{\bar{u}-1}, v_2, \ldots, v_{\bar{v}} \rangle$
9: **end if**
10: **if** ( $\pi_{u'}$ is a feasible route path and $f(\pi_{u'}) < f(\pi_u) + f(\pi_v)$ ) **then**
11: $\pi_u \leftarrow \pi_{u'}$
12: $K \leftarrow K \setminus \{v\}$
13: $N_2 \leftarrow N_2 \setminus \{(k_1, k_2) : k_1 = v \vee k_2 = v\}$
14: **end if**
15: **end while**
16: **return**

**Method 3: Repairing dock infeasibility.** The solution delivered after applying the previous methods may not satisfy all dock capacity constraints. In that case, this method attempts to repair this possible infeasibility by making the necessary time shift adjustments. To perform this task, we define $J(h,i) = \{k \in K : \lfloor s_{ik}/60 \rfloor\}$ as the set of vehicles that arrive at node $i$ during time index $h$ (measured in hours). We now can determine the set of conflict nodes, that is, those nodes where the dock capacity constraints are violated at time $h$, as $\Psi(h) = \{i \in V : f_{ih} < |J(h,i)|\}$.

The method works as follows. Starting from the lowest value of $h$ for which there are conflicts, we first choose one conflicting node $i$. Now, for this node we must determine what is the least critical vehicle, that is, the vehicle that possesses the largest amount of time adjustment flexibility. To do this, we compute for each vehicle the maximum amount of time that can be added to this vehicle without violating the future time window constraints in the remainder of its route $(i, \ldots, \bar{k})$ as follows

$$\Upsilon(i,k) = \min_{u \in (i,\ldots,\bar{k})} \{b_u - s_{uk}\}.$$

Then the vehicle with the largest possible value of $\Upsilon(i,k)$ is chosen and its future arrival times at every node in its remaining route are updated. By doing this, we guarantee that no more conflicting nodes for any time earlier than $h$ arise, and this time adjustment may bring the current vehicle $k$ at node $i$ back into feasibility. The method is depicted in Procedure 5. If no vehicle can be found, the current iteration stops reporting an infeasible solution.

**Procedure 5.** repairDocks()

```
for h = 0, ..., |H| do
    while ( Ψ(h) ≠ ∅ ) do
        i ← choose one element of Ψ(h) arbitrarily
        while ( |J(h,i)| > f_ih ) do
            k* ← arg max_{k∈J(h,i)} {Υ(i,k)}
            for all (u ∈ [i, ..., ℓ_n]) do
                s_uk* = s_uk* + 60 − (s_ik* − 60 · ⌊s_ik*/60⌋)
            end for
        end while
    end while
end for
return
```

## 4 Empirical Results

The procedures were implemented in C# on Visual Studio 2010 environment. All experiments were performed on an Intel Core i7 computer running the Windows 7 operating system. CPLEX 12.1 was used for solving the transportation

subproblem. For each of the three data sets described below 20 instances were randomly generated based on distributions from real-world instances provided by the industrial partner.

**Set A (small-size instances):** 10 nodes, 20 trailers, 10 products.
**Set B (medium-size instances):** 15 nodes, 50 trailers, 40 products.
**Set C (large-size instances):** 25 nodes, 150 trailers, 300 products.

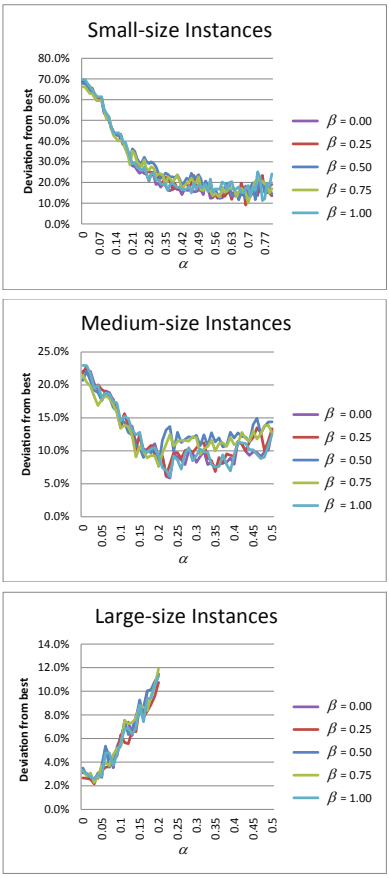



**Fig. 2.** Algorithm behavior as a function of $\alpha$ and $\beta$

*Fine-Tuning of Parameters*
The first experiment aims at finding appropriate values for algorithmic parameters $\alpha$ and $\beta$. To this end, we run a full two-factorial design. Parameter $\beta$ was fixed at values in the [0.00, 1.00] range with increments of 0.25. Parameter $\alpha$ was fixed at values in the range [0.00, 0.80], [0.00, 0.50], and [0.00, 0.20] for the small-, medium-, and large-size instances, respectively, with increments of 0.01.

This results in 15,300 runs. The iteration limit was set at 1000 iterations for each run.

Figure 2 plots the results for the different values of $\alpha$ and $\beta$. The vertical axis indicate the average relative deviation from the best known feasible solution for each instance. One of the first observations we can make from the plot is that the parameter $\alpha$ has more influence over the solution quality than $\beta$ does. This of course is a normal behavior in many GRASP implementations; however, we can also observe that the best possible value of this $\alpha$ depends on the instance size. For the smaller instances, the best solutions were obtained when $\alpha$ lies around 0.7. As the size of the instances grows, we can see how this best value of $\alpha$ goes down to around 0.20 for the medium-size instances, and 0.03 for the large-size instances. This means that reducing diversity in favor of more greedy-like solutions as instances get large provides a better strategy for finding solutions of better quality. For the remainder of the experiments we fixed $\alpha$ at 0.70, 0.20, and 0.03 for the small-, medium-, and large-size instances, respectively.

In the following experiment, we try to determine the effect that the step size $\Delta\beta$ has on algorithmic performance. While it is true that providing a finer discretization (i.e., reducing the step size $\Delta\beta$) could lead to more and better solutions, it is also true that the computational burden increases. Moreover, when the step size is sufficiently small, solutions obtained from different values of $\beta$ will look alike, resulting in very little gain. Therefore, the purpose of this experiment is to investigate this effect. To this end, we run the heuristic fixing $\Delta\beta$ at different values as $\Delta\beta = 1/x$ with $x \in \{1, 2, \ldots, 10\}$. The iteration limit was set at 1000.

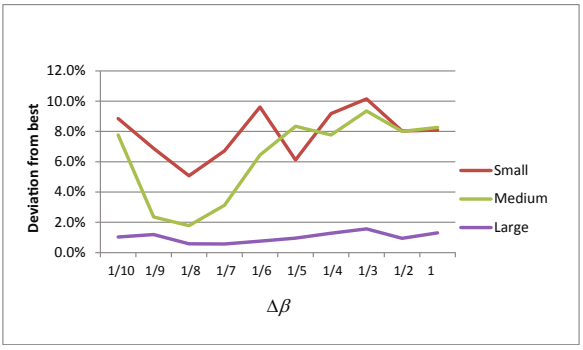



**Fig. 3.** Algorithm behavior as a function of $\Delta\beta$

Figure 3 displays the results plotting in the vertical axis the average deviation from the best known solution for every instance. The horizontal axis shows the different values of $\Delta\beta$. As can be seen from Figure 3, for the large-size instances the choice of $\Delta\beta$ did not have much impact; however, a slight improvement is observed at values around 1/8, which in fact matches the best value found for the small- and medium-size instances.

*Feasibility Analysis*
There is not guarantee that, in a given iteration of the algorithm the final solution delivered be feasible. Therefore it becomes important to assess the success rate of the heuristic in terms of its feasibility. To this end, we proceed to compute the percentage of times a feasible solution was obtained per iteration, and, given the algorithm returns the best solution found over all iterations, the percentage of times the algorithm delivered a feasible solution.

Table 1 shows the results for the different data sets. Columns 2-3 indicate the average and maximum success rate per iteration over all instances tested. The last column shows the success rate of the entire heuristic. As can be seen, even though some individual iterations may fail, it is empirically observed that the heuristic is always successful.

**Table 1.** Feasibility success rate (%)

| | Per iteration | | Per algorithm |
|---|---|---|---|
| Data set | Ave | Max | execution |
| Small | 99.6 | 100.0 | 100.0 |
| Medium | 68.0 | 100.0 | 100.0 |
| Large | 60.0 | 100.0 | 100.0 |

*Comparison with Current Practice*
Finally, we present a comparison between the solution found by our heuristic and the solution reported by the firm in a particular case study.

**Table 2.** Comparison between current practice and proposed heuristic

| | NV | NVS | NVD | NT | RC | FC | Total cost |
|---|---|---|---|---|---|---|---|
| Firm solution | 47 | 9 | 38 | 85 | $186,018 | $70,500 | $256,518 |
| Heuristic solution | 50 | 28 | 22 | 72 | $167,020 | $75,000 | $242,020 |

Table 2 shows this comparison itemizing the individual costs and vehicle usage, where NV, NVS, NVD, and NT stand for number of vehicles, single vehicles, double vehicles, and trailers used, respectively. RC and FC are the routing and fixed cost, respectively, and the last column is the total solution cost. The reduction of the heuristic solution is about 6%. It can be seen this is due in great deal to a better arrangement of the single and double vehicles. The new solution uses 3 more vehicles; however, the main difference comes from the number of single- and double-vehicles used. The new solution uses more single-vehicles which yield lower traveling costs overall. It is clear the contrast with the current practice where it was firmly believed that using fewer vehicles (i.e, more double vehicles) would be a better choice.

## 5 Conclusions

In this paper we studied a vehicle routing problem with pickup and delivery from a real-world application in a bottled-beverage distribution company. Given

the inherent problem difficulty, we have proposed a metaheuristic framework to obtain feasible solutions of good quality. The heuristic and its components were empirically assessed over a wide range of instances. When compared to current practice, it was found that the proposed method obtained solutions of better quality in very reasonable times. The different local search neighborhoods explored in this study can be improved if they can be cast into a more sophisticated local search engine such as tabu search or variable neighborhood search, for instance. This is a subject of follow-up work.

**Acknowledgements.** This work was improved thanks to the remarks by two anonymous reviewers. This work was supported by the Mexican Council for Science and Technology (CONACYT), grant CB2011-1-166397, and UANL under its Scientific and Technological Research Support Program, grants IT511-10, CE728-11, and HU769-11. The research of the third author has been funded by a scholarship for graduate studies from CONACYT.

## References

1. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: A classification scheme and survey. TOP 15(1), 1–31 (2007)
2. Castrillón-Escobar, A.: Una Metaheurística para un Problema de Carga y Descarga en la Repartición de Bebidas Embotelladas. Master thesis, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Mexico (March 2013) (in Spanish)
3. Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., Vogel, U.: Vehicle routing with compartments: Applications, modelling and heuristics. OR Spectrum 33(4), 885–914 (2011)
4. Drexl, M.: Applications of the vehicle routing problem with trailers and transshipments. European Journal of Operational Research 227(2), 275–283 (2013)
5. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133 (1995)
6. Mitrović-Minić, S., Laporte, G.: Waiting strategies for the dynamic pickup and delivery problem with time windows. Transportation Research 38(7), 635–655 (2004)
7. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems, part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58(2), 81–117 (2008)
8. Qu, Y., Bard, J.F.: The heterogeneous pickup and delivery problem with configurable vehicle capacity. Transportation Research Part C: Emerging Technologies 32(1), 1–20 (2013)
9. Ropke, S., Cordeau, J.F.: Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43(3), 267–286 (2009)
10. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4), 455–472 (2006)

# TSP with Multiple Time-Windows and Selective Cities

Marta Mesquita[1], Alberto Murta[2], Ana Paias[3], and Laura Wise[2]

[1] CIO and ISA, UTL,Tapada da Ajuda, 1349-017 Lisboa, Portugal
martaoliv@isa.utl.pt
[2] IPMA, Portuguese Institute of the Sea and Atmosphere, Av.Brasília, 1449-006 Lisboa, Portugal
{amurta,lwise}@ipma.pt
[3] CIO and DEIO, Faculdade de Ciências, Universidade de Lisboa, C6, Piso 4, 1749-016 Lisboa, Portugal
ampaias@fc.ul.pt

**Abstract.** We address a special TSP in which the set of cities is partitioned into two subsets: mandatory cities and selective cities. All mandatory cities should be visited once within one of the corresponding predefined multiple time windows. A subset of the selective cities, whose cardinality depends on the tour completion time, should be visited within one of the associated multiple time windows. The objective is to plan a tour, not exceeding a predefined number of days, that minimizes a linear combination of the total traveled distance as well as the total waiting time. We present a mixed integer linear programming (MILP) model for the problem and propose a heuristic approach to solve it. Computational experiments address two real world problems that arise in different practical contexts.

**Keywords:** Traveling salesman problem, multiple time windows, mixed integer linear programming, branch-and-bound, heuristics.

## 1 Introduction

This paper addresses a special case of the traveling salesman problem, namely the traveling salesman problem with selective cities and multiple time windows (TSPSNTW), motivated by two real world applications. A traveling salesman must plan a tour, not exceeding a maximum number of days, to visit a set of clients spread over a set of cities, with known geographic locations. There are two types of cities: mandatory cities and selective cities. For mandatory cities the visit is compulsory and must be scheduled according to the availability of the corresponding clients. Therefore, a duration time for the visit and one time window for each day of the planning horizon are associated with each mandatory city. After a mandatory city visit is completed, the salesman proceeds to the next one or must be forced to rest. Selective cities represent potential locations to rest or refueling. According to a predefined periodicity a visit to one selective city

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 158–172, 2013.

must occur. Consequently, the number of selective cities to be included in the tour depends on its completion time. Time windows defining the available rest and refueling periods are associated with each selective city. The objective is to design a tour, starting and ending at the home city, that minimizes a linear combination of the total traveled distance (time) as well as the total waiting time to start the visits.

Two practical applications of this problem are referred to in this paper. The first application arises in the design of survey trips for a research vessel to estimate the abundance of fish species. Sampling operations are made at predefined locations along the Portuguese coast. The vessel route starts and ends at the port of Lisbon and should include some regular visits to a port. The second application consists of designing the route of a sales representative that needs to visit his clients spread all over Portugal. Client visits are scheduled from Monday to Friday and the sales representative returns home for the weekend.

There are several well known variants of the TSP in which not all clients need to be visited (see [1] for a survey on TSP variants). In the Generalized TSP (GTSP), cities are partitioned into clusters and the objective is to find the minimum cost circuit visiting at least one city of each cluster. If we consider that each mandatory city defines a cluster and all the selective cities are included in an additional cluster then the TSPSNTW can be seen as variant of the GTSP with time windows and additional constraints where each 'mandatory' cluster is visited once and the 'selective' cluster is visited more than once according to a predefined periodicity. In the Orienteering Problem each node has associated a non-negative weight. The objective is to find a path, whose total traveled time does not exceed a given threshold value, visiting the subset of cities that maximizes the total weight. When the starting and final locations of the path are the same, the problem looks for a circuit and is usually called Selective TSP. In [2] the authors developed different classes of valid inequalities and used a branch-and-cut algorithm to solve a Selective TSP that includes a subset of compulsory cities. If the goal is to find $P$ paths (circuits), each limited in time by a predefined threshold value, the problem is known as the Team Orienteering Problem (see [9] for a recent survey on the Orienteering Problem). Problem TSPSNTW has some similarities with extensions of the aforementioned problems that include time windows. Heuristic algorithms to solve the (Team) Orienteering Problem with time windows have recently been proposed by [8,7,4]. In particular, in [7] the authors propose a VNS procedure to solve a multi-period orienteering problem with multiple time windows where mandatory clients should be visited exactly once and potential customers can be visited at most once.

The outline of this paper is as follows. Section 2 describes the underlying problem as a traveling salesman problem with special nodes, time windows and additional constraints and presents a mathematical programming formulation. Section 3 is devoted to the solution methodology. In Section 4, two real life applications are described and computational results for different scenarios are reported and discussed. Finally, some conclusions are drawn in Section 5.

## 2 Mathematical Programming Formulation

Given a planning horizon $H$, a set of $n$ mandatory cities and a set of $p$ selective cities, the TSPSNTW, consists of determining a circuit that minimizes an overall time function, that depends on the traveled distance and the waiting time before starting each visit, and satisfies the following constraints:

- it starts, at the home city, at day 1 and ends, at the home city, before the $|H|$th day.
- visits each mandatory city exactly once within one of the required time windows. If the traveling salesman arrives at a city location before the corresponding time window lower bound, then he has to wait before starting the sales operation. The duration of the sales operation depends on the city being visited.
- according to a predefined periodicity, a selective city must be visited within one of the corresponding time windows. The duration of such visits is set out a priori.

More formally, consider a directed graph $G = (V, A)$. The set $V = E \cup P$ is the vertex set where each vertex $i \in E = \{1, ..., n\}$ corresponds to a mandatory city and each vertex $i \in P = \{0, n+1, ..., n+p, n+p+1\}$ corresponds to a selective city or to the home city. The home city has been split into two vertices: 0 and $n+p+1$ which correspond, respectively, to the starting and ending location of the route. The arc set $A$ includes the arcs $\{(0,i), i \in E\} \cup \{(i,j) : (i,j) \in E \times E, i \neq j\} \cup \{(i,j) : (i,j) \in E \times (P - \{0\})\} \cup \{(i,j) : (i,j) \in (P - \{n+p+1\}) \times E\} \cup \{(i, n+p+1) : i \in E\}$. In any feasible solution, each vertex associated with a mandatory city has degree two, each vertex corresponding to the home city has degree one and the remaining vertices, corresponding to selective nodes, have degree 0 or 2.

A travel time $t_{i,j}$, indicating the time spent to travel from location $i$ to location $j$, is associated to each arc $(i,j) \in A$. Note that travel times need not satisfy the triangle inequality and need not be symmetric.

For each day $h = 1, ..., |H|$ a time window $[e^h, l^h]$ is defined to ensure that, for each city visited on day $h$, the sales operation starts within the times $e^h$ and $l^h$ and occurs before the end of the $|H|$th day. Each vertex $i \in E$ has to be visited in exactly one of the $|H|$ time windows.

For each of the predefined time periods where the traveling salesman needs to reach a selective city, $s = 1, ..., |S|$, time windows $(e^{|H|+s}, l^{|H|+s})$ are given. For each vertex $i \in E \cup P$ the time spent during the corresponding visit is given and denoted by $prof_i$.

To model the problem we consider three types of decision variables: route variables, time window variables and time variables. As for the route variables, let $x_{i,j}$=1 if the salesman travels directly from city $i$ to city $j$ and 0 otherwise.

Two sets of decision variables are used to represent the time windows. One set defines the time interval for the visit to each mandatory city. That is, $\delta_i^h$=1 if city $i \in E$ is visited during time window $h, h = 1, ..., |H|$ and 0 otherwise.

The other set defines the time interval for the visit to selective nodes. More precisely, $\gamma_i^s$=1 if selective node $i$ is visited during time window $s$, $s = 1, ..., |S|$ and 0 otherwise. Continuous time decision variables, $w_i$ and $e_i$, $i \in E \cup P$, indicate, respectively, the starting time of the operation and the waiting time before starting the operation in city $i$. Let $y$ represent the number of selective cities that the traveling salesman must visit. Variable $y$ depends on a predefined periodicity $d$ and on the value of variable $w_{n+p+1}$ that defines the arrival time to the home city at the end of the route.

Problem TSPSNTW can be described through a MILP formulation where the objective function is the minimization of a linear combination of two terms: the total traveled time and the total waiting time.

$$min \;\; \lambda_1 \sum_{(i,j)\in A} t_{ij}\; x_{ij} + \lambda_2 \sum_{i\in V} e_i \tag{1}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \quad j \in E \cup \{n+p+1\}, \tag{2}$$

$$\sum_{j\in E} x_{0,j} = 1 \tag{3}$$

$$\sum_{i:(i,j)\in A} x_{ij} - \sum_{i:(j,i)\in A} x_{ji} = 0, \quad j \in E \cup (P - \{0, n+p+1\}), \tag{4}$$

$$\sum_{h=1}^{|H|} \delta_i^h = 1 \quad i \in E \tag{5}$$

$$\sum_{h=1}^{|H|} e^h \delta_i^h \le w_i \le \sum_{h=1}^{|H|} l^h \delta_i^h \quad i \in E \tag{6}$$

$$\sum_{s=1}^{|S|} \gamma_j^s \le 1 \quad j \in (P - \{0, n+p+1\}) \tag{7}$$

$$\sum_{s=1}^{|S|} e^{|H|+s} \gamma_j^s \le w_j \le \sum_{s=1}^{|S|} l^{|H|+s} \gamma_j^s \quad j \in (P - \{0, n+p+1\}) \tag{8}$$

$$\sum_{s=1}^{|S|} \sum_{j\in(P-\{0,n+p+1\})} \gamma_j^s - y = 0 \tag{9}$$

$$y \ge \frac{w_{n+p+1}}{d} - 1 \tag{10}$$

$$\sum_{s=1}^{|S|} \gamma_j^s = \sum_{i:(i,j)\in A} x_{i,j} \quad j \in (P - \{0, n+p+1\}) \tag{11}$$

$$w_0 = 0 \tag{12}$$

$$w_{n+p+1} \ge w_j \quad (j, n+p+1) \in A \tag{13}$$

$$w_j = w_i + prof_i + \overline{t}_{ij} + e_j - M(1 - x_{ij}) \quad (i,j) \in A \tag{14}$$

$$x_{ij} \in \{0,1\} \quad (i,j) \in A \tag{15}$$

$$w_i \geq 0, \quad i \in E \cup P \tag{16}$$

$$\delta_i^h \in \{0,1\} \quad i \in E \tag{17}$$

$$\gamma_i^s \in \{0,1\} \quad i \in P \tag{18}$$

$$y \in \mathbb{N}_0 \tag{19}$$

Coefficients $\lambda_1$ and $\lambda_2$ allow us to attribute different weights to the components of the objective function. Note that, when $\lambda_2$ is equal to zero the objective function represents the total traveled distance while when both $\lambda_1$ and $\lambda_2$ are equal to one the objective function represents the completion time. Constraints (2) state that the traveling salesman enters once in each mandatory city as well as in his city home (end). Constraints (3) guarantee that the circuit starts at the home city. Constraints (4) are flow conservation constraints. These constraints together with (2) ensure that the salesman enters and leaves exactly once each mandatory city and each city chosen to be visited among the selective set. Constraints (5) ensure that each mandatory city is visited within exactly one time window while (6) guarantee that the visit occurs within a feasible time window. Constraints (7) state that each selective city is visited at most once and constraints (8) guarantee that when such visit occurs it must be within predefined time windows. Constraints (9) and (10) are related to the number of visits to selective cities. Constraints (11) guarantee the consistency between variables $x_{i,j}$ and variables $\gamma_j^s$. Constraints (12) and (13), respectively, state that the traveling salesman leaves the home city at time instant 0 and returns to it at the end of the circuit. Constraints (14) link variables $x$ and $w$. These constraints establish the precedence relation between two consecutive vertices visited by the traveling salesman and eliminate sub-tours. In detail, constraints (14) state that if the salesman goes from location $i$ to location $j$ then the time at which he starts visiting $j$, $w_j$, must be greater than the time at which he starts visiting $i$, $w_i$, plus the time spent at city $i$, plus a function of the time spent in transit from $i$ to $j$, $\overline{t}_{ij}$ whose expression is given by $\overline{t}_{ij} = t_{ij}$ if $i \in P$ or $\overline{t}_{ij} = \epsilon + t_{ij}$ if $i \in E$. Parameter $\epsilon$ accounts to an extra time due to possible setbacks during a sales operation at a mandatory city.

## 3 Solution Approach

Preliminary computational experiments showed that solving the TSPSNTW with a standard MILP solver tends to spend considerable CPU time, most of it spent at improving near-optimal feasible solutions, and often ends without reaching an optimal solution. This behavior suggests to apply a standard MILP solver, within a predefined time limit, to obtain feasible solutions followed by an heuristic procedure that, taking advantage of some TSPSNTW features, is able to improve the MILP solver solutions in short CPU time.

We propose a three phase algorithm to solve the TSPSNTW. The algorithm starts with a pre-processing phase where clustering techniques are used to reduce

the cardinality of the arc set $A$. In the second phase, model (1)-(19) and a generic MILP solver are used to obtain feasible solutions. In the third phase, an Adaptive Neighborhood Search (ANS) heuristic is used to improve the feasible solutions obtained previously as well as to incorporate some traveling salesman preferences. Different temporal based neighborhoods, that prevent the violation of time window constraints, are considered and combined with removal/insertion operators.

### 3.1 Pre-processing

The directed graph $G = (V, A)$ is complete. However, for each specific application, the geographic location of the cities may suggest that arcs connecting cities $i$ and $j$ whose $t_{ij}$ exceeds a predefined parameter $M$ have little chance of being used in an optimal solution. To reduce the computational complexity of TSPSNTW, we propose a pre-processing phase where clustering techniques are used to decide which arcs may be removed from the arc set A. To partition the cities into cluster sets, we applied function `pam` from Package `cluster` available at CRAN R (http://cran.r-project.org/). The `pam`, "partitioning around medoids", algorithm [3] is based on the search for $\mu$ representative objects or medoids among the observations to be clustered. The algorithm iteratively alternates between the assignment of each observation to a nearest medoid and the re-computation of the set of medoids until medoids stop changing. The number of clusters, $\mu$, to be considered depends on the application under study.

Given matrix $[t_{i,j}]$ and two parameters $tmax_1$ and $tmax_2$ such that $tmax_2 \geq tmax_1$, we have considered that arc $(i, j) \in A$ if and only if $t_{ij} \leq dmax_1$ or ($t_{ij} \leq dmax_2$ and cluster($i$)=cluster($j$)).

### 3.2 Heuristic Branch-and-Bound

In the second phase, the objective is to build feasible solutions. The model presented in Section 2 includes integer variables and branch-and-bound techniques combined with variable fixing strategies are used to obtain optimal/near-optimal solutions from an efficient standard MILP solver. We propose a heuristic branch-and-bound that works in two steps. In the first step, the LP relaxation of model (1) - (19) where (15) is replaced by $x_{ij} \geq 0$ is solved. If the resulting solution is not integer variable fixing strategies are used, within a truncated branch-and-bound algorithm, to obtain a pool of feasible solutions. Consider the following subsets of the decision variables $x_{ij} \geq 0$, whose integrality has been relaxed:

- $XEP$ contains $x_{ij}$ with $i \in E$ and $j \in P$, i.e., decision variables related to the connection from a mandatory city $i$ to a selective city or the home city $j$;
- $XPE$ contains $x_{ij}$ with $i \in P$ and $j \in E$, i.e., decision variables related to the connection from a selective city or the home city $i$ to mandatory city $j$;
- $XEE$ contains $x_{ij}$ with $i, j \in E$, i.e., decision variables related to the connection from mandatory city $i$ to mandatory city $j$;

Each one of these subsets induced a variable fixing strategy. Consider four predefined parameters $\alpha$, $\beta$, $\eta$ and $T$. Taking into account the optimal solution of the LP relaxation, we fix to 1 the values of at most $\alpha$ decisions variables $x_{ij}$ such that $x_{ij} \geq \beta$ and $x_{ij} \in XEP$ or $x_{ij} \in XPE$. Then we run a truncated branch and bound within a time limit of $T$ seconds. If the optimal solution is not reached, a second step is required where some good temporal characteristics of the best feasible solution obtained are explored. In the second step, considering the best feasible solution given by the branch and bound, we identify the daily sequences of consecutive visits to mandatory cities where at least $\eta$ clients are visited. Let $DS = \{ds_1, ..., ds_c\}$ be the set of these daily sequences where $ds_\ell$ contains the indexes of mandatory cities visited in daily sequence $\ell$. For each $\ell = 1, ..., c$, we fix to one $\alpha$ decision variables $x_{ij}$ such that $i, j \in ds_\ell$ and rerun the branch and bound restricted to the time limit of $T$ seconds.

### 3.3 Adaptive Neighborhood Search

Finally, in the third phase an adaptive neighborhood search (ANS) heuristic is used to improve solutions obtained in phase 2. The main concern is to introduce some robustness in the solutions obtained in the previous phase. Robust solutions offer more stability and flexibility to adapt and recover from disruptions that may occur during the journey. In order to obtain solutions that meet some robustness we propose an ANS algorithm that uses four competitive subheuristics to improve the traveled distance, while keeping fixed the number of clients visited in each day.

The proposed ANS is based on the ALNS procedure developed by Pisinger and Ropke in [5] and [6]. At each iteration, a removal heuristic that assigns undefined values to at most $q$ variables is followed by a repair heuristic that re-assigns feasible values to those variables. As noted by several authors, usually simple and fast removal/repair procedures lead to poor quality solutions. However, this disadvantage is offset by the fact that, large moves around the solution space lead to a diverse set of feasible solutions that potentially includes good quality solutions.

The particular structure of the TSPSNTW, with two types of city visits, mandatory or selective, leads to a set of sequences of visits to mandatory cities periodically intercalated (linked) by a visit to a selective city. Or, in a different perspective, visits to selective cities connect sequences of consecutive visits to mandatory cities. Moreover, these sequences of visits take place along a set of days, whose cardinality must not exceed $|H|$. To exploit both the different nature of the cities and the temporal structure we have considered four different removal operators.

- Daily removal - removes a subset of $q$ mandatory cities from a daily sequence of consecutive visits to mandatory cities. The subset to be removed do not include the first and the last mandatory cities visited on that day, in order to preserve connections between different daily sequences.

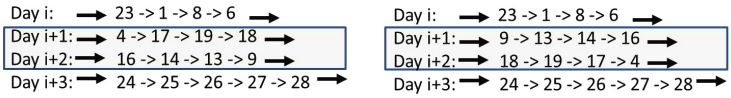



**Fig. 1.** Remove (6,4) and (9,24). Insert(6,9) and (4,24) by reversing the orientation and swapping days i+1 and i+2.

- Time oriented removal - removes a subset of $q$ mandatory cities from daily sequence $i$ and from daily sequence $i+1$. The subset to be removed does not include the first and the last mandatory cities visited, respectively, on day $i$ and day $i+1$, so as to not disrupt the connections between days other than $i$ and $i+1$.
- Route proximity removal - considers two daily sequences, $i$ and $i+k$ (k>1), which are geographically close, removes a subset of $q$ mandatory cities from daily sequence $i$ and $q$ mandatory cities from daily sequence $i+k$.
- Inter day removal - removes two arcs connecting two daily sequences of visits to mandatory cities.

The daily removal, time oriented removal and route proximity removal operators remove subsets of mandatory cities by disconnecting them from the current circuit, leaving partial circuits and/or isolated cities. By removing two arcs, the inter day removal operator is always conducted to three partial circuits in the solution. The daily removal and the route proximity removal only affect daily sequences of consecutive visits to mandatory cities. The time oriented removal steps in both daily sequences and night connections, which link consecutive daily sequences. The inter day removal moves within the set of night connections. The time oriented removal and the route proximity heuristics are based on the idea that the $q$ mandatory cities removed from daily sequence $i$ are easily interchanged with those removed from day $i+k$ due to geographical proximity, while maintaining time windows constraints feasibility.

At each iteration of the algorithm, a neighborhood defining operator is selected, among the set of four operators defined above. The execution of a removal operator leads to a set of partial sub-circuits disconnected from each other. Then, an insertion operator should be applied in order to repair the solution. We have considered that neighborhoods $N_i,\ i = 1, ..., 4$ induced by the removal operators will take care of both the removal and insertion steps. That is, the set of partial sub-circuits will be reconnected among them by a least cost insertion operator. The least cost insertion heuristic looks to sub-circuits without taking into account their orientation. As a consequence it may reverse the orientation of some circuits. This can have some impact while repairing inter day arc removals as illustrated in the example of Figure 1.

The selection of the operators is based on a roulette wheel scheme that represents the past performance of each removal/insertion heuristic. A score $\pi_i$ is associated with each operator $i$. The initialization of the score vector attributes an equal score to each removal/insertion operator. Whenever operator $i$ finds a solution $x'$ better than the current solution $x$ the score $\pi_i$ is updated to increase

the value of the correspondent probability, $\frac{\pi_i}{\sum_{j=1}^{4} \pi_j}$, for being chosen. The new solution $x'$ is accepted if the total traveled distance covered is less than or equal to the total traveled distance covered by the current solution $x$. Note that in cases where all the mandatory cities share the same time windows for all days $h \in H$, the acceptance criteria ensure that the corresponding time window constraints are valid for $x'$, without being necessary to check them in each iteration.

**Algorithm 1.** ANS algorithm pseudo-code

**Require:** $x$ feasible solution
1: **repeat**
2: Use roulette wheel selection based on the scores vector $\pi$ to choose a removal/insert operator inducing neighborhood $N_i$.
3: Consider the heuristics correspondent to the induced neighborhood $N_i$ and obtain a new solution $x'$ from $x$.
4: **if** $vopt(x') < vopt(x)$ **then**
5: $x = x'$
6: **end if**
7: Update the roulette wheel statistic score $\pi_i$ of neighborhood $N_i$
8: **until** No improvement over $x$ is achieved within $k$ consecutive iterations

## 4 Computational Results for Two Real-World Applications

In this section we present two combinatorial optimization problems that arise in practice which can be modeled through the mathematical programming formulation presented in Section 2 and solved by the solution approach proposed in Section 3. The first case concerns the route design for a ship while the second case deals with the route design for a sales representative. Computational results for both applications are presented and discussed. We used the R programming language for coding the first and third phases of the solution approach. The second phase was coded in C using the CPLEX Optimization Studio Academic Research 12.2 library. The time limit for the branch-and-bound was set to $T = 1600$ seconds. All tests were performed on a Intel Core 2 Quad Q6600, 2.4 GHz with 4GB RAM.

### 4.1 Designing Ship Routes (DSR)

Once a year, a research vessel from IPMA, the Portuguese Institute of the Sea and Atmosphere, carries out a sampling survey to obtain abundance estimates of several fish species off the Portuguese coast. Sampling is made at predefined geographic locations, fixed from year to year, scattered along the Portuguese continental shelf and slope. In each location (fishing station) only one haul is carried out: the trawl is set on the bottom of the sea, towed for 30 minutes and hauled back into the ship. The duration of the haul lasts around 60 minutes,

depending on each fishing station depth. Catch sampling is carried out while the vessel moves from one fishing station to the next one.

Surveys always start and finish at the port of Lisbon and last around 28 days. Until now, at the beginning of the survey, the skipper decides whether to start sampling, the northern half of the coast or the southern one, depending on the current weather conditions, and in each case, the sequence of fishing stations to be visited. All fishing operations must be carried out during daytime (from 7 am to 6 pm). If the vessel arrives at a fishing station after 6 pm and before 7 am, it should wait. Consequently, during the night the vessel just navigates from the last fishing station of the day to the first fishing station of the next day. This routine goes on for 7 days, after which the vessel must enter a port. The vessel must arrive at one of the ports by 6 pm and leaves it the next morning, in order to arrive at the first fishing station just before 7 am. After 14 days of survey, the vessel has to anchor at the port of Lisbon where it stays for at least 24 hours for food and water supply and to change research staff. The second part of the survey is similar to the first one, covering the fishing stations that remain to be sampled. All the above time conditions define the sets of time windows that will be assigned to the fishing stations and to the ports.

As input we have a set of 95 fishing stations (mandatory cities) and 4 ports (selective cities) namely Lisboa (Lis), Portimão (Portim), Figueira da Foz (Fig) and Matosinhos (Matos). For each fishing station and each port we know the latitude, longitude and depth. Travel times do not verify the triangle inequality, since the ship has to circumvent the coastline, and might not be symmetric due to the ocean currents. We have a forecast of the time spent in fishing operation at each station, which depends on the respective depth.

Different weather conditions, met along the journey, lead to different vessel speeds: Very nice weather conditions - the vessel speed is about 20 km/h ($\simeq$ 11 knots); good weather conditions - the vessel speed is about 17 km/h ( $\simeq$ 9 knots); not so nice weather conditions - the vessel speed is about 14 km/h ( $\simeq$ 8 knots).

Moreover, our partners from IPMA asked us to evaluate the impact on the overall fishing operation time resulting from the purchase of new equipment. It can be expected that new equipment reduces the extra time spent at each fishing station by half, $\epsilon$: Set $\epsilon = 30$, for current equipment; set $\epsilon = 15$, for new equipment. By combining all the above cases, we have considered six scenarios. Following the skipper directives, problem DSR has been decomposed into two subproblems, each subproblem being a TSPSNTW. One subproblem includes the South and South-West fishing stations (43) and ports Portim and Lis (2). The other subproblem includes the North fishing stations (52) and ports Fig, Matos and Lis (3).

The information, that is available from previous years, shows that each subproblem has a feasible solution, under the weather conditions mentioned above. Thus, the main objective is to save fuel. However, everyone on board wants to return home as soon as possible. In order to mimic the weights of these two concerns we set $\lambda_1 = 2$ and $\lambda_2 = 1$ in the objective function (1).

Concerning the first step of the heuristic branch-and-bound (Section 3.2), we realized that, taking into account the LP relaxation solution at the root node and fixing, for each of the subproblems, the value of just one of the $XPE$ decision variables was enough to reach "good" quality integer solutions. In the second step, we pick the best feasible solution given by the branch-and-bound and look to daily routes where the vessel visits five or more fishing stations. For each of these daily routes, we fix at most five decision variables, randomly chosen in subset $XEE$ and rerun the branch-and-bound. Note that, setting those variables to one is not so restrictive as it looks at first sight. For example, if on day $i$ the vessel visits fishing stations 1 - 2 - 4 - 10 - 3 then we fix the decision variables $x_{1,2} = x_{2,4} = x_{4,10} = x_{10,3} = 1$ and rerun the branch and bound algorithm. Later, one can obtain solutions where a fixed daily connection becomes a night connection changing the sequence of fishing stations visited during the day. Returning to the above example this leads to a day $j$: ... - 1 - 2 - 4 and a day $j + 1$: 10 - 3 - ... in the new solution. Table 1 shows solutions obtained by applying the solution approach described in Section 3 to each of the two subproblems under the six different scenarios. The first column describes the test instance where the notation N stands for the North subproblem and SSW for South and Southwest subproblem. Following N or SSW appears a number. The first two digits of this number describe the ship speed while the third and fourth digits show the $\epsilon$ value. Each test instance takes up two rows. The first row, 'cplex', stands for the first step of the branch-and-bound procedure. The second row 'cplex(2)' concerns the second step of the branch-and-bound, after fixing a new set of decision variables. Column 'Value' shows the objective solution value. Columns '# Days visiting' and 'Max visit/day' present, respectively, the number of days that the vessel sailed and the maximum number of fishing stations visited by the ship in one day. Finally the last two columns display the total distance traveled by boat given by, respectively, the solutions of the branch-and-bound procedure and the ANS procedure. Note that, the ANS takes as input the best feasible solution given by cplex or cplex(2), depending on which row we are talking about.

Comparing cplex and cplex(2) solutions, one can see that the decrease in the objective function value of cplex(2) is mainly due to a decrease in the number of days sailing, which corresponds to a decrease in the time completion. For the 12 instances this decrease was on average 1.5 days. However, in some cases, this improvement may be followed by an increase in total traveled distance, as occurred in $N1415$, $N1715$ and $SSW1415$. Noteworthy is the $N1415$ instance where the increase in traveled distance was about 80 km, while there was a reduction of two days sailing. This highlights the contradictory nature of the two objective function components. Given these results one may infer that the second step branch-and-bound, mainly, focus on the time completion optimization. Additionally, the ANS, which is driven to reduce the traveled distance, fulfills its role by reducing the total traveled distance on average 13.5% ($-170.2km$) for cplex solutions and on average 11% ($-143.3km$) for cplex(2) solutions. For the 12 instances, the average traveled distance for the ANS procedure combined

**Table 1.** Designing a ship route: $\alpha$=5, $\beta$=0.95, $\eta$=5, $T = 1600$

| instance | B&B | | | | | ANS |
|---|---|---|---|---|---|---|
| | **Var fix** | Value | # Days visiting | Max visit/day | Dist(km) | Dist(km) |
| $N1415$ | cplex | 389.8 | 12.9 | 6 | 1128 | 1110 |
| | cplex(2) | 345.5 | 10.8 | 6 | 1208 | 1089 |
| $N1430$ | cplex | 398.5 | 12.6 | 6 | 1346 | 1112 |
| | cplex(2) | 389.4 | 12.5 | 6 | 1262 | 1141 |
| $N1715$ | cplex | 341.4 | 10.9 | 7 | 1340 | 1245 |
| | cplex(2) | 315.9 | 9.7 | 6 | 1397 | 1028 |
| $N1730$ | cplex | 384.9 | 12.6 | 6 | 1411 | 1174 |
| | cplex(2) | 340.2 | 10.2 | 6 | 1392 | 1149 |
| $N2015$ | cplex | 298.5 | 9.8 | 7 | 1273 | 1008 |
| | cplex(2) | 263.1 | 8.7 | 6 | 1089 | 957 |
| $N2030$ | cplex | 341.7 | 11.5 | 6 | 1311 | 1094 |
| | cplex(2) | 303.7 | 10 | 6 | 1285 | 1060 |
| $SSW1415$ | cplex | 321.4 | 10.4 | 7 | 998 | 893 |
| | cplex(2) | 305.5 | 9.7 | 6 | 1010 | 885 |
| $SSW1430$ | cplex | 356.1 | 11 | 5 | 1307 | 939 |
| | cplex(2) | 328.4 | 10.6 | 5 | 1052 | 928 |
| $SSW1715$ | cplex | 241.0 | 7.9 | 6 | 895 | 883 |
| | cplex(2) | 241.0 | 7.9 | 6 | 895 | 883 |
| $SSW1730$ | cplex | 315.1 | 10.8 | 6 | 1204 | 896 |
| | cplex(2) | 268.9 | 8.6 | 6 | 1065 | 986 |
| $SSW2015$ | cplex | 231.7 | 7.7 | 7 | 943 | 928 |
| | cplex(2) | 231.7 | 7.7 | 7 | 943 | 928 |
| $SSW2030$ | cplex | 288.2 | 9.7 | 6 | 1128 | 960 |
| | cplex(2) | 237.1 | 7.9 | 6 | 945 | 897 |

with cplex was 1020.2$km$ and for the ANS procedure combined with cplex(2) was 994.3$km$. If we were only concerned about traveled distance, this small difference of 25.9$km$ raises the question whether it is worth investing in the second step branch-and-bound. The CPU time for the ANS procedure was on average 0.15 seconds.

### 4.2 Designing Sales Representative Routes

A sales representative of high quality brands needs to regularly visit a set of clients scattered across the country. The salesman visits each client to advertise new releases of the brands he represents and to collect new orders. Each visit must occur between 9 am and 7 pm and has a duration that depends on the client being visited; on average it takes around 2 hours. According to their geographic

location, the clients are partitioned into two regions North and South. Clients in each region should be visited within a predefined planning horizon. Labor agreements define, for each day from Monday to Friday, a minimum rest period of 12 hours starting between 7 pm and 9 pm and a rest period of 24 hours on Saturday and on Sunday. Thus, every Friday the sales representative returns to his home city, Lisbon, where he stays during the weekend. Monday morning he continues his appointments until all clients have been visited or, again, he returns home for the weekend. Taking into account the clients's time windows and the rest periods, the salesman has to schedule the visits to clients in order to minimize the traveled distance (fuel consumption) as well as the completion time (to return home as soon as possible).

This real-life application may be viewed as a TSPSNTW where the set of clients defines the set of mandatory cities and the rest periods, including weekends, correspond to the selective cities. Clients share the same set of time windows, one for each day from Monday through Friday. The multiple time windows for clients state that each visit to a client should start between 9 am and 5 pm, to ensure that it finishes before 7 pm. Time windows for rest periods guarantee that each rest period starts between 7 pm and 9 pm. The number of rest periods, i.e., the number of selective cities that should be visited, is given by the number of nights away from the home city and it is not known in advance, since it depends on the route duration.

The current economy of the country makes the competition between sales representatives even harder than usual. Consequently, the salesman we are working with was not keen in identifying and giving us the exact location of his clients as well as he did not allow us to reveal the brands he represents. Therefore, taking into account his information we have considered 15 North clients and 10 South clients and we randomly generated their location from a predefined set of cities. Regarding the distances between clients we have considered tabled road distances. The visit duration depends on the client and was set based on the information given by the sales representative. Additionally, we set $\epsilon = 15$ minutes as a buffer time on each client visit. Regarding the rest periods, we considered that the sales representative can rest in any of the client location cities, without taking into account some personal preferences for rest locations. We set $\lambda_1 = 2$ and $\lambda_2 = 1$ in the objective function (1). Since this is a small instance problem, the dimension of the arc set is not large, we skipped the pre-processing phase.

For this application, in the first step of the heuristic branch-and-bound, we also realized that fixing the value of just one $XEP$ decision variable was a good option to reach "good" quality integer solutions. If an optimal solution is not reached, in the second step we pick the best feasible solution obtained and look to daily routes where the sales representative visits three, or more, mandatory clients. For each of these daily routes, we fix to one the corresponding decision variables in subset $XEE$ and rerun the branch-and-bound. Table 2 shows computational results for each geographical region. To assess the quality of these solutions we run cplex for two days (172800s), without fixing any decision variable, and present the corresponding solutions, $Sol0N$ and $Sol0S$. $Sol1*$ and

**Table 2.** Designing a sales representative route: $\alpha$=3, $\beta$=0.95, $\eta$=3, $T = 1600$

| instance | **Var fix** | Value | # Days visiting | Max visit/day | Dist(km) | CPU(s) | **ANS** Dist(km) |
|---|---|---|---|---|---|---|---|
| | | | **B&B** | | | | |
| *Sol*0*N* | - | 244,2 | 7.5 | 3 | 1952 | 172800 | - |
| *Sol*1*N* | cplex | 243.5 | 7.5 | 3 | 1881 | 1600 | 1845 |
| | cplex(2) | 243.4 | 7.5 | 3 | 1873 | 3.2 | 1845 |
| *Sol*2*N* | cplex | 246.1 | 7.5 | 3 | 2143 | 1600 | 1941 |
| | cplex(2) | 245.4 | 7.5 | 3 | 2073 | 7.6 | 1859 |
| *Sol*0*S* | - | 97.2 | 3.6 | 3 | 849 | 172800 | - |
| *Sol*1*S* | cplex | 104.2 | 3.8 | 3 | 819 | 1600 | 819 |
| | cplex(2) | 104.2 | 3.8 | 3 | 819 | 41 | 819 |
| *Sol*2*S* | cplex | 97.4 | 3.6 | 3 | 866 | 1600 | 866 |
| | cplex(2) | 97.4 | 3.6 | 3 | 866 | 22 | 866 |

*Sol*2$*$ correspond to two different fixing options of one $XEP$ decision variable. Comparing *Sol*0*N* with *Sol*1*N*, one may see that a better solution was given by the heuristic branch-and-bound in much less time. Furthermore, concerning the cplex(2) solutions, we noticed that the second step branch-and-bound always found an optimal solution in few seconds. For these small test instances, we saw that the ANS was not successful to reduce the traveled distance. We think that this behavior is mainly due to the "good" quality of the heuristic branch-and-bound solutions and to the rigidity of the rest periods.

## 5 Conclusions

We address a special TSP with multiple time windows in which the set of cities is partitioned into mandatory cities and selective cities. We present a mixed integer linear programming model and propose a 3-phase heuristic approach to solve it. Computational experiments address two real world applications, namely the design of ship routes, DSR, and the design of a sales representative route. Although these two applications are special cases of the referred model, they have significant differences. For the DSR the number of mandatory cities is much higher and the periodicity of visits to selective cities is much smaller. Additionally, during the night the vessel may move towards a fishing station not yet visited while in the second application the sales representative must rest.

Concerning the DSR solutions, we have received a positive feedback from our partners from IPMA which led us to believe that the heuristic branch-and-bound proved to be successful in building a feasible solution with a short completion time which was further improved, regarding the traveled distance, by the neighborhood search heuristics.

Despite the small dimension of the second application, we may say that promising solutions were obtained with the proposed methodology. However,

the rigidity of the rest periods limited the scope of the neighborhood heuristics. This suggests the study of alternative mathematical models and alternative methodologies better tailored to handle the specificities of this second real-life problem.

**Acknowledgments.** This research was partially supported by Portuguese National Funding from FCT under project PEst-OE/MAT/UI0152.

## References

1. Fischetti, M., Salazar-González, J.-J., Toth, P.: The generalized traveling salesman and orienteering problems. In: Gutin, G., Punnen, A.P. (eds.) The Traveling Salesman Problem and its Variations, pp. 609–662. Kluwer (2002)
2. Gendreau, M., Laporte, G., Semet, F.: A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks 32(4), 263–273 (1998)
3. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. Wiley, New York (1990)
4. Labadie, N., Mansini, R., Melechovsky, J., Calvo, R.W.: The team orienteering problem with time windows: An LP-based granular variable neighborhood search. European Journal of Operational Research 220, 15–27 (2012)
5. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Computers & Operations Research 34, 2403–2435 (2007)
6. Ropke, S., Pisinger, D.: An adaptative large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4), 455–472 (2006)
7. Tricoire, F., Romauch, M., Doerner, K.F., Hartl, R.F.: Heuristics for the multi-period orienteering problem with multiple time windows. Computers & Operations Research 37, 351–367 (2010)
8. Vansteenwegen, P., Souffriau, W., Berghe, G.V., Oudheusden, D.V.: Iterated local search for the team orienteering problem with time windows. Computers & Operations Research 36, 3281–3290 (2009)
9. Vansteenwegen, P., Souffriau, W., Oudheusden, D.V.: The orienteering problem: A survey. European Journal of Operational Research 209, 1–10 (2011)

# An Application of Late Acceptance Hill-Climbing to the Traveling Purchaser Problem

Andreas Goerler, Frederik Schulte, and Stefan Voß

Institute of Information Systems (IWI), University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
a.goerler@gmx.de,
{frederik.schulte,stefan.voss}@uni-hamburg.de

**Abstract.** Late Acceptance Hill Climbing (LAHC) is a recent metaheuristic in the realm of local search based procedures. The basic idea is to delay the comparison between neighborhood solutions and to compare new candidate solutions to a solution having been current several steps ago. The LAHC was first presented at the PATAT 2008 conference and successfully tested for exam timetabling, the traveling salesman problem (TSP) and the magic square problem and the results seemed extraordinary. The purpose of this paper is to analyze the behavior of the method and to provide some extended understanding about its success and limitations. To do so, we investigate the method for a generalized version of the TSP, the traveling purchaser problem.

**Keywords:** Metaheuristic, Late Acceptance Hill-Climbing, Traveling Purchaser Problem.

## 1 Introduction

The Traveling Purchaser Problem (TPP) is a well-known generalization of the Traveling Salesman Problem (TSP) [13,15,17,21] with wide applicability [10,18] and it occurs in many real-world applications related to routing, warehousing and scheduling. Starting from home a purchaser can travel to a set of markets providing different products at different prices. The aim is to fulfill a shopping list of products while minimizing the sum of traveling and purchasing costs. We apply a recent metaheuristic to the TPP, namely the Late-Acceptance Hill-Climbing (LAHC) heuristic. LAHC may be regarded as a simplification or modification of simulated annealing or threshold accepting. It was originally introduced by Burke and Bykov [5] and it won the 1st prize in the International Optimization Competition. LAHC is a one-point iterative search procedure with the general idea of delaying the comparison between neighborhood solutions. The intention behind this late acceptance strategy is to avoid the problem of getting stuck in a local optimum that many greedy search procedures have. The method was successfully tested for some problems and the purpose of this paper is to examine if an application of the LAHC to the TPP is also successfully possible.

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 173–183, 2013.

The investigation of the properties of late acceptance strategies (LAS) is still in an early stage. Consequently, only few contributions in literature deal with LAS for heuristics and metaheuristics. Abuhamdah [1] proposed LAHC and a randomized version for solving course timetabling problems. In that case the randomized version performed better than the originally presented LAHC of [5]. Özcan et al. [14] presented a set of hyper-heuristics utilising different heuristic selection methods combined with LAS for examination timetabling. Verstichel and Berghe [20] applied LAS to a lock scheduling problem. Their experiments showed that the application of the late acceptance criterion within their local search heuristic led to an improvement of the solution quality in every instance.

The remainder of this paper is organized as follows. In Section 2 we describe the TPP in more detail. In Section 3 we explain LAHC and show how to possibly apply it while using different neighborhoods. Finally, examples of computational experiments are reported.

## 2 The Traveling Purchaser Problem

Consider a set $V = \{1, 2, \ldots, m\}$ of $m$ markets, a set $K = \{1, 2, \ldots, n\}$ of $n$ products and a domicile (or home-market) $s \in V$. Let $c_{ij}$ (with $i, j \in V$) denote the cost of travel from market $i$ to market $j$. In the symmetric TPP it is assumed that $c_{ij} = c_{ji} \; \forall \, i, j$. Every product $k$ ($k \in K$) is available in a subset of markets $V_k \subseteq V_s$ (with $V_s := V - \{s\}$) and if a product $k$ is available at market $i$, $p_{ik}$ presents the cost of $k$ at $i$. Otherwise, $p_{ik}$ is set to a prohibitively large number $M$. It is implicitly assumed that if a product $k$ is available at market $i$, its available quantity $q_{ki}$ is sufficient to satisfy the demand $d_k$ [15]. This model formulation represents a symmetric uncapacitated version of the TPP. The overall idea of the TPP is to generate a tour through a subset of the $m$ markets starting and ending at $s$, such that all $n$ products are purchased while minimizing the sum of travel and purchasing costs.

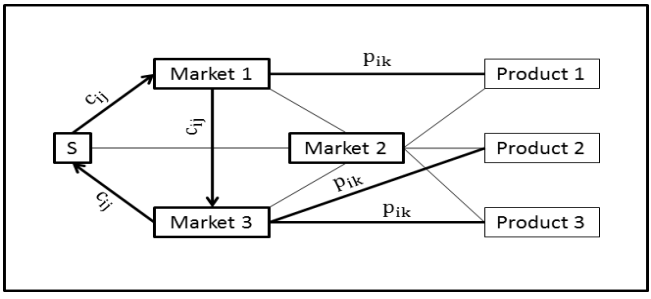



**Fig. 1.** Example of the TPP (cf. [21])

Figure 1 visualizes the TPP. In this example we have a cycle through a subset of the three markets, starting at the domicile and visiting markets 1 and 3. Product 1 is purchased at market 1 and products 2 and 3 at market 3, shown in boldface.

We assume that each product is available in at least one market and that the purchaser may pass through a market any number of times without purchasing. Or, he may purchase as many products as there are available at each market. A capacitated version of the TPP assumes that the available quantities $q_{ki}$ might be insufficient to satisfy the demand, i.e. it is assumed for each $k \in K$ that $q_{ki}$ and $d_k$ satisfy $0 < q_{ki} \leq d_k$ $(\forall i \in V_k)$ and $\sum_{j \in V_i} q_{kj} \geq d_k$ [16]. Other modifications of the TPP with additional capacity constraints, like a limit on the maximum number of markets to be visited and a limit on the number of items bought per market are also important [9].

The TPP is known to be NP-hard, since it reduces to the NP-hard TSP if each product is only available at one market and each market carries only one product. This complexity indicates that only problems of moderate size can be solved by an exact method, a suggestion that is strengthened by the scarce number of literature dealing with exact methods for the TPP. Exact algorithms proposed in the literature include a branch-and-bound algorithm calculating a lower bound by solving a relaxation similar to a simple plant location problem [18]. Branch-and-cut procedures can be found in [12,17] being able to solve instances up to 200 markets and 200 products. An approach based on constraint programming and Lagrangean relaxation is presented in [6] improving some of the earlier approaches. Despite these contributions, the literature on the TPP is mostly directed towards the development of heuristics and metaheuristics; see, e.g., [8,13,19,2]. This includes extensions of methods known from the TSP like the famous nearest neighbor method (NN) [3] as well as classical add and drop procedures [21]. First metaheuristics for the TPP were proposed in [21] showing that simulated annealing was outperformed by different dynamic tabu search strategies. Other approaches also include tabu search [4], a local search algorithm with two neighborhoods [16] and a transgenetic algorithm inspired by horizontal gene transfer and endosymbiosis [7].

## 3 Late Acceptance Strategy

LAHC is based on a simple hill-climbing algorithm. The procedure starts from a single initial solution and iteratively modifies it to produce candidate solutions. The simple hill-climbing algorithm is a greedy procedure that is regarded to be very fast but with poor results as it tends to get stuck in a local optimum very quickly. To overcome this problem, the basic idea of LAHC is to delay the comparison of solutions by "memorizing" previous current solutions in a list of a particular length, called fitness array $F_a$ of length $L_{fa}$ ($F_a = \{f_0, f_1, \ldots, f_{L_{fa}-1}\}$ with $f_a$ denoting the cost function value of solution $a$. Contrary to pure hill-climbing (note that LAHC with $L_{fa} = 1$ degenerates into pure hill-climbing) LAHC compares a candidate not to its direct previous current solution but to

the last element of the fitness array. Therefore, the LAHC accepts a decline of the objective function value by a candidate in comparison to the direct previous solution if the candidate has the same or a better objective function value than the solution at the end of the list. If accepted, the candidate solution is put on the list and the last element is removed from the list. To eliminate the shifting of the whole list the authors propose the use of a "virtual" shifting of the fitness array. Therefore, they compute the virtual beginning $v$ at the $I^{th}$ iteration by $v = I \bmod L_{fa}$, i.e., $v$ is equal to the division of the current number of iterations $I$ to the length of the list $L_{fa}$. Now the candidate solution is compared with the cost function $f_v$ and if accepted the value of the candidate solution is assigned to $f_v$.

### 3.1 Initial Solutions

As described above the LAHC starts from a single initial solution. In case of the TPP this solution is represented by the cost function value of an initial tour through the considered graph. An initial tour for the TPP is determined by applying the NN to all markets without considering any purchases. The NN method is described as follows: Starting at the depot $s$ the purchaser goes to the market with the lowest travel costs from the depot. From there he visits the nearest market that was not visited before. The procedure is repeated until there are no unvisited markets left and the purchaser returns to the depot. The obtained TSP tour can be transformed into a TPP tour by buying each product at the market where it is available at the lowest possible price. An alternative approach is described by Burke and Bykov [5]. They build an initial feasible path for a TSP instance and randomly exchange markets to ensure different initial solutions. Starting with such an inital tour we apply a simple drop procedure [21]. In each step it drops a market giving the largest reduction to the objective function value. If no more reduction is possible or infeasibility occurs, the procedure is terminated and the obtained solution serves as the initial solution ($f(\sigma)$) for the LAHC algorithm.

### 3.2 Neighborhood Definitions

Several neighborhoods may be applied. The first approach is the IMP1-procedure from [21]. Starting with a feasible solution in each iteration exactly one market is excluded from the tour (called drop step) whose removal gives the best improvement or least deterioration of the cost function value $f$ (note that strategic oscillation into infeasibility is permitted). Afterwards a number of consecutive insertions (add steps) are performed as long as an improvement in the cost function value is achieved. IMP1 is terminated after a complete iteration, i.e., when a market that has been previously removed was added again. This set of moves defines a neighborhood of candidate solutions.

The l-ConsecutiveExchange procedure from [17] represents a generalization of IMP1. Given a feasible solution $\sigma$, a starting value $l$ is chosen ([17] obtained best results for $2 \leq l \leq 25$). Afterwards the procedure has two steps,

namely $l$-ConsecutiveDrop and a step for restoring feasibility if needed. For $l$-ConsecutiveDrop $l$ markets are selected according to an estimation of the trade-off between the travel cost reduction occuring by dropping the $l$ markets and the increase in purchasing costs due to the fact that the products that were purchased in the dropped $l$ markets have to be bought elsewhere. These trade-off evaluations are performed for each possible path consisting of $l+1$ consecutive edges belonging to the initial tour. The different paths are ranked according to the travel cost reduction minus the purchase costs increase and the path with the highest rank is dropped. After dropping the $l$ markets NN is applied to reduce the travel costs of the new (possibly infeasible) tour $\sigma$.

If feasibility has to be restored, new markets are added. To guarantee the generation of a diverse neighborhood only markets are permitted to be added that have not been in the initial feasible tour. The restore feasibility step proceeds with computing subsets of permitted markets ($V_p \subseteq V$). For the capacitated TPP the non-satisfied amount of each product $k$ in the infeasible tour has to be calculated and only subsets of markets are permitted that are selling the required quantity of product $k$ needed to restore feasibility. For each market $i \in V_p$ the travel cost increase ($\rho(i, \sigma)$), describing the increase in travel costs if market $i$ is added to $\sigma$, and the decrease in purchasing costs ($\mu(i, \sigma)$) for adding $i$ is computed as product $k$ may be available at a cheaper price at market $i$. Markets in the selected subset $V_p$ are added one after another according to their trade-off between travel cost increase and purchase cost decrease. Finally, we apply LAHC to possibly re-optimize the obtained tour.

### 3.3 LAHC Algorithms

The general approach of the LAHC algorithms starts with the calculation of an initial solution. Afterwards the length of the fitness array $L_{fa}$ and an initial $l$ are specified. An inner loop performs an iterative procedure to remove $l$-consecutive markets from the initial tour $\sigma$. If the removal of the $l$ markets leads to an infeasible tour the following add procedure grants that the feasibility is restored. This procedure randomly adds markets from a list of earlier dropped markets at random position within the tour of the candidate solution. To discourage staying at local optima the procedure does not allow the add of markets dropped in the current iteration. Markets are added until the solution of the candidate tour is feasible and its penalty can no longer be reduced. Finally, the method LAHCTSP tries to improve the order of markets in the candidate tour. After building the neighborhoods LAHC is called. If the cost function value of the candidate $f(\sigma')$ is better or equal to the value of the solution $f_v$ with $v$ being the solution at the beginning of the virtual list the candidate is accepted and replaces $f_v$; otherwise it is rejected. DeltaPenalty follows the idea of an incremental evaluation to reduce CPU time, where only changes in the cost function values are calculated and compared to changes in cost function value of neighborhood solutions rather than calculating the complete objective function value in every iteration. After accepting or rejecting a candidate solution the iteration number $I$ is set to $I + 1$.

```
Input: A TPP instance
σ := InitialTSP (V)
σ := InitialDrop (σ)
Calculate initial cost function f(σ)
Specify L_fa
For all k ∈ {0...L_fa−1} f_k := f(σ)
Specify l
First iteration I = 0
σ' := σ
repeat
   while l ≥ 1 do
      σ'' := σ'
      σ' := l - ConsecutiveDrop (σ, l)
      if f(σ'') ≥ f(σ') or σ'' not feasible then
         l := l − 1
      else
         σ' := σ''
      end if
   end while
   repeat
      σ'' := σ'
      σ'' : AddRandomMarket (σ'')
   until σ'' ≥ σ' AND σ'' is feasible
   σ' := LAHCTSP (σ'')
   f(σ') = f(σ) + DeltaPenalty (σ)
   v := I mod L_fa
      if f(σ') ≤ f(σ) or f(σ') ≤ f_v then
         σ := σ'
         f_v := f(σ)
         I := I + 1
until stopping condition
return σ
```

**Fig. 2.** Pseudocode of the LAHC Algorithm

We developed two different types of LAHC algorithms to solve the TPP. The first algorithm (LAHC, see Figure 2) strictly applies the list approach of LAHC for the evaluation of candidate solution and the LAHCTSP method. The candidate solution is created by a sequence of a drop step, an add step and a TSP heuristic and accepted if the LAHC constraints are fulfilled. The second algorithm (sLAHC) simplifies this idea and accepts changes after an evaluation in the drop step, add step and the TSP heuristic.

## 4 Computational Results

The algorithms described in Section 3.3 are implemented in Java and run on a Intel Core 2.50GHz with 8 GB RAM. We tested the performance of the two

algorithms on the instances of Class 1 (http://webpages.ull.es/users/jriera/TPP.htm) defined by Singh and van Oudheusden [18]. This class contains 33-market symmetric instances and the product prices are generated according to a uniform distribution in an interval of [1,500]. The routing costs are taken of the 33-vertex TSP described in Karg and Thompson [11] and the first vertex corresponds to the depot. All markets sell all products and the quantity of products varies between 50 and 500. The algorithms runs were aborted after a number of 15000 iterations for every instance.

25 instances with 50 to 250 products have been solved to optimality (note that the optimal values for these instances are also provided at the webpage mentioned above). Table 1 compares the results of the sLAHC and the LAHC algorithm to these optimal values. Optimal values for the instances with 300 to 500 products have not yet been presented. In Table 2 we provide the objective function values for 25 instances out of this range with a number of products from 300 up to 500. The results show, that both algorithms achieve optimal results or small optimality gaps for almost all tested instances and outperform known results from literature (see Table 3). Especially for small instances the sLAHC algorithm seems to perform better than the LAHC algorithm. The LAHC tends to get stuck in a local optimum quickly for those instances. The sLAHC, in contrast, has a lower risk to stay early at a local optimum but seems to be weaker for larger instances. For small instances the LAHC fitness array seems to guide the LAHC algorithm to local optima while the sLAHC can come closer to the global optimum without the guidance of a LAHC fitness array. For large instances this disadvantage seems to turn into an advantage and the LAHC seems benefit from the LAHC mechanism. The sLAHC achieves optimal solutions for all five instances with 50 products and performs more homogenously over all instances without any spikes. Note that the results in Table 1 were created with a single batch run. For small instances the LAHC results highly depend on the initial feasible solution, which were generated randomly. Thus exceptionally large optimality gaps might occur. The average gaps are usually much lower like, e.g., for instance 3 another run led to a 0% instead of a 14% optimality gap.

The column headings in the tables are described as follows:

| | | |
|---|---|---|
| sLAHC | : | simplified Late Acceptance Hill Climbing; |
| $LAHC_{5000}$ | : | Late Acceptance Hill Climbing with $L_{fa} = 5000$; |
| $LAHC_{10000}$ | : | Late Acceptance Hill Climbing with $L_{fa} = 10000$; |
| $V$ | : | number of markets; |
| $K$ | : | number of products; |
| $V^*$ | : | number of markets involved in the best solution; |
| $V^*_{LA}$ | : | number of markets involved the best solution of the LAHC; |
| $\%gap$ | : | quality of the LAHC over an optimal solution; |

Furthermore, we compare different list lengths of the LAHC fitness array. No variation of the list length is presented for the sLAHC algorithm as it could be seen throughout the experiments that a variation of the list length has no

**Table 1.** Numerical results Class 1 (50 - 250 products)

| | | | sLAHC | | $LAHC_{5000}$ | | $LAHC_{10000}$ | |
|---|---|---|---|---|---|---|---|---|
| $V$ | $K$ | $V^*$ | $\%gap$ | $V^*_{LA}$ | $\%gap$ | $V^*_{LA}$ | $\%gap$ | $V^*_{LA}$ |
| 33 | 50 | 9 | 0 | 9 | 1,021 | 9 | 1,986 | 8 |
| 33 | 50 | 8 | 0 | 8 | 1,019 | 8 | 3,226 | 9 |
| 33 | 50 | 8 | 0 | 8 | 14,108 | 8 | 14,174 | 8 |
| 33 | 50 | 8 | 0 | 8 | 12,726 | 7 | 1,593 | 8 |
| 33 | 50 | 9 | 0 | 9 | 8,660 | 8 | 9,695 | 9 |
| 33 | 100 | 11 | 0,237 | 11 | 1,039 | 10 | 0,289 | 9 |
| 33 | 100 | 10 | 0,703 | 10 | 4,104 | 11 | 8,940 | 10 |
| 33 | 100 | 11 | 0,971 | 10 | 3,434 | 10 | 3,800 | 10 |
| 33 | 100 | 11 | 0,94 | 11 | 5,886 | 10 | 2,736 | 10 |
| 33 | 100 | 9 | 1,232 | 11 | 1,613 | 10 | 3,429 | 10 |
| 33 | 150 | 12 | 0,95 | 11 | 3,316 | 12 | 2,506 | 12 |
| 33 | 150 | 14 | 1,928 | 12 | 2,079 | 13 | 3,000 | 12 |
| 33 | 150 | 13 | 0,771 | 11 | 2,785 | 12 | 0,268 | 12 |
| 33 | 150 | 13 | 1,659 | 12 | 4,632 | 13 | 2,575 | 12 |
| 33 | 150 | 14 | 1,864 | 13 | 3,845 | 12 | 3,706 | 12 |
| 33 | 200 | 12 | 2,795 | 13 | 3,783 | 13 | 1,863 | 13 |
| 33 | 200 | 14 | 2,227 | 12 | 4,410 | 14 | 4,093 | 12 |
| 33 | 200 | 14 | 0,778 | 15 | 3,689 | 14 | 4,943 | 14 |
| 33 | 200 | 15 | 2,792 | 11 | 4,352 | 14 | 3,603 | 14 |
| 33 | 200 | 14 | 1,177 | 12 | 7,648 | 13 | 3,199 | 13 |
| 33 | 250 | 15 | 1,915 | 14 | 3,238 | 15 | 1,954 | 15 |
| 33 | 250 | 15 | 0,958 | 16 | 2,300 | 15 | 3,874 | 16 |
| 33 | 250 | 15 | 4,846 | 15 | 5,872 | 16 | 4,669 | 16 |
| 33 | 250 | 16 | 2,963 | 15 | 3,171 | 15 | 2,818 | 17 |
| 33 | 250 | 15 | 2,647 | 15 | 0,641 | 15 | 2,462 | 16 |

significant impact due to the fact that the late acceptance strategy is only used for calculating the TSP in the sLAHC as described in Section 3.3. It can be seen that the calculation with a list length $L_{fa} = 10000$ compares favorably to the case with a list length of $L_{fa} = 5000$. The LAHC achieves better objective function values for most instances with a longer list length.

In Table 3 we compare our results to results from literature, i.e., with the (truncated) branch-and-cut approach of Laporte et al. [12]. As they built an average over five random instances of Class 1 we also calculated the average results for our algorithms to allow a comparison. Table 3 shows that the sLAHC algorithm performs significantly better on average than the average results of [12]. The LAHC implementation performs also better than the approach of [12] for the instances with more than 50 products.

Figure 3 shows an iterations-to-target chart for the LAHC and the sLAHC. It can be seen that the LAHC has a higher average starting value than the sLAHC. For the sLAHC a rapid drop in the average objective function value can be observed in the first hundred iterations and afterwards the sLAHC curve

**Table 2.** Numerical results Class 1 for unknown instances (300-500 products)

| | sLAHC | | $LAHC_{5000}$ | | $LAHC_{10000}$ | |
|---|---|---|---|---|---|---|
| $V$ $K$ | $f(\sigma)$ | $V^*_{LA}$ | $f(\sigma)$ | $V^*_{LA}$ | $f(\sigma)$ | $V^*_{LA}$ |
| 33 300 | 14576 | 17 | 14637 | 15 | 14346 | 16 |
| 33 300 | 14712 | 16 | 14733 | 14 | 14710 | 14 |
| 33 300 | 14463 | 16 | 14695 | 16 | 14354 | 15 |
| 33 300 | 14347 | 16 | 14189 | 19 | 14410 | 15 |
| 33 300 | 14530 | 15 | 14471 | 17 | 14329 | 16 |
| 33 350 | 15901 | 17 | 15990 | 19 | 16006 | 18 |
| 33 350 | 15047 | 13 | 14665 | 17 | 15035 | 16 |
| 33 350 | 16383 | 15 | 16464 | 16 | 16108 | 17 |
| 33 350 | 16252 | 17 | 16400 | 16 | 16023 | 17 |
| 33 350 | 16118 | 16 | 16398 | 17 | 16249 | 18 |
| 33 400 | 17790 | 18 | 17805 | 17 | 17905 | 19 |
| 33 400 | 16906 | 18 | 16836 | 17 | 16832 | 16 |
| 33 400 | 17335 | 18 | 17547 | 19 | 17480 | 18 |
| 33 400 | 17313 | 19 | 17824 | 18 | 17703 | 19 |
| 33 400 | 17825 | 17 | 18029 | 17 | 17832 | 19 |
| 33 450 | 18140 | 18 | 18360 | 18 | 18309 | 19 |
| 33 450 | 18164 | 18 | 18312 | 20 | 18097 | 18 |
| 33 450 | 18555 | 17 | 18677 | 19 | 18737 | 18 |
| 33 450 | 17819 | 17 | 17952 | 20 | 17875 | 18 |
| 33 450 | 19021 | 18 | 19081 | 18 | 18795 | 18 |
| 33 500 | 19897 | 19 | 19821 | 21 | 19827 | 19 |
| 33 500 | 19888 | 18 | 19604 | 19 | 20028 | 19 |
| 33 500 | 19774 | 20 | 19967 | 19 | 19885 | 19 |
| 33 500 | 19863 | 18 | 19925 | 19 | 19883 | 19 |
| 33 500 | 19497 | 20 | 19818 | 19 | 19643 | 20 |

**Table 3.** Comparison of results

| | sLAHC | | $LAHC_{5000}$ | | $LAHC_{10000}$ | | Laporte et al. [12] |
|---|---|---|---|---|---|---|---|
| $V$ $K$ | $\%gap$ | $V^*_{LA}$ | $\%gap$ | $V^*_{LA}$ | $\%gap$ | $V^*_{LA}$ | $\%gap$ |
| 33 50 | 0,000 | 8,4 | 7,507 | 8 | 6,135 | 8,4 | 6,312 |
| 33 100 | 0,816 | 10,4 | 3,215 | 10,2 | 3,839 | 9,8 | 4,150 |
| 33 150 | 1,434 | 11,8 | 3,331 | 12,4 | 2,411 | 12 | 4,760 |
| 33 200 | 1,954 | 12,6 | 4,776 | 13,6 | 3,540 | 13,2 | 8,567 |
| 33 250 | 2,666 | 15 | 3,044 | 15,2 | 3,155 | 16 | 5,478 |

is flatter than the LAHC. For both algorithms it is apparent, that the most improvement in the average objective function value occurs in the first thousand iterations. After 10000 iterations almost no more significant improvement takes place. The figure also illustrates that the sLAHC has a lower average objective function value than the LAHC.

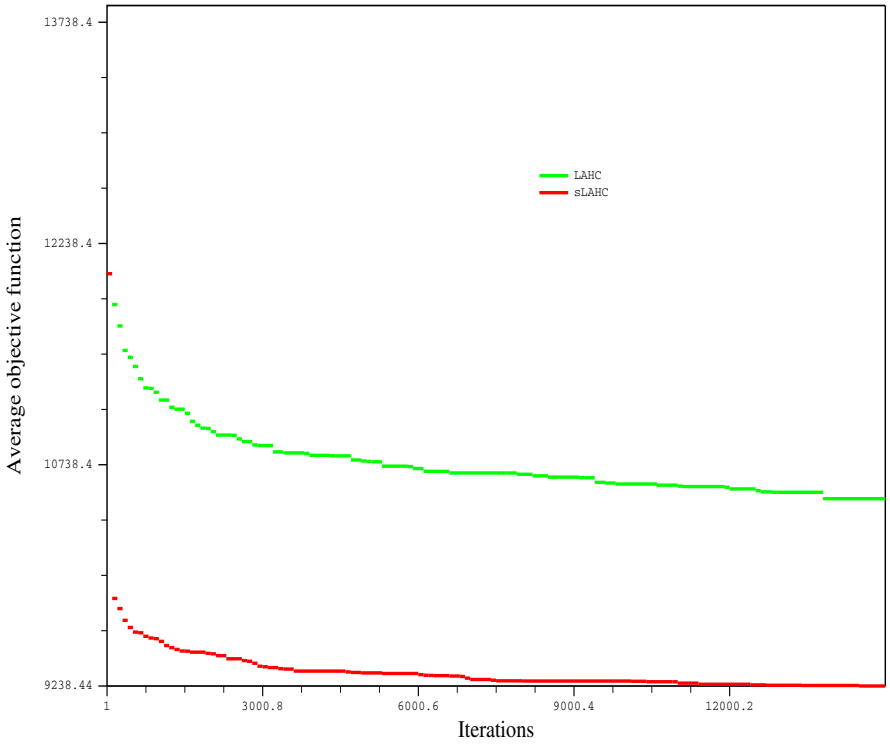



**Fig. 3.** Iterations-to-target evaluation (LAHC/sLAHC). LAHC and sLAHC have been applied to 25 instances each with 15000 iterations in all cases.

## 5 Conclusion

We have applied Late Acceptance Hill Climbing to the Traveling Purchaser Problem. We combined several ideas from literature for building initial solutions and for defining neighborhoods. We tested our approach on academic benchmarks and the results indicate that the LAHC procedure is suitable for solving the TPP. Furthermore, we also presented a simplified application of LAHC which led to promising results, especially for instances with a smaller number of products. The relatively weak performance of LAHC for small instances seems to indicate a small drawback of LAHC in comparison to other metaheuristics. While LAHC requires less effort for parameter setting, it does not allow a fine tuning of acceptance probabilities like, e.g., simulated annealing allows. On the other hand, LAHC might work well with an auto-adaptive parameter tuning that adjusts the neighborhood construction methods and the LAHC fitness array length. This could be a promising future research approach.

## References


1. Abuhamdah, A.: Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. International Journal of Computer Science and Network Security 10(1), 192–200 (2010)
2. Angelelli, E., Mansini, R., Vindigni, M.: Look-ahead heuristics for the dynamic traveling purchaser problem. Computers & Operations Research 38, 1867–1876 (2011)
3. Bellmore, M., Nemhauser, G.L.: The traveling salesman problem: A survey. Operations Research 16, 538–558 (1968)
4. Boctor, F., Laporte, G., Renaud, J.: Heuristics for the traveling purchaser problem. Computers & Operations Research 30, 491–504 (2003)
5. Burke, E.K., Bykov, Y.: A late acceptance strategy in hill-climbing for exam timetabling problems. In: Proceedings of the 7th Int. Conf. on the Practice and Theory of Automated Timetabling, PATAT 2008 (2008)
6. Cambazard, H., Penz, B.: A constraint programming approach for the traveling purchaser problem. In: Milano, M. (ed.) CP 2012. LNCS, vol. 7514, pp. 735–749. Springer, Heidelberg (2012)
7. Goldbarg, M.C., Bagi, L.B., Goldbarg, E.F.G.: Transgenetic algorithm for the traveling purchaser problem. European Journal of Operational Research 199, 36–45 (2009)
8. Golden, B., Levy, L., Dahl, R.: Two generalizations of the traveling salesman problem. Omega 9, 439–441 (1981)
9. Gouveia, L., Paias, A., Voß, S.: Models for a traveling purchaser problem with additional side-constraints. Computers & Operations Research 38, 550–558 (2011)
10. Infante, D., Paletta, G., Vocaturo, F.: A ship-truck intermodal transportation problem. Maritime Economics & Logistics 11, 247–259 (2009)
11. Karg, R.L., Thompson, G.L.: A heuristic approach to solving travelling salesman problems. Management Science 10, 225–248 (1964)
12. Laporte, G., Riera-Ledesma, J., Salazar-González, J.: A branch-and-cut algorithm for the undirected traveling purchaser problem. Operations Research 51, 940–951 (2003)
13. Ong, H.L.: Approximate algorithms for the travelling purchaser problem. Operations Research Letters 1, 201–205 (1982)
14. Özcan, E., Bykov, Y., Birben, M., Burke, E.K.: Examination timetabling using late acceptance hyper-heuristics. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 997–1004 (2009)
15. Ramesh, T.: Traveling purchaser problem. Opsearch 18, 78–91 (1981)
16. Riera-Ledesma, J., Salazar-González, J.: A heuristic approach for the travelling purchaser problem. European Journal of Operational Research 162, 142–152 (2005)
17. Riera-Ledesma, J., Salazar-González, J.: Solving the asymmetric traveling purchaser problem. Annals of Operations Research 144, 83–97 (2006)
18. Singh, K.N., van Oudheusden, D.L.: A branch and bound algorithm for the traveling purchaser problem. European Journal of Operational Research 97, 571–579 (1997)
19. Teeninga, A., Volgenant, A.: Improved heuristics for the traveling purchaser problem. Computers & Operations Research 31, 139–150 (2004)
20. Verstichel, J., Berghe, G.: A late acceptance algorithm for the lock scheduling problem. In: Voß, S., Pahl, J., Schwarze, S. (eds.) Logistik Management, pp. 457–478. Physica, Heidelberg (2009)
21. Voß, S.: Dynamic tabu search strategies for the traveling purchaser problem. Annals of Operations Research 63, 253–275 (1996)

# Airport Gate Assignment Considering Ground Movement

Urszula M. Neuman and Jason A.D. Atkin

University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom
{psxun,jason.atkin}@nottingham.ac.uk

**Abstract.** Airports all over the world are becoming busier and many of them are facing capacity problems. The actual airport capacity strongly depends on the efficiency of the resource utilisation. Although simultaneously handling all of the problems may result in more effective resource utilisation, historically different types of airport resources have been handled independently. Despite introducing new support systems the historical separation has often remained. This may increase congestion, which has a negative impact on both the passengers' comfort and the environment. This paper focuses on modelling the gate allocation problem taking into consideration possible conflicts at taxiways around gates. Introducing the taxiway information in the early stage of the allocation planning is a step forward in integration of the two airport operations. Various configurations of the model have been tested using a real data set to evaluate how the new anti-conflict and novel towing constraints influence the final allocation.

**Keywords:** Airport Gate Assignment, Mathematical Modelling, Mixed Integer Programming.

## 1 Introduction

Air traffic is increasing all over the world. Airports are becoming busier and many of them are facing capacity problems. Effective resource management, therefore, has a key importance for airports.

Airport resources can be divided into an air-side part, which consists of runways, taxiways and gate areas, and a land-side part which includes terminal buildings. In practice different parts of the airport are often managed and maintained in isolation from each other, often due to the fact that they had to be solved manually in the past, and by different organisations. So, conflicting objectives are common. An airport may, for example, prefer to allocate a flight to a particular gate because of a lower traffic in its neighbourhood while an airline prefers to use a different gate which is closer to the runways. If the whole airport was managed as one inseparable operation, its operations would be smoother, but this may require major changes in the airport management structure. This is starting to happen (especially as a part of the Airport Collaborative Decision

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 184–198, 2013.

Making [1] project), and this paper considers moving another step forward in the integration of airport operations.

The presented paper discusses a model for the gate allocation problem taking into consideration the expected traffic around the gates in order to improve the early stage of gate allocation planning. Gate allocation aims to find appropriate gates for aircraft at an airport. There are many rules and constraints which are used to decide the value of an allocation. In contrast to previously published models, the one presented here contains a new constraint which limits the number of aircraft which are expected to block each other while manoeuvring at the area close to the gates. The possible blockages are identified in groups of gates which are reached using similar routes. Including the expected blockages in the early stage of planning should result in smoother operation during the day. We believe this is a good first step in the process of integrating gate allocation with ground movement, never discussed before. Additionally a novel approach to the towing procedure is also presented. This is modelled in a flexible way which may be more practical than some other published models.

Other blockages which occur on taxiways could be modelled similarly and handled by adding appropriate constraints. This moves the system closer towards a ground movement system which is planned future work.

The problem description and the background literature are introduced in Section 2 and Section 3. The model formulation is given in Section 4. Experimental settings and the data used in the experiments are described in Section 5. Section 6 presents the results, where we discuss how the various model constraints influence the final allocation. Finally conclusions and potential future work are presented in Section 7.

## 2 Problem Description

There are many operations which need to be synchronized at airports. For example, departing aircraft are routed from gates to runways and the routes that they take and their take off sequence both have to be considered. Similarly arriving aircraft also need to be sequenced and, once on the ground, they must be routed from runways to gates. There are usually several taxiways which could be used to get from one point to another. The chosen taxiway should normally be the shortest possible route without conflicts. Terminal 1 of Manchester Airport is shown in Figure 1 as an example. Observe that some gates are placed behind bottlenecks, for example the gates at $GR1$. Access to those gates is limited and a poor quality allocation plan can result in severe delays during the day of operation. However, a good plan created in advance which considers the physical limitations could potentially reduce delays, by limiting the number of aircraft planned to pass the bottlenecks at the same time.

Examples of physical blockages which can be taken into account in the early stage of planning are push-back blocking (Figure 2a) and taxi blocking (Figure 2b). Push-back blocking occurs when a departing aircraft is ready to leave a gate but cannot due to another aircraft is blocking its push-back trajectory.

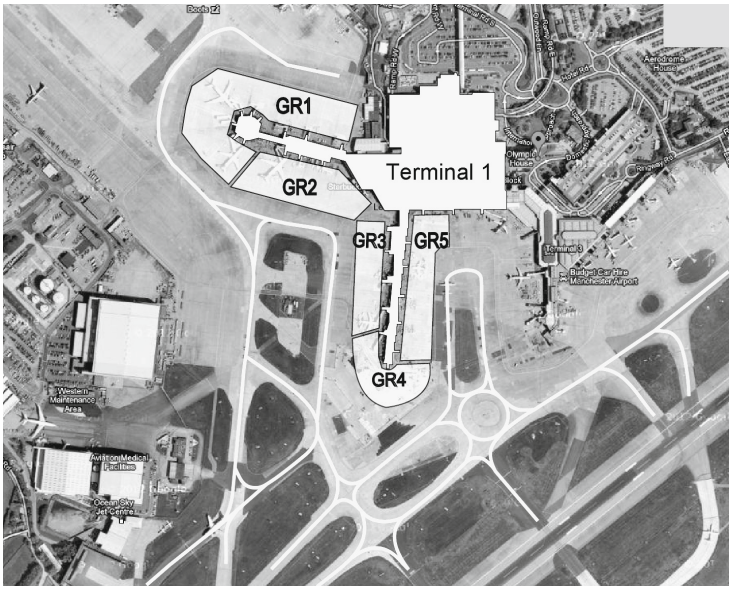
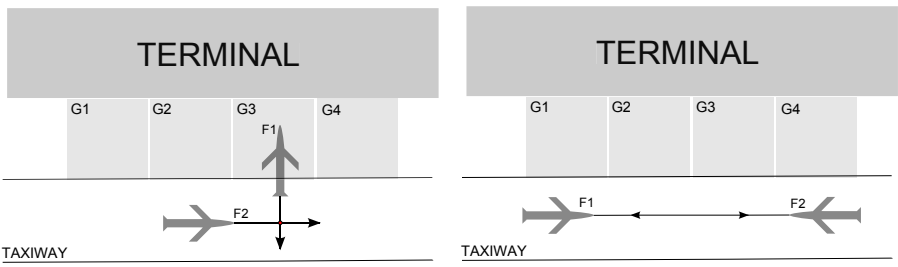



**Fig. 1.** Map of Manchester Airport showing Terminal 1, the taxiways and groups of gates, Google satellite map, Imagery ©2013 TerraMetrics, Map data ©2013 Google

This type of blocking can be observed only at the area close to the gates. Figure 2a shows an example of push back blocking where flight $F1$ is pushing back towards flight $F2$. One of the aircraft needs to wait until the other has finished.




(a) push-back blocking (b) taxi blocking

**Fig. 2.** Two different types of conflicts

Taxi blocking occurs either when there are two aircraft going in opposite directions along a straight piece of a taxiway, as shown in Figure 2b, or when two taxiways cross and two aircraft happen to be at the crossing point at the same time. Taxi blocking may occur not only at the gates but also further along the taxiways, for aircraft on the way to/from the runway. It occurs especially around bottlenecks where multiple taxiways converge. Only the area at the gates

is considered in our experiments but we plan to include the bottlenecks in the future and the model can handle this by including some time offsets to allow for the time reaching the bottlenecks.

The model of the gate allocation problem (GAP) which is presented here is meant to be an early stage planning tool which attempts to resolve conflicts at gates. Gates are divided into groups and the number of conflicting allocations is limited within each of the groups. The groups which are visible in Figure 1 have been introduced based upon the layout of the taxiways and our understanding of the problem.

Gate constraints, which result from gate attributes, are also modelled. A gate is an area adjacent to a terminal building where an aircraft can park so that passengers can easily be loaded and unloaded. While an aircraft is on a gate all necessary work such as fuelling, cleaning, baggage loading and offloading are also performed. The work and time that an aircraft needs on a gate depends primarily upon the airline that is operating it and the flight destination. There are also areas which are not attached to terminals where aircraft can park, called remote stands. When an aircraft is on a remote stand passengers are usually transported to and from a terminal building by bus. The remote stands are usually used only when all gates are occupied or if specifically requested by an airline. A number of constraints result from the gate attributes, such as size restrictions and/or shadowing restrictions (see Section 4.2). Some aircraft will need to be allocated to different stands for their arrival and departure processes, for instance due to different security requirements for the arriving and departing flights. These may require a tug operation in between them to move the aircraft between stands. Such aircraft can be modelled as two flights within this model, although the experimental data used for these results does not detail such aircraft and these are not the main focus of this work.

The GAP for one day (whole airport) can be very hard to solve using exact methods. However, the whole airport problem can usually be decomposed by terminals, since it is rare for an aircraft to be moved between terminals (all passengers would have to change which terminal they checked in at) and airlines usually have agreements with airports specifying which terminal (and sets of gates) they will use. Allocating flights to a different terminal would be very inconvenient for the airline because all of the necessary ground services would have to be moved to a different part of the airport. However, synchronisation of terminals could be needed later (when bottlenecks are included) since some of taxiways are used to reach multiple terminals.

The proposed terminal based decomposition allowed the solution of instances used in the experiments. But the solution process is slow due to the number of possibilities that have to be checked. Similar problems have been reported by other researchers working on the GAP [2,10]. It would be very hard to solve larger instances for busy hub terminals to optimality, hence future work will consider potential decompositions of the problem and/or heuristic approximations to generate additional constraints and prune the search space.

Like many real world problems the GAP has several objectives. The presented model includes four objectives, which mirror the interests of the two sites involved in this problem: an airport and an airline. From the perspective of an airport one of the most important objectives is to obtain a robust allocation, i.e. an allocation with gaps between flights which are allocated to the same gate are as large as possible. It allows the absorption of minor delays and maintains smoother airport operation. Another airport related objective is to use the gate space effectively, i.e. to match the size of the aircraft. The aim is to avoid using unnecessarily large gates, keeping the larger gates free since these have more flexibility for use by flights which have to be moved on the day of operation. Airline related objectives involve specific gate preferences and avoiding remote allocation where possible. The multi-objective character has been modelled here using a weighted sum of several objectives. This is a commonly used procedure which results in the optimal solution potentially being strongly dependent on the chosen weights. This is sufficient for the aims of this paper. However, different multi-objective approaches [3], which allow the analysis of the trade-off and/or tuning of the weights for the various objectives will be investigated in future.

## 3 Previous Literature

The multi-objective nature of the problem has been gradually acknowledged by researchers working on this problem. Initially research focused mainly on various passenger comfort objectives. For example, Haghani and Chen [10] and Xu [15] minimised the distance a passenger walks inside a terminal to reach a departure gate. Yan and Hou [16] introduced an integer programming model to minimise the total passenger walking distance and the total passenger waiting time.

Ding et al. [6,7] shifted the research interest slightly from passenger comfort to airport operations and solved a multi-objective IP formulation of the GAP with an additional objective: minimization of the number of ungated flights.

Dorndorf et al. [8] went further and claimed that not enough attention has been given to the airport side when solving the GAP. They focused on three objectives: maximization of the total flight to gate preferences, minimization of the number of towing moves and minimisation of the absolute deviation from the original schedule. Perhaps surprisingly at the time, in the context of many previous publications, they omitted the walking distance objective, arguing that the airport managers do not consider it an important aspect of the GAP. Our experience with airports indicates that this is probably correct, which is why the objective function presented in Section 4 does not consider passenger walking distance but aims instead to reduce the conflicts by the gates, to allocate gates so that the airline and the size preferences are maximised and to ensure that the time gaps between two adjacent allocations are large enough.

The constraints that we have modelled in Section 4 include other aspects of the problem which have already been discussed by other researchers. Dorndorf et al. discussed relative sizes of gates and aircraft, airline preferences and shadowing restrictions [8,9]. Similarly the importance of the maximisation of time gaps

between allocations and the robustness of solutions was discussed by Bolat [2] and Diepen et al. [5]. This is also included in our objective function. We aim to avoid small gaps by maximizing the time gaps within a defined time window.

Dorndorf et al. [8,9] and Kumar et al. [12] included towing in their models. They modelled the towing using three flights: (arrival)+(tow away); (tow away)+(tow back); and (tow back)+(departure). In this paper we assume that there are always enough remote stands for towed aircraft and model it using two flights: arrival+tow away and tow back+departure. This reduces the number of variables used by the model and eliminates the associated constraints. In both the current and the previous models, the decision about an exact towing time has been left in controllers' hands: they can pick the most appropriate time within an established time period.

Cheng [4] analysed the push-back conflicts that can be observed between adjacent gates. He analysed flights which can be in conflict due to scheduled departing/arriving times and did not allow these to be allocated to neighbouring gates. His idea of detecting conflicts and resolving them by introducing hard constraints into his model is similar to the idea used in our model. Our extension allows a gate to be a member of one of the arbitrarily defined groups of gates which are considered, rather than only considering neighbouring gates. Kim et al. [11] introduced probable conflicts into the model as a soft constraint and minimised them. That approach introduces many new variables into the model, which makes it harder to solve. Moreover, it is dedicated to a specific architecture of an airport, where gates are grouped on ramps and two non-crossing taxiways which lead to them are present. However, at many airports the taxiways cross and there are often bottlenecks which make access to gates harder.

## 4 Model Formulation

A mixed integer programming model has been used here to model the problem.

### 4.1 Notation and Definitions

- $F$ - the set of all flights
- $n$ - the total number of flights
- $f \in F := \{1, ..., n\}$ - a flight
- $e_f$ - the on-gate time of flight $f$, a constant for each flight $f$
- $l_f$ - the off-gate time of flight $f$, a constant for each flight $f$
- $G$ - the set of all gates
- $m$ - the total number of gates available
- $g \in G := \{1, ..., m\}$ - a gate
- $F(g)$ - the subset of flights that can use gate $g$
- $Sh$ - the set of pairs of gates that cannot be used simultaneously due to shadow restriction
- $GR \in \{GR1, ...., GR5\}$ - a subset of gates (group of gates)
- $M$ - the number of conflicting flights allowed to be allocated to one GR

- $X_{g,f}$ - a decision variable that takes the value 1 if flight $f$ is allocated to gate $g$ or 0 if not
- $U_{g,f_1,f_2}$ - an indicator variable that is 1 if the two flights $f_1$ and $f_2$ are both allocated to gate $g$
- $p_{f_1,f_2}$- the constant penalty function for each flight pair which indicates the penalty to apply if flights $f_1$ and $f_2$ are both allocated to the same gate
- $r_{f,g}$ - the constant penalty for each gate-flight pair which indicates the penalty to apply if flight $f$ is smaller than the maximal size of flight that can be allocated to gate $g$
- $a_{f,g}$ - the constant penalty for each gate flight pair which indicates how strongly gate $g$ is preferred by an airline that is operating flight $f$
- $freq_{f,g}$ - the constant which indicates how often an airline which is operating flight $f$ has used gate $g$, based upon statistic data analysis
- $d$ - the constant penalty for usage of 'dummy gate'
- $C_d(f)$,$C_a(f)$ - a set of flights which can be in conflict with the departure time ($C_d(f)$) or arrival time ($C_a(f)$) of flight $f$ if the flights are allocated to the same group of gates.

### 4.2 Constraints

$$X_{g,f} = 0,\ \forall f \notin F(g), \forall g \in G. \tag{1}$$

$$U_{g,f_1,f_2} = 0,\ \forall (f_1 \vee f_2) \notin F(g), \forall g \in G. \tag{2}$$

Each flight has to be allocated to a gate. A 'dummy gate' (gate $m+1$) is also included in this constraint to which ungated flights will be allocated:

$$\sum_{g=1}^{m+1} X_{g,f} = 1, \quad \forall f \in \{1, ..., n\}, \tag{3}$$

No two flights can be allocated to the same gate at the same time. If two flights overlap or are within a minimum time gap of each other then they must be assigned to different gates, Inequality 4. This gap is set to 10 minutes, since some time is needed to clear the gate. However, the precise value is not important for these experiments, since there is an objective to increase these gaps.

$$X_{f_1,g} + X_{f_2,g} \leq 1,\ \begin{array}{l} \forall f_1, f_2 \in F(g),\ \forall g \in G, f_1 \neq f_2, \\ (e_{f_2} \geq e_{f_1}) \wedge (e_{f_2} - l_{f_1} \leq 10). \end{array} \tag{4}$$

The mathematical formulation of the time gap constraint is given below, it introduces a new variable which is set if two flights are allocated to the same gate. The constraint is introduced only for gaps smaller than 1.5 hour, which are more than sufficient. The gaps which are smaller than 1.5 hour are maximised in the objective function using the new variable so that potential, minor delays on the day of operation can be absorbed.

$$U_{g,f_1,f_2} \geq X_{f_1,g} + X_{f_2,g} - 1,\ \begin{array}{l} \forall f_1, f_2 \in F(g),\ \forall g \in G, f_1 \neq f_2, \\ (e_{f_2} \geq e_{f_1}) \wedge (10 < e_{f_2} - l_{f_1} < 90). \end{array} \tag{5}$$

Some gates cannot be used simultaneously, because usage of one gate blocks usage of another one, this constraint is called a 'shadow restriction' [9]. Moreover, sometimes one large gate can be treated as either two smaller gates (for smaller aircraft) or one large gate (for a single large aircraft). These large gates are modelled as three separate gates with a shadowing constraint between them. Figure 3 shows a large gate ($G3$) which would be modelled as three gates: $G3F$ (full size gate), $G3L$ (the left half of $G3$) and $G3R$ (the right half of $G3$). $G3L$ and $G3R$ could be used at the same time by two small aircraft ($F2$, $F3$) but if there is a large aircraft ($F1$) on $G3F$ then neither $G3L$ nor $G3R$ can be used. Let $Sh$ denote the set of pairs of gates which have to be used exclusively (pairs $(G3F, G3L)$ and $(G3F, G3R)$ would belong to this set). The 'shadow restriction' can then be described as follows:

$$X_{f_1,g_1} + X_{f_2,g_2} \leq 1, \forall f_1 \in F(g_1),\ \forall f_2 \in F(g_2),\ \forall (g_1, g_2) \epsilon Sh, f_1 \neq f_2, \\ (e_{f_2} \geq e_{f_1}) \wedge (e_{f_2} - l_{f_1} \leq 10). \tag{6}$$

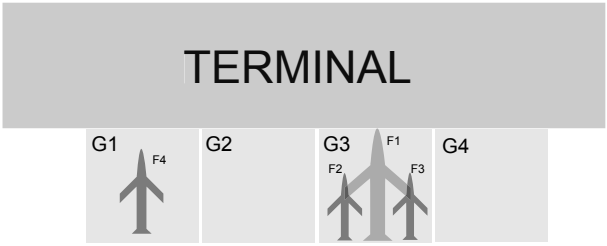



**Fig. 3.** Large gates can be used as two small gates by small aircraft

When an aircraft is at the airport for a long time, it is often towed from a gate to a remote stand so that it is not occupying a gate that could be used by other aircraft at that time. The aircraft is then usually towed back to the same or a different gate some time before the scheduled departure time. Although having many towing operations is not desired by airports (towed aircraft may cause additional conflicts on the taxiways) there is often no other option if a terminal does not have enough gates to serve all aircraft. It is assumed in this model that an aircraft is towed away from a gate if it is going to stay at the airport for more than 4 hours. This time can be changed according to airport preferences and has been chosen based upon the data set that has been used in the experiments. These long flights are modelled as two one-hour long flights. The first flight represents the arrival and lasts one hour starting from the scheduled arrival time. The second flight starts one hour before the scheduled departure time. As previously discussed, the middle part is not modelled and sufficient remote stands are assumed, which is usually the case for most airports. According to the data used in our experiments, an aircraft needs no more than an hour to do all servicing after it arrives and no more than one hour before it departs, the towing procedures are considered to be included in this hour. An exact time of

towing is not specified in the model, so the controllers may decide when to tow the flight and do so when the procedure causes fewest conflicts.

The conflict constraint is defined as follows:

$$X_{f_1,g_1} + \sum_{g \epsilon GR} \sum_{f \in C_d(f_1)} X_{f,g} \leq M, \forall f_1 \in F(g_1), \quad \forall g_1 \in GR, \quad \forall GR \in \{GR1, ..., GR5\} \tag{7}$$

where $GR$ is one of the gate groups from Figure 1 and $C_d(f_1)$ is a set of flights which can be in conflict with the departure time of flight $f_1$ if the flights are allocated to the same group of gates. The flights which are in $C_d(f_1)$ are identified from the scheduled arrival and departure times. It is assumed that arrivals up to 10 minutes before/after the departure of $f_1$ can conflict (i.e., a 10 minute long time gap is required). Similarly if an aircraft departs close to the departure time of $f_1$ it would be in $C_d(f_1)$, but the time margin can be shorter because it is a less conflicting situation so a 3 minute long time gap has been used. The 10 and 3 minute gaps can be set to any values desired, but have here been set to be large values to investigate and illustrate the effects of this constraint. An analogous set exists for arrival time conflicts so there is a separate set $C_a(f_1)$ which contains all flights which are in conflict with the arrival of $f_1$ and a constraint equivalent to Constraint 7 is applied. This idea of finding conflicting flights can also be viewed as a type of time window method. The conflicts within time windows would be identified by two windows for the arrival conflicts (short and long gaps) and two windows for the departure conflicts. But in a typical time window approach an artificial step size would usually be used to move the windows along the problem whereas here the positions of the windows are implicitly defined by the flight times, with consequent benefits of avoiding time discretisation problems. Our constraint allows up to $M$ flight movements for each group of gates. The number of allowed flights should be chosen depending on the particular airport preference. For this paper, the concept is the important element, resulting in lower congestion on the taxiways close to the gates.

**Objective.** The objective function is a weighted sum of four elements. It aims to ensure that time gaps between allocated flights are large enough to absorb minor delays, that the gate spaces are used effectively, that the airline preferences are maximised and that the number of remote ('dummy gate') allocations is minimised. The following function is minimised:

$$\sum_{f_1=1}^{n} \sum_{f_2=1}^{n} \sum_{g=1}^{m} p_{f_1,f_2} U_{g,f_1,f_2} + \sum_{f=1}^{n} \sum_{g=1}^{m} r_{f,g} X_{f,g} + \sum_{f=1}^{n} \sum_{g=1}^{m} a_{f,g} X_{f,g} + \sum_{f=1}^{n} dX_{f,dummy} \tag{8}$$

The first component of the function refers to the time gaps, the variable $U_{g,f_1,f_2}$ is forced to be one if two flights are at the same gate (see Inequality 5). The gap cost function $p_{f_1,f_2}$ that is used in the objective function to penalise such allocations is given by Equation 9:

$$p_{f_1,f_2} = \begin{cases} \frac{90}{(gap_{f_1,f_2})+1} & if\ 10 < gap_{f_1,f_2} < 90 \\ 1 & if \quad gap_{f_1,f_2} \geq 90, \end{cases} \tag{9}$$

where $gap_{f_1,f_2} = e_{f_2} - l_{f_1}$ if $e_{f_2} > e_{f_1}$ or $gap_{f_1,f_2} = e_{f_1} - l_{f_2}$ if $e_{f_1} > e_{f_2}$. The shape of $p_{f_1,f_2}$ is appropriate for the problem since the smaller gaps are penalised much more heavily than the larger gaps. Moreover $p_{f_1,f_2}$ saturates, which means that the penalty for really large gaps is fixed (equal to 1). It is reasonable to argue that only the gap from the immediately preceding flight on the gate should matter, and that gaps from earlier flights should be irrelevant.

The second component refers to effective usage of the gate space. The size of a flight that is allocated to a gate should correspond to the size of a gate, possibly being equal to the maximal size ($maxSize$) that gate $g$ can absorb. It cannot be larger, which is guaranteed by the first constraint (Equation 1). When it is smaller, then such allocation is penalised, to keep the larger gates clear in case they are needed later if flights have to be reallocated. The gate-flight size function $r_{f,g}$ that is used to penalise the allocation is given by Equation 10, where $biggestGateSize$ refers to the size of the biggest gate at a terminal.

$$r_{f,g} = \frac{maxSize(g) - size(f)}{biggestGateSize}, \tag{10}$$

The third component of the objective function refers to the airline preferences. The preferences are established based upon statistical data analysis, by checking how often particular gates have been used by particular airlines. The airline preference factor $a_{f,g}$ which is used in the objective function to weight the allocation of flight $f$ to gate $g$ is calculated according to Equation 11, where $maxFreq$ is the maximal value of all frequencies and refers to the most frequently used gate by the most popular airline at the terminal.

$$a_{f,g} = \begin{cases} 1 - \frac{freq_{f,g}}{maxFreq} & if\ freq_{f,g} > 0 \\ 1 & if\ freq_{f,g} = 0, \end{cases} \tag{11}$$

The last component refers to the 'dummy gate' allocations. 'Dummy gate' symbolises remote stands or holding places at an airport representing aircraft which cannot be allocated to gates so usage of it is strongly penalised.

## 5 Experimental Settings

The data which is currently used in the experiments has been provided by Manchester Airport and contains a record of five days of airport allocations: both the planned allocation and the final actual allocation, and includes information about gate usage (which gates have been used by which aircraft types and airlines), size group information for aircraft (that divides aircraft into groups according to their sizes) and sizes of gates. It is for Terminal 1 which is the busiest terminal at the airport. The numerical results shown in Section 6 have been obtained by solving five days of airport operations one after another.

Information about occupied gates for flights that arrive one day and depart the next day has been passed from day to day. Information about the airlines and airport preferences is hard to capture because it often relies on human

knowledge. In this research it has been estimated using statistical analysis of the data. For example, how often each airline uses each gate was measured and used as an estimation of airline preferences.

As mentioned in the previous sections the model contains several settings, which have been determined based upon our observations in relation to Terminal 1 at Manchester Airport and these are summarised in Table 1.

**Table 1.** Model settings

| Name | Symbol | Value |
|---|---|---|
| Average number of flights per day | $n$ | 110 |
| Number of gates | $m$ | 25 |
| Time at which flights are split for towing | | 240 min |
| Size of the gap for which $p$ saturates | | 90 min |
| Penalty for remote allocation | $d$ | 90 |
| Minimal allowed size of the gap between flights | | 10 min |
| Shortest time margin for conflicts within group of gates | | 3 min |
| Longest time margin for conflicts within group of gates | | 10 min |
| Maximal number of conflicting flights per GR | M | 2 |

There was only one shadowing constraint in the data, for the large gate 12, which can be divided into two smaller gates, left 13 and right 14. Figure 1 shows the groups of gates for Terminal 1. The gate-group adherence is as follows: GR1: 22, 24, 26, 28, 29, 32, 31; GR2: 21, 23, 25, 27; GR3: 2, 4, 6, 8; GR4: 10, 12, 13, 14; GR5: 1, 5, 7, 9, 11, 15.

## 6 Results and Discussion

The first part of this section shows how the new conflict constraint influences the final allocation. Results for one chosen day are displayed and allocations obtained for the model with and without the conflict constraint are briefly compared with the actual recorded allocation. The second part of the section shows numerical results obtained using five days of data. For these calculations the full configuration (with the conflict constraint) of the model has been used. The results are compared against the actual allocations recorded during the day of airport operation. All results have been obtained using CPLEX (version 12.2) executed on a reasonable powerful desktop PC (24GB RAM, 500GB disk space, 3.06 GHz Intel i7-950). An optimal solution was reached for each of the calculated days.

More detailed results are also shown for an illustrative day having 111 flights. Figure 4 compares three allocations for the day: Figure 4a shows the real allocation, Figure 4b the allocation obtained for the model without the conflict constraint and Figure 4c the allocation for the model with the conflict constraint. The horizontal axis of each of the charts shows time and the vertical axis the names of the gates, flights are symbolised by chains of markers. The gates are

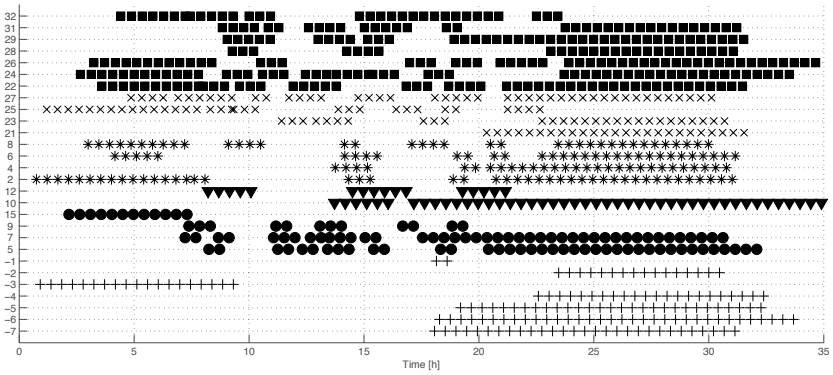
**Table 2.** Numerical results

| | Flights in total | On-Pier [%] | Remote [%] | Resolved conflicts |
|---|---|---|---|---|
| Model | 680 | 95 | 5 | 34 |
| Real | 626 | 92 | 8 | 0 |

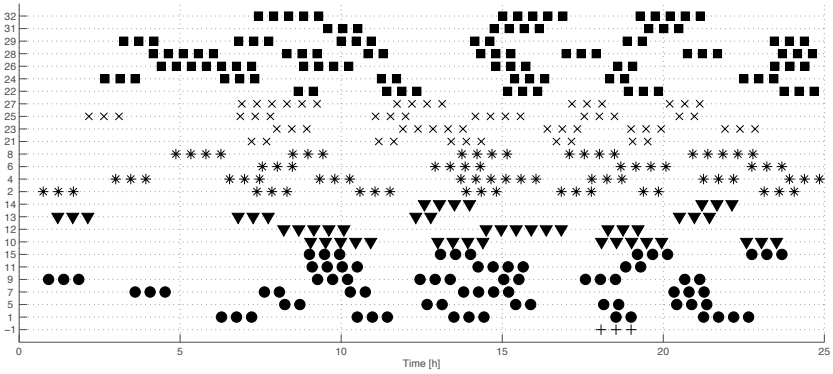
grouped, with each of the groups introduced in previous sections being assigned a different marker, so that it is easier to observe how the conflict constraint changes the allocation and resolves possible conflicts. All of the flights that have been assigned to the group of gates called GR1 are displayed as chains of squares (■, effectively rectangular bars), GR2 as chains of x-shaped markers (×), GR3 as chains of star-shaped markers (∗), GR4 as chains of triangles (▼) and GR5 as chains of dots(•). The remote allocations are also shown on the charts as chains of crosses (+). The negative gate numbers symbolise remote gates. Each remote allocation has been allocated to a different remote gate, so that it is easier to see how many remote allocations there are. Note that in practice the towed flights can be excluded from the conflict constraint thanks to the flexible implementation of towing. Since no strict towing time is assumed for each tow, the potential conflicts can be resolved by controllers by postponing or speeding up some tows if necessary. Four important observations can be made based upon the graph: (1) The number of conflicting allocations is reduced. (2) The real solution has fewer towed flights, since the long flights are kept on the gates in most cases, resulting in lack of gates available for flights more often, and therefore: (3) the remote stands having to be used more often. (4) The size of the time gaps between allocations is much bigger in the model solutions.

An important objective of the presented model is maximisation of time gaps between allocations (increasing the robustness of the solution). Figure 5 is introduced in order to show the distribution of the gap sizes in the allocation obtained for the full model (a) and for the real situation (b). The smallest gap in the model solution is 90 minutes while in the real situation there were many gaps which were smaller than that. The numerical results obtained for five days are presented in Table 2. The first column of Table 2 shows the total number of flights in the five days, the second one shows the percentage of on-pier allocations, the third the percentage of the remote allocations and the last the number of conflicts which have been resolved by the introduction of the new constraint.

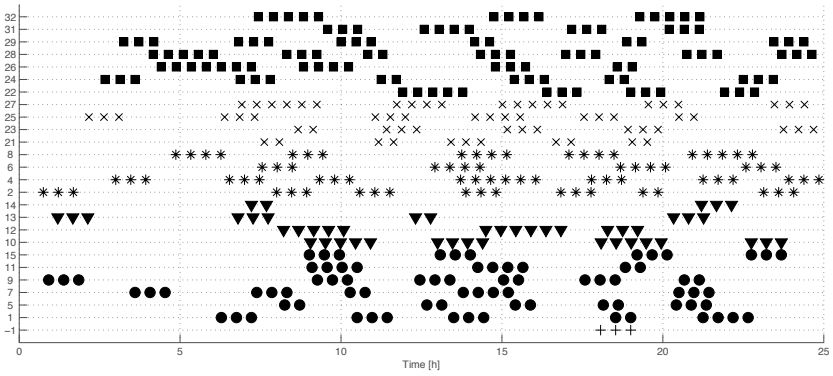
There are fewer remote allocations in the generated solution and 34 conflict constraints have been resolved. However, it can be observed based upon the total number of flights that the model introduces more tows into the solution, which may not be good, especially at very busy periods. Another drawback of the presented solution method is the calculation time, since it took in total over three days to calculate the five days of allocations and, therefore, heuristic methods will be considered now that the effects of the model have been established.




(a) Real allocation

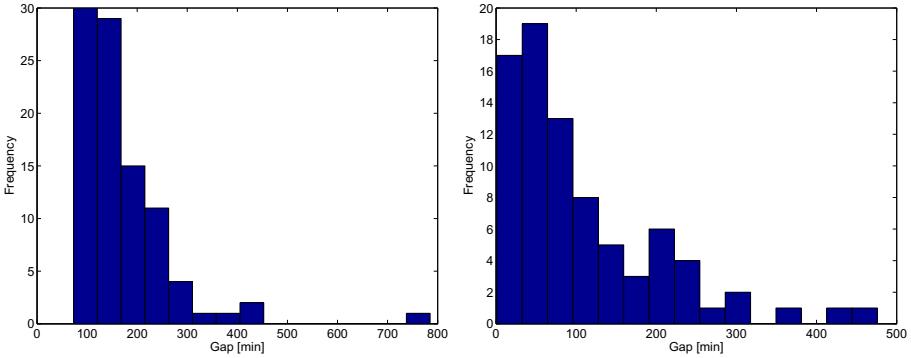



(b) Model without conflict constraint




(c) Model with conflict constraint excluding tows

**Fig. 4.** Results obtained for the example day: the real allocation (a), model without conflict constraint (b) and model with conflict constraint excluding towed flights (c)




(a) Model solution (b) Recorded allocation

**Fig. 5.** Gap size distribution

## 7 Conclusion

This paper considers the gate allocation problem and the conflicts which occur on the taxiways close to the gates. Integration of various airport operations is gaining increasing importance due to the increasing air traffic. Closer integration of different parts of airports could potentially result in smoother operations, fuel burn reductions and lower harmful gas emissions. The model of the gate allocation problem which is presented here moves this integration a step closer by incorporating some of the taxiway conflicts into the allocation process.

The major gains of the presented model compared to the real solution are an increase in the robustness of the solutions and a reduction in both the number of conflicts around the gates and the number of the remote allocations. However, there are significant differences in the number of tow operations. Having the towing procedure in the objective instead of explicitly splitting all the long flights would probably improve the performance of the model in this regard. Changing the conflict constraint into a soft constraint may be also a good idea, as the number of remote allocations and the resolved conflicts could then be balanced by setting appropriate weights in the objective function. The long calculation times indicate that alternative solution methods should be considered.

Future work will consider potential decompositions of the problem and/or heuristic approximations to speed up the calculations. Moreover, a deeper analysis of the conflicts which occur on taxiways, not only close to the gates but also further away and incorporating this into the allocation process appears to be a worthwhile area for future work. An analysis of the ground movement with a view to integration has already been provided by Ravizza et al. [13,14]. Their system assumes that the gates are already allocated and provides routing information based upon the assumed gate allocations. Integration of these approaches is an interesting research area moving forward. Finally, an assessment of the impact of the parameters and weights used in the model on the obtained results is also important future work.

## References

1. Airport Collaborative Decision Making, http://www.euro-cdm.org/
2. Bolat, A.: Procedures for providing robust gate assignments for arriving aircraft. European Journal of Operational Research 120(1), 63–80 (2000)
3. Burke, E.K., Kendall, G.: Multi-Objective Optimization. In: Search Methodologies, pp. 273–316. Springer (2005)
4. Cheng, Y.: Solving push-out conflicts in apron taxiways of airports by a network-based simulation. Computers & Industrial Engineering 34(2), 351–369 (1998)
5. Diepen, G., van den Akker, J.M., Hoogeveen, J.A.: Integrated Gate and Bus Assignment at Amsterdam Airport Schiphol. In: Ahuja, R.K., Möhring, R.H., Zaroliagis, C.D. (eds.) Robust and Online Large-Scale Optimization. LNCS, vol. 5868, pp. 338–353. Springer, Heidelberg (2009)
6. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: Aircraft and gate scheduling optimization at airports. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (2004)
7. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: The over-constrained airport gate assignment problem. Computers & Operations Research 32, 1867–1880 (2005)
8. Dorndorf, U., Drexl, A., Nikulin, Y., Pesch, E.: Flight gate scheduling: State-of-the-art and recent developments. Omega 35, 326–334 (2007)
9. Dorndorf, U., Jaehn, F., Pesch, E.: Modelling robust flight-gate scheduling as a clique partitioning problem. Transportation Science 42(3), 292–301 (2008)
10. Haghani, A., Chen, M.C.: Optimizing gate assignments at airport terminals. Transportation Research Part A: Policy and Practice 32(6), 437–454 (1998)
11. Kim, S., Feron, E., Clarke, J.: Assigning gates by resolving physical conflicts. In: AIAA Guidance, Navigation, and Control Conference (2009)
12. Kumar, P., Bierlaire, M.: Multi-objective airport gate assignment problem. In: Proceedings of the Swiss Transport Research Conference (2011)
13. Ravizza, S., Atkin, J.A.D., Burke, E.K.: A more realistic approach for airport ground movement optimisation with stand holding. Journal of Scheduling (2013), doi:10.1007/s10951-013-0323-3
14. Ravizza, S., Chen, J., Atkin, J.A.D., Burke, E.K., Stewart, P.: The trade-off between taxi time and fuel consumption in airport ground movement. Public Transport (2013), doi:10.1007/s12469-013-0060-1
15. Xu, J., Bailey, G.: The airport gate assignment problem: Mathematical model and a tabu search algorithm. In: Hawaii International Conference on System Sciences, vol. 3, pp. 30–32 (2001)
16. Yan, S., Huo, C.M.: Optimization of multiple objective gate assignments. Transportation Research Part A: Policy and Practice 35(5), 413–432 (2001)

# Comparing Revenue Optimization Models for Low Cost Carriers Supporting Connecting Flights - A Case Study

Ulrich Derigs, Benjamin Juds, and Andreas Palm

Department of Information Systems and Operations Research (WINFORS)
University of Cologne, Pohligstr. 1, 50969 Cologne, Germany
ulrich.derigs@uni-koeln.de, juds@gmx.de, andi@andipalm.de

**Abstract.** Traditionally low cost carriers (LCC) offer simple point-to-point relations only and follow the pricing paradigm to only offer one price at a time with prices increasing towards the end of the booking period. Recently, LCC have started to also offer connecting flights. With this change in the business model from point-to-point relations only, the booking as well as the revenue management system has to be adapted appropriately. In this paper we present a simple LP-model for revenue management at LCC which is based on the realistic situation that for every flight a set of possible prices is prespecified and which can consider direct as well as connecting flights. In a simulation study we demonstrate the efficiency and the advantage of this model over models established in practice where the optimal solution has to be adapted to the pricing schema, i.e prices have to be rounded and capacities have to be split.

## 1 Introduction

During the last 20 years the appearance of low cost carriers (LCC) like Easy Jet, Ryanair, Germanwings etc. has revolutionized air passenger transportation (see [1]). The business model of an LCC is substantially different from the traditional full-service carriers (FSC) like British Airways, KLM, Lufthansa etc. in many ways but can be characterized by one aspect, i.e., to reduce complexity on all levels. Thus the business model includes

- offering simple point-to-point relations with a single passenger class and a simple fare schema only with no-frills, i.e., passengers paying charges for extras like advanced seat reservation, snacks and drinks as well as prohibitively high rebooking fees,
- operating a single type of aircraft and flying to cheaper and less congested airports to cut charges and allowing fast turnaround times with the consequence of a high aircraft utilization,
- direct sales of tickets over proprietary call center/ booking systems avoiding cost for travel agents and use of reservation systems.

For the customer the main difference compared to full-service carriers is ticket price. With the need for high load factors and price being the only instrument for

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 199–212, 2013.

customer discrimination, capacity and revenue management that allows highly reactive and time-based pricing is vital for LCC.

Revenue Management (RM) has been "invented" by American Airlines in 1987 and has developed into an important strategy in many business areas like car rental, hotels etc. Revenue management deals with the problem of effectively using perishable resources or products in industries or markets with high fixed cost and low margins, which are price-segmentable (see [4], [9], and [5]). RM can shortly be described as "selling the right product to the right customer at the right time through the right channel for the right price" (cf. [8], [6]). Thus, RM is a complex tactical planning problem which has to be supported by sophisticated forecasting models/systems and optimization models/systems.

In contrast to the pricing and capacity planning at full-cost carriers, which operate a hub and spoke network enabling numerous relations through connecting flights, RM for LCC follows a completely different logic. The central (dynamic) pricing paradigm of a LCC is to offer for each flight only one single price at a time and to increase this price constantly towards the end of the booking period. Now, as the low-cost market is becoming more saturated, some LCC also try to achieve further growth by offering connecting flights. Yet, the pricing paradigm remains still valid and it is, therefore, the central assumption for our development. With this trend traditional RM systems for LCC designed only for point-to-point traffic become ineffective.

In this paper we present an optimization model which can handle direct as well as connecting flights. The model reflects the real situation that the airline has established a basic price schema $P$ and that it specifies for every flight a subset of potential prices in advance. The advantage of our model is twofold. First, there is no need to adapt the solution to the price schema, i.e., to round optimal prices and to split optimal capacities as is necessary for some established systems which by this post-processing loose the optimality property. Secondly, the model can quite straightforwardly be extended to also consider connecting flights. We present computational results with this model on real data.

The intention of our case study has been the development of a model which can easily replace the existing model without changing the basic logic of the overall process and the systems. The development or testing of new approaches for modeling demand and customer choice, as well as different pricing paradigms such as bid price based methods and virtual nesting, is not in the focus of the paper.

The purpose of the computational study has been the identification of the loss in revenue if the RM-system for point to point connections is used for small connection networks with manual adaptions only (as it was done by the airline) instead of applying our simple LP-model.

The paper is structured as follows. In Section 2 we shortly introduce revenue management in LCC. In Section 3 after presenting a basic model we introduce our models. In Section 4 we report some experiments with these models on empirical data and we close with a conclusion in Section 5.

## 2 Revenue Management at Low Cost Carriers

In this section we first introduce the basic concepts of revenue management and outline the logic which is the conceptual basis of RM systems in FSC. Then we analyze and describe the different situation a LCC faces. This exposition is to settle ground for the models given in Section 3.

Revenue Management (RM) addresses the situation where a fixed amount of perishable goods or services has to be sold to different customers willing to pay different prices. Thus it addresses pricing as well as capacity planning. RM is essential in situations where the fixed cost related to the provision of the perishable goods or service far outweighs the variable costs. RM is critically important in the airline industry. Here the schedule offers a fixed capacity of seats on a flight. Airplane seats are perishable goods since as soon as a flight has left the gate unsold seats on board are worth nothing.

The approach taken by a FSC is to divide the seat capacity on an aircraft into different products, so called *fare classes*, based on different classes of service like first class, business and economy with different degrees of flexibility like minimum stay, advanced booking etc. The different products are offered and sold at different prices at the same time. Then, RM is to manage the relationship between price, demand, capacity and customer behavior in a way that maximizes revenues.

Due to its business model, consumer segmentation using fare classes related to different service levels or restrictions makes no sense for LCC. All booking requests refer to the same product and each request is dealt with uniquely and offered a price which is economically appropriate at the time of request. Requests are treated by a first-come first-served process with passengers getting cheaper fares by booking earlier and the airline controlling revenue by increasing price, i.e., sequentially closing low price fare classes. Note that there are ad-hoc offers if demand does not meet expectation and special offers like blind booking which are to attract revenue for otherwise unused seats. Also, ancillary revenue from non-ticket sources has become an important revenue component which we do not consider in this study. Note that overbooking is also neglected in our study, here strategies of simply enlarging aircraft capacity are applied.

In the LCC business environment where classes have no differentiation other than price the knowledge of price sensitivity has become even more crucial. Generally, a market-response-function is used to express the connection between price and demand. And such a function has to be estimated separately for every single flight by analyzing historical data for so called reference flights. Those are flights with the same relation on similar days of the week, comparable flight times and same seasonality etc.

Now customer behavior, i.e price sensitivity, changes over the booking period. Therefore, several forecasts of market response are estimated for different time intervals before the day of departure with intervals getting smaller when approaching the day of departure. Figure 1 shows an example of two market-response-functions created by the forecasting component.

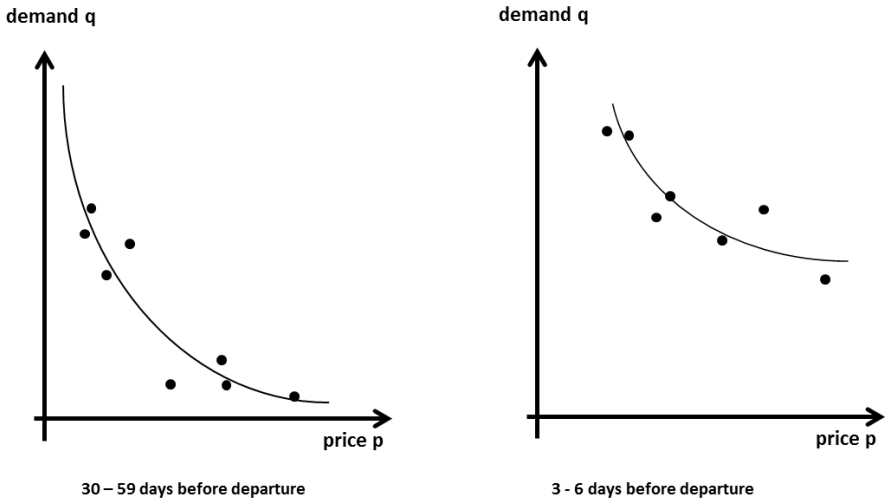



**Fig. 1.** Exemplary market-response-functions

Based on these forecasts, the optimization module of RM has to determine for each time interval a set of price/capacity pairs as basis for a fare class control, which is then implemented by the online booking system.

Since pricing has become the only mechanism to shape, the demand pattern fine-grained approaches respecting specific and small price differences are necessary. Since generating valid forecasts is not always possible, especially under changing markets with new relations and competitors, LCC need highly responsive RM systems where both forecasting and optimization are frequently recalculated. Also, the system should allow interactivity, i.e., manual corrections of booking limits in light of the actual booking situation. In the next section we focus on the optimization module of a RM system for LCC.

## 3 Optimization Models

At the heart of the RM-optimization model is a revenue function which is to be maximized subject to a capacity constraint as described in [7]. They assume that the reference time unit is a single day, i.e., demand and price are fixed over a single day. Then, considering a booking period of $N$ days we have to determine a price $p_i$ and a capacity $q_i$ for each day $i \in T$, with $T := \{1, 2, 3, ..., N\}$. Let the aircraft/flight capacity be $Q$ then the model in the notation of [7] is as follows:

$$max \sum_{i \in T} p_i q_i \qquad s.t. \sum_{i \in T} q_i \leq Q$$

[7] show how this model can be solved for different functional forms of the demand curve $q_i = q_i(p_i)$ by analyzing the derivative of the associate lagrangian

function. The following model QPDF (quadratic problem with direct flights), which is established in practice, is an operational implementation of this idea. It incorporates that the reference units are not single days but a set of time intervals $I = \{365 - 181, 180 - 121, \ldots, 7 - 0\}$, which are also used by the forecasting module to estimate the market response. The market response during time interval $i \in I$ for a price $p$ is denoted as $MR_i(p)$[1].

Model 1: QPDF

$$\max \sum_{i \in I} q_i p_i \tag{1.1}$$

$$\text{subject to} \sum_{i \in I} q_i \leq Q \tag{1.2}$$

$$q_i \leq MR_i(p_i) \tag{1.3}$$

$$p_{365-181} \leq p_{180-121} \leq \ldots \leq p_{7-0} \tag{1.4}$$

$$\frac{\sum_{i \in I} q_i}{Q} \geq minLF \tag{1.5}$$

$$p_i \geq 0 \qquad \forall i \in I \tag{1.6}$$

$$q_i \geq 0 \qquad \forall i \in I \tag{1.7}$$

The objective function (1.1) expresses the revenue calculated as price per seat times the number of seats made available for that price. (1.2) ensures that total aircraft capacity is not exceeded and (1.3) limits the number of seats for each interval to the market response. (1.4) models that prices increase over the booking period and with (1.5) we require a minimum loadfactor. Note that this constraint is neglected in some optimization runs.

This model is rather straightforward and easy to understand. Yet, unfortunately, it is computationally rather complex and impractical. First, the objective function is quadratic and non-convex. Thus specific solvers are required and the solution time is significantly larger than solution times for equal sized linear programs. Even more critical is the fact that a solution will most likely be non-integer and not be compatible with the prespecified price schema. Thus, it will for instance recommend to sell 34.45 tickets for a price of 76.13 Euro. With a solution like that, two problems arise. First, since it is not possible to sell 0.45 tickets the solution has either to be rounded, which may cause feasibility problems or the model has to be modified to an integer program. Defining $q_i$ as an integer variable turns the model into a quadratic mixed integer problem and the solution becomes even more complex and time-consuming. Secondly, and more importantly, an LCC in general does not want to sell tickets at arbitrarily patched prices like 76.13 Euro. Germanwings, for instance, uses a predefined schema of potential prices for all flights starting from 19.99 Euro and increasing in specific steps including 69.99 Euro and 79.99 Euro, for instance.

[1] For the notation we refer to [9].

As these models indicate, the problem can be seen as a combination of two sub-problems: determining the price schema and calculating assigned contingents or capacities. These sub-problems are separated in the RM approach applied by Germanwings, for instance. Here a discrete price-structure is specified for every market, i.e., a bundle of related flights beforehand and then economic capacities are determined through optimization. Figure 2 displays an example for such a price/contingent plan with the assignment of prices and contingents to time intervals in the booking period.

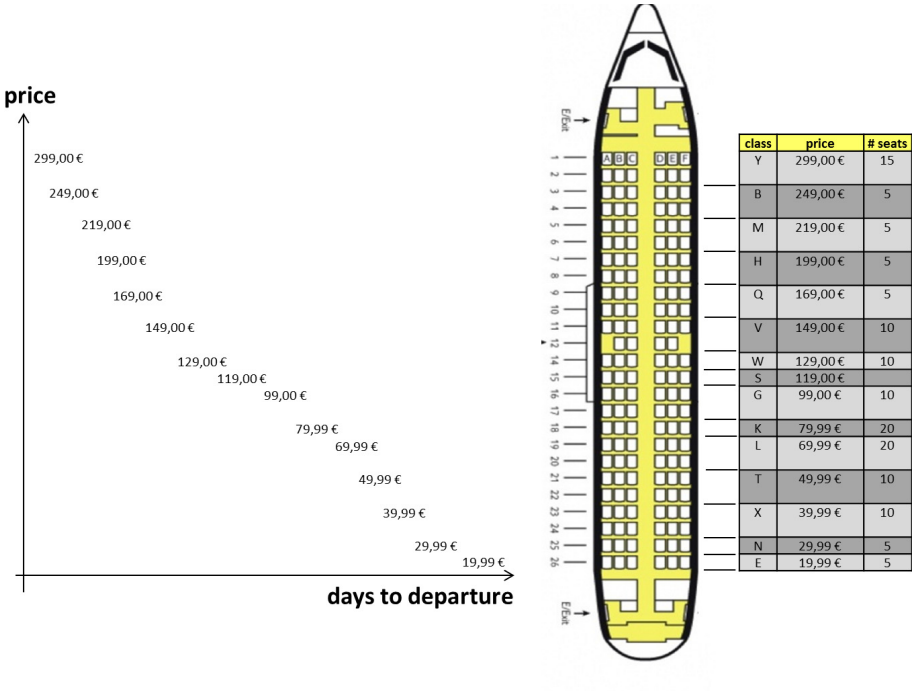



**Fig. 2.** Example for a price and contingent schema

Now, when applying model QPDF to this schema the prices $p_i$ given in the solution have to be matched to the prices contained in the predetermined schema. Thus for the example above one would assign 60% of the recommended number of seats, i.e., 20 seats to fare class 69.99 Euro and 40%, i.e., 14 seats to fare class 79.99 Euro. This postoptimal procedure is rather easy to perform and can be implemented such that no capacity constraint is violated, yet, the resulting pricing and allocation scheme is no longer guaranteed to be optimal at all.

To cope with these deficiencies we propose a different approach and model which is directly based on the pricing schema of the LCC. The concept of using discrete prices has already been proposed by [9]. Instead of finding arbitrary prices we restrict the solution to prices in the set $P$ of valid prices with $P = \{19.99, 29.99, 39.99, \ldots, 489, 499\}$, for instance. Thus, we let the optimization select a subset of prices from $P$ and determine the optimal number of seats

$q_{i,p}$ assigned to price $p \in P$ in time interval $i \in I$. To ease notation and to abstract from a specific time scale we set $I := \{1, 2, ..., t\}$, i.e., we replace the interval description by the associated index or serial number with $t$ referencing the interval closest to departure. We denote this model as LPDF (linear problem with direct flights).

Model 2: LPDF

$$\max \sum_{i \in I} \sum_{p \in P} p \cdot q_{i,p} \tag{2.1}$$

$$\text{subject to} \sum_{i \in I} \sum_{p \in P} q_{i,p} \leq Q \tag{2.2}$$

$$q_{i,p} \leq d_{i,p} \cdot b_{i,p} \qquad \forall p \in P, i \in I \tag{2.3}$$

$$\sum_{p \in P} b_{i,p} = 1 \qquad \forall i \in I \tag{2.4}$$

$$b_{i,p} \leq B_p \qquad \forall i \in I, p \in P \tag{2.5}$$

$$\sum_{p \in P} B_p \leq n \tag{2.6}$$

$$\sum_{p \leq p_0} b_{i,p} \geq \sum_{p \leq p_0} b_{i-1,p} \qquad \forall p_0 \in P, i \in I \backslash \{1\} \tag{2.7}$$

$$B_p \in \{0, 1\} \qquad \forall p \in P \tag{2.8}$$

$$b_{i,p} \in \{0, 1\} \qquad \forall i \in I, p \in P \tag{2.9}$$

$$q_{i,p} \geq 0, \text{integer} \qquad \forall i \in I, p \in P \tag{2.10}$$

The model is easy to interpret. The objective function states total revenue with prices as parameters. Total seat capacity is respected by (2.2). Given the discrete set of prices $P$, the market-response-function has to be evaluated at those discrete points only, i.e., we set $d_{i,p} := MR_i(p)$. Therefore, we can linearize the demand constraint (2.3). The binary variable $b_{i,p}$ specifies if a price $p \in P$ is selected for interval $i \in I$ . Then (2.4) ensures that only one price per time interval will be used. In the model we can limit the overall number of prices used to $n$, say. For that purpose the binary variable $B_p$ specifies if a price $p \in P$ is used at all and with (2.5) and (2.6) we limit the the total number of different prices to $n$. With (2.7) we ensure that prices are increasing over time. If the model is applied for reoptimization during the booking period we would add constraint

$$q_{i,p} \geq sold_{i,p} \qquad \forall p \in P, i \in I$$

which ensures that the capacity of each price includes the number of tickets that have already been sold.

Defining all prices $p$ to be an element of the schema and all contingents $q_{i,p}$ to be integer values avoids any necessity for rounding the optimal solution. Moreover

the objective function is linear. For solving the model standard MIP/IP-solvers like CPLEX or GUROBI are available. Note that the dimension of the model is determined by the cardinality of $P$ and $I$. With these cardinalities being around 15 and 7, respectively, instances can be solved in milliseconds of CPU-time which is an essential property if problem instances with a large number of flights have to be (re-) optimized in a daily run. We will report computational experience in Section 4.

The models presented so far are designed for optimizing independent flights. They are not intended for revenue management on connecting flights. Germanwings, for instance, started to offer connecting flight services in 2007. Thus RM for complex not only point-to-point flight networks becomes necessary. Here our model offers a straightforward modification.

In a complex flight network we have to distinguish between two concepts: flights and legs. Here a *leg* is a direct connection between two airports while a general *flight* may be composed from a sequence of legs.

Now, while the price schema for the connecting flight may be specified independently from the schemata of the basic single leg flights, the price offered for a connecting flight has to be based on the booking situation of the associated legs. For instance, the relevant fare classes for the (two) single leg flights are compared and the higher fare class is chosen for the connecting flight. Then the airline offers the price which is assigned to this fare class in the pricing schema of the connecting flight. If the price is accepted, i.e., the customer books the connecting flight then in the reservation system both single leg flights are booked. Figure 3 displays an example for this procedure. Note that with this and any alternative logic quite undesirable pricing decisions may occur, i.e., the price of the connecting flight may be higher than the sum of the two single leg prices.

Leg 1

| class | price | state |
|---|---|---|
| 4 | 79.99 | open |
| 3 | 49.99 | open |
| 2 | 39.99 | open |
| 1 | 19.99 | closed |

Leg 2

| class | price | state |
|---|---|---|
| 4 | 149.99 | open |
| 3 | 99.99 | open |
| 2 | 69.99 | closed |
| 1 | 39.99 | closed |

$\longrightarrow$

Connecting flight

| class | price |
|---|---|
| 4 | 199.99 |
| 3 | 129.99 |
| 2 | 99.99 |
| 1 | 59.99 |

**Fig. 3.** Conversion from connecting flight booking into two direct flight bookings

In the occurrence of connecting flights, different flights compete for leg capacities and we cannot partition the optimization by flights. Thus, in our model we have to determine prices and capacities for all flights simultaneously. Note that based on the structure of the flight network there may be a decomposition into leg-disjoint subsets possible. For the following we assume that such a decomposition has already been performed on a higher level.

Model LPCF (linear problem for connecting flights) for (re-)optimizing the entire network is an immediate extension of our first model. Let $F$ be the set of flights offered and $L$ the set of legs which are flown. Again, $P$ denotes the set of

valid prices used by the LCC then the model has to specify capacities $q_{f,i,p}$ for each price, flight and time interval. We introduce a binary parameter $u_{f,l}$ with $u_{f,l} = 1$ if flight $f$ uses leg $l$ and $u_{f,l} = 0$ otherwise.

Model 3: LPCF

$$\max \sum_{f\in F}\sum_{i\in I}\sum_{p\in P} p \cdot q_{f,i,p} \tag{3.1}$$

$$\text{subject to} \sum_{f\in F}\sum_{i\in I}\sum_{p\in P} u_{f,l} \cdot q_{f,i,p} \le Q_l \qquad \forall l \in L \tag{3.2}$$

$$q_{f,i,p} \le d_{f,i,p} \cdot b_{f,i,p} \qquad \forall p \in P, f \in F, i \in I \tag{3.3}$$

$$\sum_{p\in P} b_{f,i,p} = 1 \qquad \forall f \in F, i \in I \tag{3.4}$$

$$b_{f,i,p} \le B_{f,p} \qquad \forall f \in F, i \in I, p \in P \tag{3.5}$$

$$\sum_{p\in P} B_{f,p} \le n \qquad \forall f \in F \tag{3.6}$$

$$q_{f,i,p} \ge sold_{f,i,p} \qquad \forall p \in P, f \in F, i \in I \tag{3.7}$$

$$\sum_{p\le p_0} b_{f,i,p} \le \sum_{p\le p_0} b_{f,i-1,p} \qquad \forall f \in F, p_0 \in P, i \in I\backslash\{1\} \tag{3.8}$$

$$B_{f,p} \in \{0,1\} \qquad \forall p \in P, f \in F \tag{3.9}$$

$$b_{f,i,p} \in \{0,1\} \qquad \forall p \in P, f \in F, i \in I \tag{3.10}$$

$$q_{f,i,p} \ge 0, \text{integer} \qquad \forall p \in P, f \in F \tag{3.11}$$

In the objective function we sum revenue over all flights. While class limits are on flights, seat capacity constraints are on legs. Here we allow each leg to have a different capacity $Q_l$ which gives flexibility for use of different aircraft types. As for the first model no rounding is necessary, the model can be solved by standard MIP-solvers, yet, the dimension of an instance is now depending on the number of prices in schema $P$, the number of flights $F$ and the number of time intervals to be optimized. We will report on computational results in the next section.

## 4 Application - A Simulation Study

We have compared the models introduced in the previous section using real data provided by Germanwings. This data covers a subset of the flight network with six direct flights and five connecting flights between seven European airports. The starlike network is shown in Figure 4. All flights are scheduled on a weekly basis and operated up to 3 times a day which results in a total of 118 direct flights per week.

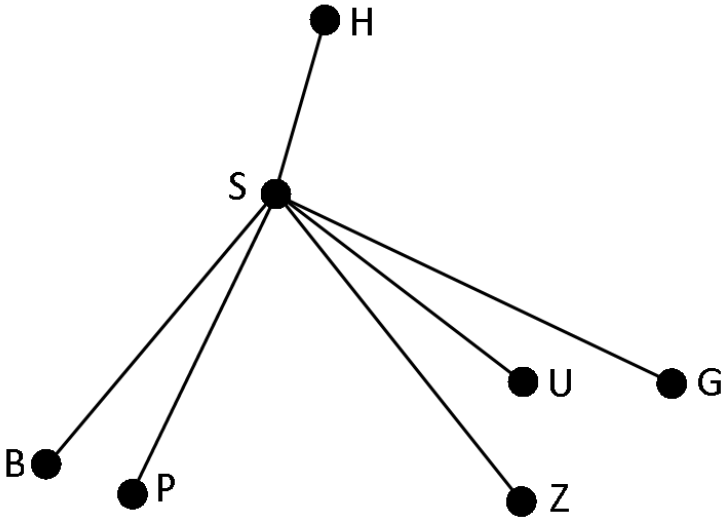



**Fig. 4.** Shape of the flight network used for our experiments

Booking records for all flights departing within a period of three months with a booking period of one year in advance were provided. Booking records contain information on the number of tickets sold, the price paid and the date the booking was placed. Booking records resulting from special promotions have been removed. Finally, the (manually created) pricing schemata for the (connecting) flights as well as information on aircraft capacity was given.

Note that in the data set about 96 percent of the total number of tickets sold were results of direct flight bookings and thus only four percent were for connecting flights. This small number is due to the fact that the concept of connecting flights had only been introduced recently, and thus this distribution may not be representative in the future. To analyze the effect of connecting flight bookings on the performance of the models, we created four additional instances from the given data set with different ratios between direct and connecting flight bookings. First, a set only containing direct flights was derived from the basic set by simply deleting all connecting flights. Then, instances were created by duplicating and time shifting connecting flight bookings, i.e., for a randomly determined connecting flight booking an additional booking with the same price and quantity was created two days earlier than the original booking. This way we generated another three instances with 8, 14 and 25% connecting flights.

To compare the performance of the three models introduced in the previous section we have implemented a simulator that mimics the entire revenue management process. A process chart of the simulator is shown in Figure 5. The simulator proceeds along the time line and it considers all flights simultaneously, i.e., it starts on the first day of the booking period of the first flight and ends after booking for the last flight has been closed. Then, during execution the simulator advances in steps of one day executing the three activities: forecasting, optimization and booking for each flight in sequence.

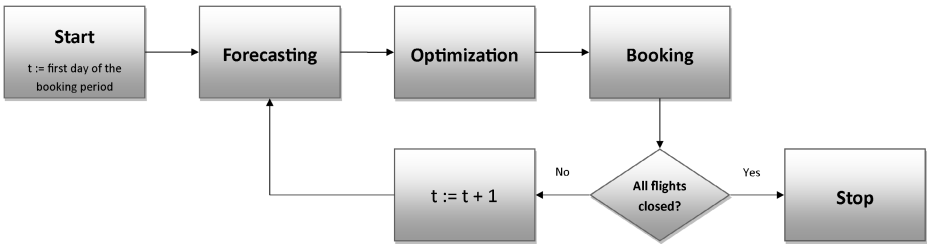



**Fig. 5.** Simulator process chart

In the *forecasting*-step, we compute market-response-functions from the historical booking data using the Q-forecasting model described by [3] and [2] which is used in their Passenger Origin-Destination Simulator (PODS). In this approach demand by fare-classes is converted to the so called "Q-equivalent demand" for every flight. This demand is then estimated using a standard forecasting algorithm and the resulting forecasted q-equivalent demand is reconverted to a demand by fare-classes structure. The logic of both conversions is based on the *sell-up theory*, also introduced by [2].

Estimation of several parameters by performing analyses and regressions on a large amount of historical data is an essential prerequisite for the quality of this forecasting procedure. As our data covers only a small time period of three month this was not possible. Therefore, our forecasting procedure is by far not optimal and solid forecasts could be generated if the parameters were estimated like in PODS with many historical data records. Yet, we are not focussing on the forecasting aspect of the RM process but on the effectivity and efficiency of optimization. For the sole purpose of comparing the performance of the different optimization models, in particular with connecting flights, the quality seems to be sufficient. For a realistic test applying the models in combination with the forecasting method used by the airline to the current booking situation would be the optimal scenario.

Now, depending on the optimization model selected in the simulation run, forecasting has to be applied in two different ways. For LPCF separate forecasts for each direct and connection flight are created. For models QPDF and LPDF, which are not able to handle connecting flights, the demand for connecting flights is added to the related direct flights.

Forecasting is followed by an *(re)optimization* of all future flights. Models LPDF and LPCF are solved using Gurobi Optimizer 3.01 Academic Edition. For solving QPDF we used the KNITRO 6.0 solver. With model QPDF the optimal values are rounded as described in Section 3. Models LPDF and LPQF return integer demands and valid prices, thus no rounding is necessary. The optimization result is a set of price-quantity pairs that state how many tickets should be offered for that particular price.

Following optimization, the last activity in each simulator step is *booking*. Here, the historical booking records are used to derive a list of booking requests. For these the time the booking was placed and the price paid are relevant for the

simulation. Now for each future flight all historical bookings that were placed at the same time (relative to the departure date) are selected and used as booking requests from (virtual) customers. This method ensures that requests are following a realistic distribution and variation. The booking logic itself is rather straightforward. For every booking request we assume that the customer accepts, i.e., the booking is placed, if the offered price resulting from the optimization is not exceeding the price paid; otherwise the booking is declined and the customer leaves the system and will not return.

Note that when simulating model QPDF or LPDF all booking requests for connecting flights are translated into bookings for the two related direct flights as described in Section 3.

To evaluate the models for use in a revenue management system, total revenue over all flights is the most important performance indicator. Besides this measure we count the total number of tickets sold.

Since we have simulated different and hypothetical data sets absolute revenue values are not meaningful. Also, we only want to evaluate the relative performance among the models on different scenarios. Therefore, we have calculated for every dataset the revenue obtained from offering every customer the price paid as recorded in historical data which can be used as a meaningful reference value. Note that since we assume in our simulation that customers decline to book if the offered price is higher than the price paid this reference value is the highest achievable value. In Table 1 we state the revenues as well as the number of tickets sold as percentage of this benchmark value together with the average CPU - running time in sec measured as the average time needed in every simulation step to reoptimize all flights for one day.

As expected model QPDF performs worst on all datasets and models LPDF and LPCF yield significant higher revenues. This is partially due to the fact that no rounding is necessary. A closer look at the two LP models reveals that

**Table 1.** Simulation results

| Instance | Model | Total revenue | Tickets sold | Runtime |
|---|---|---|---|---|
| 0% | QPDF | 60.17% | 64.95% | 24.584 |
| | LPDF | 66.05% | 68.11% | 8.328 |
| | LPCF | 65.89% | 68.90% | 17.305 |
| 4% | QPDF | 60.12% | 64.88% | 24.466 |
| | LPDF | 66.19% | 68.18% | 8.189 |
| | LPCF | 66.18% | 69.05% | 17.196 |
| 8% | QPDF | 59.78% | 64.37% | 24.863 |
| | LPDF | 65.73% | 68.08% | 8.465 |
| | LPCF | 66.28% | 69.42% | 17.271 |
| 14% | QPDF | 59.37% | 63.96% | 24.614 |
| | LPDF | 65.21% | 67.79% | 8.211 |
| | LPCF | 66.56% | 69.91% | 17.516 |
| 25% | QPDF | 58.69% | 62.64% | 24.962 |
| | LPDF | 64.12% | 67.66% | 8.293 |
| | LPCF | 67.18% | 70.36% | 17.938 |

they perform much the same on the first three datasets while a substantial gap occurs in the datasets with 14 and 25 percent connecting flights. Therefore, we can conclude that with an increasing amount of connecting flights the application of the simple one-leg flight models becomes intractable and the application of network optimization models becomes inevitable.

When comparing runtimes there is no significant difference between the datasets. Applying models QPDF and LPDF leads to a set of several small instances, one per flight, to be solved in every optimization step of the simulator. Here QPDF instances are harder to solve and it takes significantly more time than to solve the corresponding linear LPDF instances. Applying model LPCF all flights are optimized in a single model instance and hence the size is considerably larger. Nevertheless, the running time of LPCF is significantly less than the time to solve the suboptimal QPDF-model.

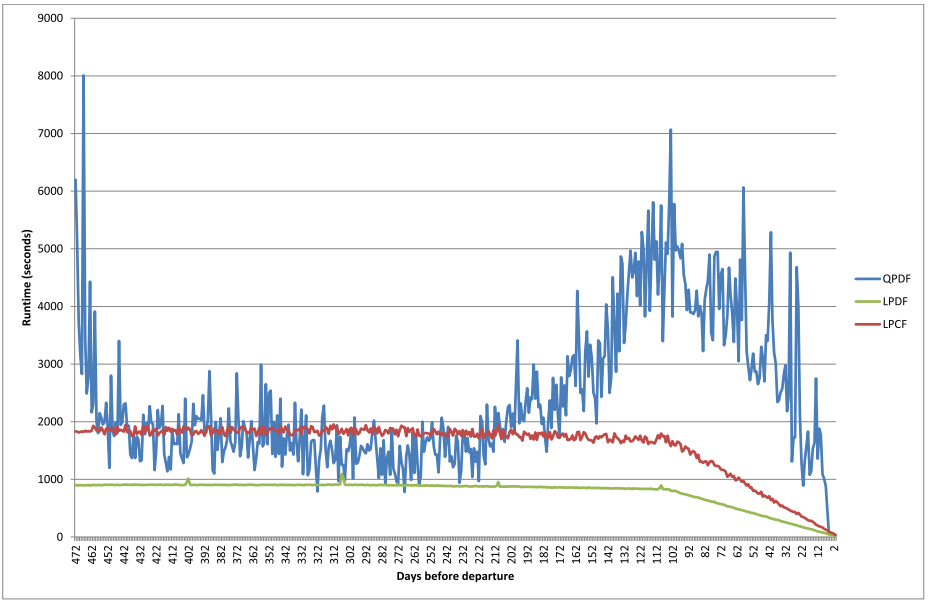



**Fig. 6.** Optimization runtime during the booking period

Figure 6 shows another interesting difference. Over the booking period the runtime of QPDF for reoptimizing all future flights shows a large variation and we can observe an increase in time near the departure date. On the other hand both linear models show a more or less constant optimization time over the whole booking period with a decrease at the end of the booking period since fewer flights have to be optimized.

## 5 Final Remarks

In this paper we have introduced a new optimization model for revenue management at low cost airlines which is based on the planning paradigm of assigning capacities under fixed price schemata. The motivation for introducing this model was the development that low cost carriers have recently changed their business model of offering direct flights only to also offering connecting flights. Now, our model can easily be extended to complex networks with connecting flights. In a simulation study using real world data we have shown that the model can be solved rather efficiently using standard software. Analyzing different scenarios with increasing portions of connecting flight bookings it becomes evident that in such environments RM systems which are based on models for direct flights only become highly ineffective.

## References

1. Barlow, G.: easyJet: An Airline that Changed our Flying Habits. In: Yeoman, I., McMahon-Beattie, U. (eds.) Revenue Management and Pricing: Case Studies and Applications, ch. 2, pp. 9–23. Thomson (2004)
2. Belobaba, P.P., Hopperstad, C.: Algorithms for Revenue Management in Unrestricted Fare Markets. In: INFORMS Revenue Management Section Meeting, Cambridge (2004)
3. Cléaz-Savoyen, R.L.: Airline Revenue Management Methods for Less Restricted Fare Structures. Master's thesis, Massachusetts Institute of Technology (Mai (2005)
4. Cross, R.G.: Revenue Management: Hard-Core Tactics for Market Domination. Broadway Books, New York (1997)
5. Klein, R., Steinhardt, C.: Revenue Management - Grundlagen und Mathematische Methoden. Springer, Heidelberg (2008)
6. Makhlouf, K.: Gestaltung von Flugpreisen in einem Niedrigpreisumfeld: Wie man Kostenvorteile kreativ und dynamisch nutzen kann. In: Hess, T., Doeblin, S. (eds.) Turbulenzen in der Telekommunikations- und Medienindustrie, ch. 11, pp. 163–177. Springer (2006)
7. Malighetti, P., Paleari, S., Redondi, R.: Pricing strategies of low-cost airlines: The Ryanair case study. Journal of Air Transport Management 15(4), 195–203 (2009)
8. Smith, B.C., Leimkuhler, J.F., Darrow, R.M.: Yield management at american airlines. Interfaces 22(1), 8–31 (1992)
9. Talluri, K.T., van Ryzin, G.J.: The Theory and Practice of Revenue Management. Springer, Heidelberg (2004)

# From Preparedness to Recovery: A Tri-Level Programming Model for Disaster Relief Planning

Takashi Irohara[1], Yong-Hong Kuo[2], and Janny M.Y. Leung[3]

[1] Department of Information and Communication Sciences
Faculty of Science and Technology, Sophia University
Tokyo, Japan
irohara@sophia.ac.jp
[2] Faculty of Management and Administration
Macau University of Science and Technology
Avenida Wai Long, Taipa, Macau
yonghongkuo@gmail.com
[3] Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
janny@se.cuhk.edu.hk

**Abstract.** This paper proposes a tri-level programming model for disaster preparedness planning. The top level addresses facility location and inventory pre-positioning decisions; the second level represents damage caused by the disaster, while the third level determines response and recovery decisions. We use an interdiction framework instead of a stochastic or chance-constrained model. This allows the extent of damage to be treated as a parameter to facilitate scenario exploration for decision-support. We develop an iterative dual-ascent solution approach. Computational results show that our approach is efficient, and we can also draw some insights on disaster relief planning.

**Keywords:** disaster preparedness, inventory pre-positioning, facility location, humanitarian logistics, bi-level programming.

## 1 Introduction

The past decade has witnessed major natural and man-made disasters all around the world. From blizzards, earthquakes, flooding, hurricanes, tsunami and wildfires, to famine, oil-spills and radiation accidents, no continent has been spared. Practitioners have recognised that strategic location of depots and pre-positioning of inventory greatly facilitate the speed and efficiency of evacuation and/or delivering supplies in the crucial days immediately after disaster strikes. In determining the level of supplies to pre-position and their location, one must balance the cost of maintaining such stockpiles against the probability of disasters that would utilize them in relief efforts. Anticipating scenarios where some of the depots might

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 213–228, 2013.

be damaged and rendered un-usable should somehow be considered in preparedness planning. The interdiction approach used in the tri-level model presented in this paper allows the consideration of "worst-case" damage scenarios, instead of an expected-value approach. This provides a framework where the trade-offs between the effectiveness of relief efforts and preparedness costs can be explored as a function of the extent of damage caused by the disaster.

In the following section, we give a brief literature review. In Section 3, we present the tri-level programming framework. We show that the framework can be re-formulated as a bi-level programme in Section 4. For a simplified model, this can be reduced further to an equivalent mixed-integer programme. In Section 5, we present an iterative dual ascent solution approach using cutting planes. In Section 6, computational results indicate that our solution approach is efficient and viable for realistic-sized problems. From these preliminary computations, we can also draw some insights on disaster relief planning.

## 2 Brief Literature Review

We give below a brief literature review of papers related to disaster relief operations. The reader is also referred to the following tutorial and review papers: Tomasini and van Wasenhove (2009), Balcik et al. (2010) and Celik et al. (2012). Disaster management can be classified into four main stages: *mitigation, preparedness, response* and *recovery*. In the past two decades, researchers have investigated problems arising in each of these stages.

Much research on disaster relief operations focussed on evacuation route planning, for example, Lu et al. (2005). Huang et al. (2012) discussed the importance of equity considerations in route planning for humanitarian logistics. As early as 1991, Sherali and Carter (1991) recognized the impact of shelter locations on evacuation time, and developed a joint location-route planning model. Jia et al. (2007) presented a scenario-based facility-location model for large-scale emergencies. Doerner et al. (2008) studied a multi-criteria model for public-facility location in tsunami-prone areas. Some practitioners and researchers have begun to address the efficacy of good preparedness planning on the effectiveness of response. Rawls and Turnquist (2010, 2011, 2012) considered the issues related to pre-positioning and delivery of supplies in the immediate aftermath of the disaster. Balcik and Beamon (2008) considered the network design problem for relief distribution. Yushimoto and Ukkusuri (2008) considered the location and pre-positioning of supplies by considering the reliability of delivery paths. Bozorgi-Amiri et al. (2012) developed a mixed-integer non-linear programming model for location and inventory pre-positioning. Nurre et al. (2012) considered the problem of restoration of infrastructure post-disaster. Mete and Zabinsky (2010) developed a stochastic model for the location and delivery of medical supplies. Some authors have addressed the linkage between asset pre-positioning and post-disaster logistics deployment (e.g. Salmeron and Apte, 2010; Doyen, et al., 2012), but these models adopted a stochastic approach by considering expected cost over all scenarios. Noyan (2012) considered risk-averseness

by using conditional Value-at-risk as the risk measure. O'Hanley and Church (2011) is one of the few papers that takes an interdiction (worse-case) approach when considering location of emergency facilities. As far as we know, no systematic optimization–based approach exists that provides an anticipatory tri-level framework as the one presented in this paper.

## 3 Model Formulation

In this section, we present a formulation for the tri-level disaster preparedness planning model. The first and third levels correspond to decision-making in the preparedness stage and the response/recovery stage, respectively, while the second level corresponds to the damage inflicted by the disaster. The three levels are linked by the fact that decisions at each level are affected by decisions taken in the previous level, and also the fact that the objectives anticipate subsequent costs. Our model includes the following **parameters and sets:**

$I$ = the set of possible locations for evacuation centres,
$J$ = the set of communities,
$p_j$ = size of population of community $j$,
$d_{ij}$ = distance from community $j$ to location $i$,
$C_i$ = capacity of evacuation centre at location $i$,
$m$ = maximum distance between a community and its assigned centre,
$E$ = maximum number of evacuation centres to be established,
$f_i$ = fixed cost of establishing an evacuation centre at location $i$,
$h_i$ = cost per unit of inventory pre-positioned at location $i$,
$D$ = number of evacuation centres destroyed by the disaster,
$F_i$ = fixed cost of (re)-establishing a centre at location $i$ post-disaster,
$H_i$ = cost per unit of delivering inventory to location $i$ post-disaster, and
$T_{ik}$ = cost per unit of transferring inventory from location $i$ to $k$ post-disaster;

and the following binary and continuous **decision variables:**

$$x_{ij} = \begin{cases} 1 \text{ if community } j \text{ is assigned to evacuation centre in location } i, \\ 0 \text{ otherwise;} \end{cases}$$

$$y_i = \begin{cases} 1, \text{ if an evacuation centre is established in location } i, \\ 0, \text{ otherwise;} \end{cases}$$

$z_i$ = amount of inventory pre-positioned in location $i$,

$$\delta_i = \begin{cases} 1, \text{ if the evacuation centre in location } i \text{ is destroyed by the disaster,} \\ 0, \text{ otherwise;} \end{cases}$$

$$u_i = \begin{cases} 1 \text{ if an evacuation centre is (re-)established in location } i \text{ post-disaster,} \\ 0 \text{ otherwise;} \end{cases}$$

$r_{ij}$ = proportion of population of community $j$ (re-)assigned to location $i$ post-disaster,
$w_i$ = amount of inventory available in location $i$ post-disaster,
$s_i$ = amount of new inventory supplied to location $i$ post-disaster, and
$t_{ik}$ = amount of inventory transferred from location $i$ to location $k$ post-disaster.

We first present the formulation of each level separately, and then describe the linkages.

The first level addresses the decision of inventory pre-positioning, the location of evacuation centres and assignment of the communities to the centres, formulated as follows:

$$(\mathbf{L1}): \quad \min_{x,y,z} \quad \sum_{i\in I}(f_i y_i + h_i z_i) + \mathit{RecoveryCost}(x, y, z; \delta; r, u, s, t)$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i\in I} y_i \leq E, && (1)\\
& \sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J, && (2)\\
& d_{ij}x_{ij} \leq m y_i \qquad \forall i \in I, j \in J, && (3)\\
& z_i \leq C_i y_i \qquad \forall i \in I, && (4)\\
& \sum_{j\in J} p_j x_{ij} \leq z_i \qquad \forall i \in I, && (5)\\
& y_i, x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J, && (6)\\
& z_i \geq 0 \qquad \forall i \in I. && (7)
\end{aligned}$$

The objective minimizes the sum of the costs of establishing the centres, of maintaining the pre-positioned inventories and the recovery costs post-disaster. The constraint (1) limits the maximum number of evacuation centres to be established. Constraints (2) and (3) determine the assignment of communities to their designated centres, which must be within a distance $m$. Constraints (4) and (5) ensure that the level of inventory to be pre-positioned at the centres established is within the capacity of the centre and sufficient to support the designated communities should a disaster occur. For modelling simplicity, we aggregate all necessary supplies as one commodity and assume that one unit is needed per person. (Considering commodities separately does not substantially alter the structure of the model. The amount of different commodities needed are generally proportional to the number of people served and likely to be sourced from the same location.)

In the second level, the disaster is considered as an adversary that tries to inflict maximum damage. The "decision" for this level is formulated as:

$$(\mathbf{L2}): \quad \max_{\delta} \quad \mathit{RecoveryCost}(x, y, z; \delta; r, u, s, t)$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i\in I} \delta_i \leq D, && (8)\\
& \delta_i \leq y_i \qquad \forall i \in I, && (9)\\
& \delta_i \in \{0,1\} \qquad \forall i \in I. && (10)
\end{aligned}$$

The righthand side of constraint (8) is the maximum number of centres destroyed by the disaster, which can be considered as a parameter for the severity of the damage, on which sensitivity analysis can be performed. Constraints (9) state that a centre is not destroyed unless it has been established (in the first level).

The third level represents the response and recovery stage after the disaster, where the decisions involve (re-)establishment of evacuation centres, re-supply and/or transfer of goods to/between centres, and re-allocation of people to the centres. It is formulated as follows.

$$(\mathbf{L3}): \min_{r,u,s,t,w} \mathit{RecoveryCost}(x,y,z;\delta;r,u,s,t) = \sum_{i\in I}(F_i u_i + H_i s_i) + \sum_{i\in I}\sum_{k\in I} T_{ik} t_{ik}$$

$$\begin{aligned}
\text{subject to} \quad 1-(y_i-\delta_i) &\geq u_i, && \forall i \in I, && (11)\\
\sum_{i\in I} r_{ij} &= 1 && \forall j \in J, && (12)\\
u_i + (y_i-\delta_i) &\geq r_{ij} && \forall i \in I, j \in J, && (13)\\
C_i(y_i-\delta_i) &\geq w_i && \forall i \in I, && (14)\\
w_i &\leq z_i && \forall i \in I, && (15)\\
\sum_{k\in J} t_{ik} &\leq w_i && \forall i \in I, && (16)\\
w_i + s_i + \sum_{k\in J} t_{ki} - \sum_{k\in J} t_{ik} &\geq \sum_{j\in J} p_j r_{ij} && \forall i \in I, && (17)\\
u_i &\in \{0,1\} && \forall i \in I, && (18)\\
r_{ij}, t_{ik}, s_i, w_i &\geq 0 && \forall i,k \in I, j \in J. && (19)
\end{aligned}$$

The *RecoveryCost* includes cost of (re-)establishing centres and cost of supplying and transhipping inventory to and among operational centres, in order to cover the population accommodated at the centres. Constraint (11) states that a centre can be established in a location not previously used or in a location where an established centre has been destroyed. Constraints (12) and (13) indicate in which evacuation centres (that are operational post-disaster) the population for each community should be accommodated. In this level, the population of each community may be split among several centres. The variables $w_j$ with constraints (14) and (15) determine how much inventory is available post-disaster; we assume all inventory is rendered unusable if a centre is destroyed. Finally, constraints (17) ensure that the inventory at each centre (including existing, newly-supplied and transhipped) is sufficient to support the population accommodated at the centre post-disaster.

The linkage between the three levels of the formulation is via the *RecoveryCost* post-disaster. The rationale for the formulation is that the decisions at the preparedness stage (in the first level) should anticipate the recovery cost post-disaster. Our formulation also takes an interdiction approach rather than an expected-value or chance-constrained approach in modelling the disaster. Probability-based formulations are much affected by the pool of scenarios considered and their associated probabilities, which may be difficult to ascertain for

rare events. The interdiction approach can be viewed as a "worst-case" approach, and sensitivity analysis can be performed on the parameter $D$ which denotes the number of facilities destroyed by the disaster in the second level.

## 4 Solution Methodology

As far as we know, there is no solution methodology developed for tri-level programming. In this section, we present an integrated formulation for the tri-level model. In the next section, we present a dual-ascent iterative solution approach. An efficient solution approach is important, since the coverage of the planning area could be quite extensive. According to a report by the World Bank, at the peak of the relief effort after the Tohoku earthquake and tsunami, over 470,000 people were housed in 2,500 evacuation facilities (World Bank, 2012).

### 4.1 Level 3 as a Location-Allocation Problem

The decision variables $u_i$ for (**L3**) represent location decisions. We note that once these locations are selected, the remaining decisions can be optimised by solving a network-flow problem. For fixed values of $(x, y, z)$, $\delta$ and $u$, (**L3**) is equivalent to a min-cost network flow problem, where the supply nodes correspond to $i$ with $u_i + y_i - \delta_i = 1$ and a special node $n$ represents the source of supply for new inventory. The demand nodes correspond to $J$, each with demand $p_j$. The solution to this min-cost network flow problem gives the minimum recovery cost given the initial preparedness decisions, the damage incurred in Level 2, and the "location" decisions $u_i$. The flows determine the allocation of the population in the communities to the operational evacuation centres and the corresponding delivery of supplies. Thus, (**L3**) is a location-allocation problem, which is still NP-hard. To facilitate subsequent discussion, we denote the optimal value to (**L3**) for given $x, y, z$ and $\delta$ as $R(x, y, z; \delta)$.

### 4.2 Re-formulation of Level 2

For fixed $x, y, z$ and $\delta$, the optimal recovery cost for Level 3 can be determined by solving a location-allocation problem as described in the previous subsection. Next, we consider a re-formulation of Level 2 in a column-generation format. Let $\{\delta_1, \delta_2, \cdots, \delta_Q\}$ be the extreme points of

$$\mathcal{F} \in (x, y, z) = conv\{\delta \in \mathbf{B}^I : \sum_{i \in I} \delta_i \leq D; \delta_i \leq y_i, \forall i \in I\}.$$

Then Level 2 can be re-formulated as:

$$(\mathbf{L2a}): \qquad \max_{\lambda} \sum_{q=1}^{Q} \lambda_q R(x, y, z; \delta_q)$$

$$\text{subject to} \qquad \sum_{q=1}^{Q} \lambda_q = 1, \tag{20}$$

$$\lambda_q \geq 0, \quad \text{for } q = 1, 2, \cdots, Q. \tag{21}$$

We note that the dual of (**L2a**) is:

$$\begin{aligned} (\mathbf{D-L2a}): \quad & \min \quad \pi \\ \text{such that} \quad & \pi \geq R(x,y,z;\delta_q), \quad \text{for } q = 1,2,\cdots,Q. \end{aligned} \tag{22}$$

### 4.3 An Integrated Formulation

With the optimal value of the Level 3 formulation defined as $R(x,y,z;\delta)$ associated with a given $(x,y,z)$ and $\delta$, we can consider the Level 3 optimization as implicit within the Level 2 optimization. Thus, our model can be considered as a bi-level programming problem. By adopting a column-generation formulation for Level 2, the corresponding bi-level programme can be further re-formulated as an integrated model as follows:

$$(\mathbf{M}): \qquad \min_{x,y,z;\delta,\pi} \quad \sum_{i\in I}(f_i y_i + h_i z_i) + \pi$$

subject to (1) to (7) and

$$\pi \geq R(x,y,z;\delta) \quad \forall \delta \in \mathcal{F}\in(x,y,z). \tag{23}$$

The formulations for Level 2 and 3 are implicitly included as the constraints (23). We note, of course, that (23) contains many constraints (even if we consider only the $\delta$'s corresponding to the extreme points of $\mathcal{F}\in(x,y,z)$), and the righthand sides are obtainable only by solving NP-hard location-allocation problems.

## 5 A Dual Ascent Solution Approach

In this section, we present a dual ascent solution approach for the integrated model for the case when no (re-)establishment of centres is possible after the disaster. This corresponds to the situation when the time-frame of the recovery stage is quite short.

### 5.1 Evaluation of $R(x,y,z;\delta)$

In this case, the Level 3 formulation $(L3)$ can be simplified as:

$$(\mathbf{L3a}): \qquad \min_{r,s,t,w} \sum_{i\in I} H_i s_i + \sum_{i\in I}\sum_{k\in I} T_{ik} t_{ik}$$

subject to (12), (14), (15), (16), (17) and (19).

We introduce a new set of variables $v_{ij} = p_j r_{ij}$ representing the number of people of community $j$ accommodated at centre $i$. As discussed in Subsection 4.1, for fixed values of $y, \delta$ and $z$, (**L3a**) becomes a min-cost network flow problem for the network $G = (\{I_1 \cup n\} \cup I_2 \cup J, (\{I_1 \cup n\} \times I_2) \cup (I_2 \times J))$, where $I_1 = I_2 = I$ and node $n$ represents the source of supply for new inventory. The nodes in $I_1$

are supply nodes with a supply of $z_i(y_i - \delta_i)$; node $n$ is a supply node with supply $\sum_{j \in J} p_j - \sum_{i \in I} z_i(y_i - \delta_i)$. The nodes in $J$ are demand nodes, each with demand $p_j$. $I_2$ are transshipment nodes. The arc $(n, i)$ has a cost of $H_i$ while arc $(i, k) \in I_1 \times I_2$ has a cost of $T_{ik}$. As noted in Subsection 4.1, the solution to this min-cost network flow problem gives the minimum recovery cost given the initial preparedness decisions and the damage incurred in Level 2.

By considering the dual of this network flow problem, we note that $R(x, y, z; \delta)$ is a linear function of $z \cdot (y - \delta)$. Specifically, let $\alpha_i$ be the dual variable corresponding to the flow balance constraint for supply node $i \in I_1$, $\beta_j$ be the dual variable corresponding to the flow balance constraint of demand node $j \in J$, and let $\alpha_0$ be the dual variable for the supply node $n$. Therefore,

$$R(x, y, z; \delta) = \sum_{i \in I} (\alpha_i - \alpha_0)\, z_i(y_i - \delta_i) - \sum_{j \in J} (\beta_j - \alpha_0) p_j. \tag{24}$$

Without loss of generally, we can set $\alpha_0 = 0$, since one of the flow-balance constraints is redundant in a network flow problem. We note that $R(x, y, z; \delta)$, unfortunately, is not linear in the decision variables of (**L1**).

### 5.2 An Iterative Solution Approach

Our iterative approach begins with solving a relaxation of (**M**) which ignores constraints (23), and then successively refines this relaxation by adding constraints (23) with the righthand side written as a function of $(x, y, z)$.

We define the following *Restricted Master Problem*:

$$(\mathbf{RM}): \qquad \min_{x,y,z;\pi} \quad \sum_{i \in I} (f_i y_i + h_i z_i) + \pi$$

$$\begin{aligned}
\text{subject to} \quad \sum_{i \in I} y_i &\leq E, && && (25)\\
\sum_{i \in I} x_{ij} &= 1 && \forall j \in J, && (26)\\
d_{ij} x_{ij} &\leq m y_i && \forall i \in I, j \in J, && (27)\\
z_i &\leq C_i y_i && \forall i \in I, && (28)\\
\sum_{j \in J} p_j x_{ij} &\leq z_i && \forall i \in I, && (29)\\
y_i, x_{ij} &\in \{0, 1\} && \forall i \in I, j \in J, && (30)\\
z_i &\geq 0 && \forall i \in I, && (31)\\
\pi &\geq \sum_{i \in I} \alpha_i^{\delta}\ \bar{z}_i^{\delta}(y_i - \delta_i) - \sum_{j \in J} \beta_j^{\delta} p_j, && \delta \in \mathcal{D}. && (32)
\end{aligned}$$

In the dual ascent approach we propose here, constraints (32) are to be generated iteratively given each set of values $(x, y, z)$ considered. The set $\mathcal{D}$ contains

vectors $\delta \in \mathbf{B}^I$ each corresponding to a constraint generated. We note that the definition of $R(x, y, z; \delta)$ contains the terms $z_i y_i$ and hence is non-linear. Therefore, we assume the value of $z$ to be fixed for each constraint generated, so (32) become linear constraints to facilitate solution of (**RM**) using mixed-integer programming solvers. The dual ascent approach is as follows:

**Dual Ascent Algorithm** (**DA**)
*Step 0:* (Initialisation) $\mathcal{D} = \emptyset$. Let $V = 0, k = 1$.
*Step 1:* (Restricted Master Problem) Solve (**RM**), let the optimal solution be $(\bar{x}, \bar{y}, \bar{z})$ and the optimal value be $\bar{V}$.
*Step 2:* (Termination) If all extreme points of $\mathcal{F}\in(\bar{x}, \bar{y}, \bar{z})$ are already in $\mathcal{D}$, or if $k \geq K$, terminate. (The complete solution is obtained by solving (**L2a**).) Otherwise,
*Step 3:* (Cut Generation) Consider some $\delta \in \mathcal{F}\in(\bar{x}, \bar{y}, \bar{z})$ not in $\mathcal{D}$. Solve (**L3a**) to obtain the dual values $\alpha$ and $\beta$. Add $\delta$ to $\mathcal{D}$ and the corresponding constraint to (**RM**). Increase $k$ by 1. Go to *Step 1.*

### 5.3 Cut Generation

The efficiency of the dual ascent algorithm depends heavily on how quickly it converges, since each iteration involves solving the *Restricted Master Problem* which is a mixed-integer-programming problem. Thus, efficiency might be improved if more than one cut is generated per iteration. Of course, if constraints corresponding to *all* the extreme points of $\mathcal{F}\in(\bar{x}, \bar{y}, \bar{z})$ are generated at each iteration, the number of iterations needed is expected to be small. However, this entails enumerating these extreme points and also adding a large number of cuts to (**RM**). In our approach, we propose a heuristic for identifying three values of $\delta$ to be used for cut generation in *Step 3* of (**DA**), as follows. The heuristic is based on the **ADD-DROP** heuristic for facility location (Kuehn and Hamburger, 1963).

**Cut Generation Heuristic** (**CG**)
*Step 0:* (Initialisation) Let $\delta^a = 0, \delta^b = 0, \delta^d = \bar{y}$. Let $R_{max} = 0$. Let $I^a = I^d = \bar{I} = \{i \in I : \bar{y}_i = 1\}$.
*Step 1:* (Termination of ADD) If $I^a = \emptyset$, go to *Step 3.* Otherwise,
*Step 2:* (ADD) For each $i$ in $I^a$, consider $\delta = \delta^a + \mathbf{1}_i$ and solve (**L3a**). Let $i^*$ be the index that corresponds to the highest optimal value. We set $\delta^a = \delta^a + \mathbf{1}_{i^*}$. If the optimal value is larger than $R_{max}$, then we set $\delta^b = \delta^a$ and update $R_{max}$. Go to *Step 1.*
*Step 3:* (Termination of DROP) If $|I^d| < D$, terminate. Otherwise,
*Step 4:* (DROP) For each $i$ in $I^d$, consider $\delta = \delta^d - \mathbf{1}_i$ and solve (**L3a**). Let $i^*$ be the index that corresponds to the highest optimal value. We set $\delta^d = \delta^d - \mathbf{1}_{i^*}$. If the optimal value is larger than $R_{max}$, then we set $\delta^b = \delta^d$ and update $R_{max}$. Go to *Step 3.*

If no centre is destroyed by the disaster, the recovery cost is zero. The ADD part of the heuristic successively selects one additional centre to destroy to cause the

maximum additional recovery cost, until $D$ centres are destroyed. The DROP part of the heuristic starts with all centres destroyed, and greedily "releases" centres from damage to minimize the reduction in recovery cost. We track the recovery cost of the $\delta$ values considered in the course of the heuristic, and also select the $\delta$ value with the maximum $R(x, y, z; \delta)$. Cuts corresponding to all three $\delta$ values will be added in the dual ascent algorithm.

Each iteration of the heuristic entails solving up to $|\bar{I}|$ network flow problems, so could still be quite computationally expensive. We can consider experimenting with other selection heuristics for $\delta$ for cut generation. One possibility to consider is some way of approximating the Shapley value for each location $i$, by keeping track of the solution values of (**L3a**) each time it is solved.

## 6 Computational Results

This section reports on computational experiments on the efficiency of the solution approach proposed. By examining the results, we can also draw some insights on disaster relief planning. As indicated in Section 5, our experiments consider only the cases when no (re-)establishment of evacuation centres is possible after the disaster (i.e. $u_i = 0 \ \forall i \in I$). The computations were done on an Intel(R) Core(TM)2 Quad CPU Q9400 2.66 GHz personal computer with 2.87 GB of RAM. We used CPLEX 12.1 as our mixed-integer solver (branch-and-cut), with the optimality tolerance set as 0.5%.

### 6.1 Data Sets and Parameters

We generate our data sets in a similar fashion as Huang et al. (2012). The locations of evacuation centres and communities are generated randomly with a uniform distribution on a 1000 × 1000 square. The distance between two locations, $d_{ij}$, is defined as the Euclidean distance. Some model parameters are generated randomly according to the discrete uniform distributions listed in Table 1. Other parameters are set as follows: $m = 300$, $E = 20$, $f_i = 10C_i$ and $T_{ik} = d_{ik}$. We consider seven different problem sizes: (i) $|I| = 30, |J| = 50$; (ii) $|I| = 30, |J| = 100$; (iii) $|I| = 40, |J| = 80$; (iv) $|I| = 40, |J| = 100$; (v) $|I| = 50, |J| = 100$; (vi) $|I| = 50, |J| = 150$; and (vii) $|I| = 60, |J| = 100$, and for each problem size, we consider three values for the maximum number of evacuation centres destroyed by the disaster (i.e. $D = 3, 6$ and 9). Thus, there are 21 instances in total.

**Table 1.** The probability distributions of the randomly generated parameters

| Parameter | $C_i$ | $h_i$ | $H_i$ | $p_j$ |
|---|---|---|---|---|
| Probability distribution | U(1,500) | U(1,10) | U(1,50) | U(1,100) |

### 6.2 Computational Study: Efficiency

Each instance was solved using the dual ascent approach ((**DA**) with (**CG**)) as described in Section 5. Results are reported in Table 2. The number of iterations refers to the number of times the restricted master problem (**RM**) was solved (Step 1 in (**DA**)) until the solution converges. We also report the solution time for solving the first (**RM**) (with no constraints (23)), the total time spent in solving (**RM**) for all the iterations, and the total computational time until convergence.

**Table 2.** Numbers of iterations and solution times for the computational instances

| Instance | $\|I\|$ | $\|J\|$ | $D$ | Number of iterations | Solution time for 1st (**RM**) (sec) | Solution time for all (**RM**) (sec) | Total time (sec) |
|---|---|---|---|---|---|---|---|
| 1a | 30 | 50 | 3 | 4 | 0.44 | 0.48 | 4 |
| 1b | 30 | 50 | 6 | 7 | 0.44 | 0.69 | 6 |
| 1c | 30 | 50 | 9 | 9 | 0.42 | 0.59 | 9 |
| 2a | 30 | 100 | 3 | 6 | 6.53 | 9.54 | 32 |
| 2b | 30 | 100 | 6 | 4 | 6.53 | 250.67 | 266 |
| 2c | 30 | 100 | 9 | 5 | 6.55 | 6.91 | 25 |
| 3a | 40 | 80 | 3 | 3 | 269.59 | 269.65 | 279 |
| 3b | 40 | 80 | 6 | 8 | 269.72 | 274.41 | 308 |
| 3c | 40 | 80 | 9 | 8 | 269.58 | 272.18 | 305 |
| 4a | 40 | 100 | 3 | 3 | 2114.75 | 2114.77 | 2139 |
| 4b | 40 | 100 | 6 | 4 | 2164.36 | 2169.6 | 2203 |
| 4c | 40 | 100 | 9 | 7 | 2145.64 | 2185.57 | 2243 |
| 5a | 50 | 100 | 3 | 3 | 18.58 | 18.60 | 69 |
| 5b | 50 | 100 | 6 | 5 | 18.48 | 22.65 | 133 |
| 5c | 50 | 100 | 9 | 7 | 18.55 | 24.71 | 167 |
| 6a | 50 | 150 | 3 | 3 | 521.77 | 521.83 | 889 |
| 6b | 50 | 150 | 6 | 3 | 507.01 | 507.06 | 896 |
| 6c | 50 | 150 | 9 | 6 | 506.19 | 533.02 | 1443 |
| 7a | 60 | 100 | 3 | 5 | 299.47 | 300.05 | 492 |
| 7b | 60 | 100 | 6 | 5 | 299.63 | 304.98 | 466 |
| 7c | 60 | 100 | 9 | 5 | 311.05 | 315.80 | 457 |

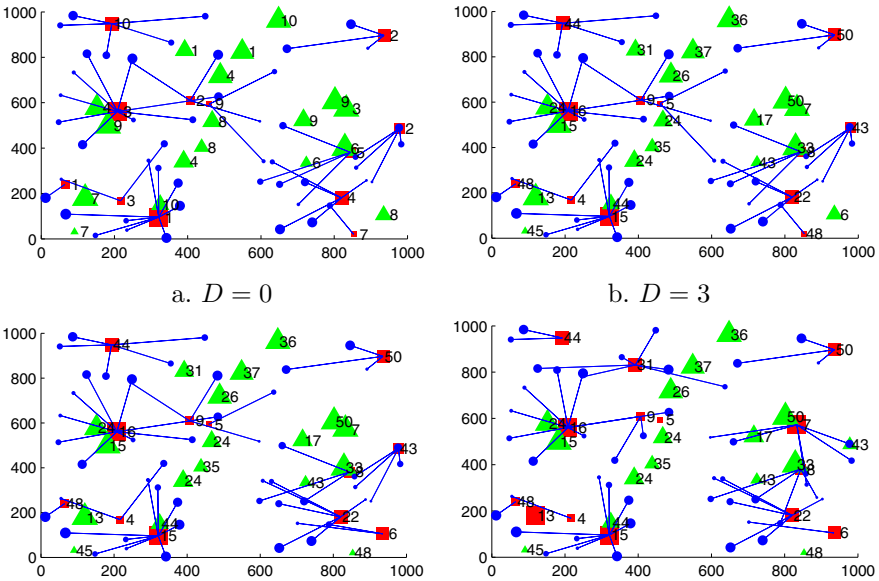
The results indicate that the proposed algorithm ((**DA**) together with (**CG**)) converges very quickly — with fewer than 10 iterations in all instances — due to the strength of the cuts generated in (**CG**). Interestingly, the computational time does not increase monotonically as $D$ increases. In several cases when the damage is low or severe, perhaps the low-cost options are *easier* to identify, the solution converges quickly. When the damage is moderately high, many different subsets of centres may have comparable costs, and the algorithm takes a longer time to determine the optimal solution.

We also note that a major proportion of computational effort is spent on solving the (**RM**)s, in particular the first (**RM**). Since the first restricted master problem does not contain constraints (23), it is equivalent to determining centre locations and inventory pre-positioning without consideration of *Recovery Cost*.

Thus, our results indicate that a model that considers recovery cost post-disaster adds only a modest amount of computation time compared to one that considers only pre-positioning and location costs pre-disaster.

### 6.3 Computational Study: Insights

In this section, we present some insights drawn from the computational results. We illustrate our findings with Instances 1a-c with $|I| = 30$ and $|J| = 50$; similar results are observed for the other problem sizes. Figure 1 depicts the optimal solutions for $D = 0, 3, 6$ and 9. The established evacuation centres are denoted by red squares, unused evacuation centre locations by green triangles, and communities by blue circles. The size of a node is proportional to its capacity or population size. A blue edge connects a community to its assigned centre. Note that, in an optimal solution, an established centre carries pre-positioned inventory if and only if there is a blue edge connecting it to a community. In Figure 1.a, an evacuation centre is labelled with its cost per unit of inventory pre-positioned; in Figures 1.b to 1.d, the label indicates the cost per unit of delivering inventory post-disaster.



a. $D = 0$ b. $D = 3$

c. $D = 6$ d. $D = 9$

**Fig. 1.** The locations for the case $|I| = 30, |J| = 50$

Note that the case of $D = 0$ (Figure 1.a) corresponds to (**RM**) without constraints (23), that is, when the recovery costs are not considered. In this case, when no disaster strikes ($D = 0$), the optimal solution has fewer evacuation centres established to reduce fixed costs and tends to establish evacuation centres of

low inventory cost. Evacuation centres with high inventory cost, e.g. at location (193,948), are established only due to insufficient capacities of the low-inventory-cost centres and the remoteness of the communities served.

By examining the solutions for the cases with $D = 3, 6$ and 9, we can see how the optimal strategy changes as the anticipated level of damage increases. When there are a small number of evacuation centres destroyed (Figure 1.b), the location-allocation is the same as for the case of $D = 0$, since there are enough evacuation centres with low post-disaster inventory-delivery cost, e.g. the centres at (217,168), (321,96), (407,609) and (459,594), to support the communities assigned. When the disaster damage is moderately high (Figure 1.c), the best strategy established some centres with low inventory-delivery cost post-disaster but a higher inventory pre-positioning cost. For example, the centre at (853,20) is replaced by the centre at (935,105). It indicates that, when a significant number of centres are destroyed by the disaster, there is a trade-off between the costs of pre-positioning inventory and of delivering inventory post-disaster.

Another interesting finding is shown in Figure 1.d. When a large number of evacuation centres are expected to be destroyed by the disaster, it is helpful to establish *more evacuation centres* with low post-disaster inventory-delivery costs, even if some of them may not have any inventory pre-positioned, e.g. the centres at (120,179) and (459,594). The number of evacuation centres increases from 12 to 14 in this case. Again, there is also a trade-off between the pre-positioning and post-disaster costs, as illustrated in the replacement for the centre at (978,486) by the centre at (833,572). A solution that does not anticipate post-disaster costs would be much more expensive when a severe disaster strikes.

This computational study shows that the optimal location-allocation strategy that accounts for recovery cost post-disaster may be very different from one that does not consider the costs after a worst-case scenario. Therefore, in disaster preparedness planning, it is very important that post-disaster scenarios are considered when determining the location of evacuation centres.

### 6.4 Case Study

We investigated a case study on hurricane preparedness in southeast USA, based on the study of Rawls and Turnquist (2011). There are 30 cities, each being a possible location for an evacuation centre; see Figure 2. We use the actual distances and populations for our computations. Other parameter values are adapted from Rawls and Turnquist (2011) as follows. The size and fixed cost of the centre considered for each city is pre-selected (with an equal chance) to be either *Small* ($F_i$*=58,800,* $C_i$*=109,200)*, *Medium* ($F_i$*=565,200,* $C_i$*=1,224,600)* or *Large* ($F_i$*=900,000,* $C_i$*=2,340,000)*. For each evacuation centre, $h_i$ is randomly generated from a discrete uniform distribution $U[20, 50]$ with the post-disaster inventory delivery cost $H_i$ being 40 times higher.

We solve the problem for $D = 0$, 3, 6 and 9; the optimal set of centres established and pre-positioned inventory levels are shown in Table 3. Similar to results reported in Section 6.3, when no disaster strikes ($D = 0$), the optimal solution establishes fewer evacuation centres and pre-positions inventory to levels

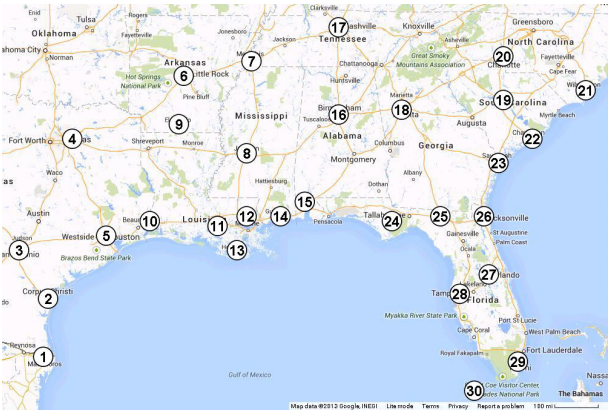



**Fig. 2.** Cities in the case study – Southeast USA. (Map data: Google, INEGI)

close to capacity to reduce fixed costs. When the disaster damage is higher, it is beneficial to establish more centres that are low cost (e.g., centre 23 for $D = 3$, and centres 4 and 23 for $D = 6$). When a large number of evacuation centres are expected to be destroyed by the disaster ($D = 9$), the optimal solution relinquishes some high capacity (but high-cost) centres (e.g., centres 13 and 18) and replaces them by more lower-cost centres. We also note that if the damage is severe so that more evacuation centres are needed, the pre-positioned inventory levels may not fully use the centres' capacities.

**Table 3.** Pre-positioned inventory level (thousand units)

| Centres established | Size | $h_i$ | $D = 0$ | $D = 3$ | D=6 | $D = 9$ |
|---|---|---|---|---|---|---|
| 2. Corpus Christi | Large | 25 | 2105 | 2105 | 1875 | 1875 |
| 3. San Antonio | Large | 20 | 2161 | 2161 | 2161 | 2161 |
| 4. Dallas | Medium | 21 | — | — | 249 | 249 |
| 7. Memphis | Medium | 26 | — | — | — | 1207 |
| 8. Jackson | Large | 29 | — | — | — | 607 |
| 9. El Dorado | Large | 27 | 2306 | 2306 | 2287 | 2034 |
| 13. New Orleans | Medium | 30 | 725 | 725 | 725 | — |
| 18. Atlanta | Medium | 38 | 1110 | 1110 | 1110 | — |
| 19. Columbia | Large | 32 | 2289 | 2289 | 2289 | 1016 |
| 23. Savannah | Small | 22 | — | 20 | 20 | 20 |
| 24. Tallahassee | Medium | 29 | — | — | — | 1121 |
| 26. Jacksonville | Medium | 28 | — | — | — | 1211 |
| 27. Orlando | Medium | 30 | 1218 | 1198 | 1198 | 413 |
| 29. Miami | Small | 27 | 25 | 25 | 25 | 25 |

— indicates that the centre is not established.

We summarize the insights for disaster relief planning from our computational study as follows:

1. When post-disaster cost are neglected when locating evacuation centres, a minimal number of centres are established with high levels of pre-positioned inventory. Such a solution may still be optimal if the disaster damage is mild (i.e., most evacuation centres can still operate and most of the inventory is still usable).
2. When the damage after a disaster becomes more moderate (i.e., a significant number of the evacuation centres cannot function), there is a trade-off between pre-positioning and post-disaster costs. It is beneficial to establish evacuation centres with relatively low costs of both.
3. When the damage after disaster is severe (i.e., a large number of evacuation centres are not operational), it is helpful to establish more spare evacuation centres with low post-disaster cost, which may not even need to carry any pre-positioned inventory. These centres can supply new inventory post-disaster at low cost.

## 7 Summary

This paper proposes a tri-level programming model for disaster preparedness planning that allows for anticipation of the cost of recovery post-disaster at the preparedness stage. We adopt an interdiction framework instead of using a stochastic or chance-constrained formulation to represent the disaster damage; our rationale being that generating the full profile of possible scenarios and their corresponding probabilities may be quite difficult. This approach also allows a sensitivity trade-off analysis between the cost and the anticipated severity of the disaster. We proposed a dual-ascent approach for solving this tri-level programming model. Our preliminary results indicate that the solution approach is efficient enough to be able to evaluate the many scenarios necessary for decision-support for disaster preparedness. We also draw some managerial insights from the computational instances.

**Acknowledgment.** This research was initiated when the last author was a Visiting Professor at Sophia University in 2013. The award of a Sophia Lecturing-Research grant by the Science and Technology Exchange Committee (STEC) of Sophia University is gratefully acknowledged.

## References

1. Balcik, B., Beamon, B.M.: Facility Location in Humanitarian Relief. International Journal of Logistics: Research and Applications 11(2), 101–121 (2008)
2. Balcik, B., Beamon, B.M., Krejci, C.C., Muramatsu, K.M., Ramirez, M.: Coordination in Humanitarian Relief Chains: Practices, Challenges and Opportunities. International Journal of Production Economics 126, 22–34 (2010)
3. Bozorgi-Amiri, A., Jabalameli, M.S., Alinaghian, M., Heydari, M.: A Modified Particle Swarm Optimization for Disaster Relief Logistics under Uncertain Environment. International Journal of Advanced Manufacturing Technologies 60, 357–371 (2012)

4. Celik, M., Ergun, Ö., Johnson, B., Keskinocak, P., Lorea, A., Pekgün, Swann, J.: Humanitarian Logistics. Tutorials in Operations Research, Informs 2012 (2012)
5. Doerner, K.F., Gutjahr, W.J., Nolz, P.C.: Multi-criteria location planning for public facilities in tsunami-prone costal areas. OR Spektrum 31, 651–678 (2009)
6. Döyen, A., Necati, A., Barbarosoglu, G.: A Two-echelon Stochastic Facility Location Model for Humanitarian Relief Logistics. Optimization Letters 6, 1123–1145 (2012)
7. Huang, M., Smilowitz, K., Blacik, B.: Models for Relief Routing: Equity, Efficiency and Efficacy. Transportation Research Part E 48, 2–18 (2012)
8. Jia, H., Ordez, F., Dessouky, M.: A modeling framework for facility location of medical services for large-scale emergencies. IIE Transactions 39(1), 41–55 (2007)
9. Kuehn, A.A., Hamburger, M.J.: A Heuristic Program for Locating Warehouses. Management Science 9(4), 643–666 (1963)
10. Lu, Q., George, B., Shekhar, S.: Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. In: Bauzer Medeiros, C., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 291–307. Springer, Heidelberg (2005)
11. Mete, H.O., Zabinsky, Z.B.: Stochastic Optimization of Medical Supply Location and Distribution in Disaster Management. International Journal of Production Economics 126, 76–84 (2010)
12. Nemhauser, G.L., Woolsey, L.A.: Integer and Combinatorial Optimization. John Wiley & Sons, New York (1999)
13. Noyan, N.: Risk-averse Two-stage Stochastic Programming with an Application to Disaster Management. Computers and Operations Research 39, 541–559 (2012)
14. Nurre, S.G., Cavdaroglu, B., Mitchel, J.E., Sharkey, T.C., Wallace, W.A.: Restoring Infrastructure Systems: An Integrated Network Design and Scheduling (INDS) Problem. European Journal of Operational Research 223, 794–806 (2012)
15. O'Hanley, J.R., Church, R.L.: Designing Robust Coverage Networks to Hedge against Worst-case Facility Losses. European Journal of Operational Research 209, 23–36 (2011)
16. Rawls, C.G., Turnquist, M.A.: Pre-positioning of Emergency Supplies for Disaster Response. Transportation Research Part B 44, 521–534 (2010)
17. Rawls, C.G., Turnquist, M.A.: Pre-positioning Planning for Emergency Response with Service Quality Constraints. IT OR Spectrum 33, 481–498 (2011)
18. Rawls, C.G., Turnquist, M.A.: Pre-positioning and Dynamic Delivery Planning for Short-term Response Following a Natural Disaster. Socio-Economic Planning Sciences 46, 46–54 (2012)
19. Salmerón, J., Apte, A.: Stochastic Optimization for Natural Disaster Asset Prepositioning. Production and Operations Management 19(5), 561–574 (2010)
20. Sherali, H.D., Carter, T.B., Hobeika, A.G.: A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. Transportation Research Part B 25(6), 439–452 (1991)
21. Tomasini, R.M., van Wassenhove, L.N.: From Pre-paredness to Partnerships: Case Study Research on Humanitarian Logistics. International Transactions on Operational Research 16, 549–559 (2009)
22. World Bank: The Great East Japan Earthquake: Learning from Megadisasters. The World Bank, Washington DC, USA (2012)
23. Yushimito, W.F., Ukkusuri, S.V.: A Location-routing Approach for the Humanitarian Pre-positioning Problem. Transportation Research Record: Journal of the Transportation Research Board 2089, 18–25 (2008)

# Throughput Evaluation of Buffered Unreliable Production Lines with Machines Having Multiple Failure Modes

Yassine Ouazene, Alice Yalaoui, Hicham Chehade, and Farouk Yalaoui

Laboratoire d'Optimisation des Systèmes Industriels (UMR-STMR-CNRS 6279)
Université de Technologie de Troyes, 12 rue Marie Curie, CS 42060
10004 TROYES, France
{yassine.ouazene,alice.yalaoui,hicham.chehade,farouk.yalaoui}@utt.fr
http://www.utt.fr

**Abstract.** This paper presents an analytical method to evaluate the system throughput of buffered serial production lines. The machines are unreliable and each machine can have more than one failure and repair modes. The flow of parts through the production line is considered as a continuous flow of material. The proposed method is composed of two main steps. The first one consists of transforming each machine into a single failure and repair mode machine based on a well-known aggregation technique. In the second step, the system throughput is evaluated using an approximate approach called Equivalent Machine Method. This method is based on the analysis of the different states of each buffer using birth-death Markov processes. For each buffer, the birth and death ratios are defined taking into account the probabilities of blockage and starvation of its related upstream and downstream machines. Then, each original machine is defined by its equivalent production rate considering its processing rate, its availability and the influence of the buffers limitations and the other machines. The system throughput is defined as the bottleneck between the equivalent production rates. This method considers both homogeneous and non-homogeneous production lines.

A comparative study based on different test instances is presented and discussed. The obtained results prove the effectiveness and the accuracy of the proposed method comparing with both simulation and aggregation methods.

**Keywords:** Manufacturing Systems, Performance Evaluation, Throughput Analysis, Equivalent Machine Method, Multiple Failure Modes.

## 1 Introduction

An accurate estimation of a production or transfer line performance is necessary for the optimization of its design and utilization. A system performance can be expressed using many measures such as: machines availability, work in process and throughput. This last performance measure can be defined as the number

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 229–241, 2013.

of parts produced by the last machine of a manufacturing system over a given period of time [1]. Throughput can be defined also in the steady state of the system as the average number of parts produced per time unit.

Due to this crucial importance, analytical methods to investigate the performance of production lines have been widely studied in the literature. An interesting literature review which summarizes the recent studies devoted to developing these analytical methods has been parented by Li et al. [10]. An important conclusion of this study is that most of these analytical methods are based on aggregation or decomposition techniques. The first technique consists of replacing a two-machine line by a single equivalent machine that has the same throughput in isolation. Then, the obtained machine is combined with the next machine to obtain a new aggregated machine. This technique can be done in a forward as well as in a backward direction. One of its earliest applications is that of Buzacott [4], who applied this method to a fixed-cycle three-stage production line. De Koster [7] tested the aggregation approach to estimate the efficiency of an unreliable line with exponential failure and repair times. The second technique consists of decomposing the longer production line into a set of two-machine lines when the analysis of these subsystems is available [10]. The set of two-machine lines is assumed to have equivalent behavior to the initial production line. This method was proposed for the analysis of discrete models of long homogeneous lines by Gershwin [8]. The computational efficiency of this method was later improved by the introduction of the so-called DDX algorithm [5,6].

Recently, Ouazene et al. [12] have proposed an equivalent machine method based on the analysis of a serial production line using a combination of birth-death Markov process to analyze the behavior of the different buffers in theirs steady states. This approach allows the analysis of the serial production line without decomposing or aggregating this line into smaller sub-lines.

Most of these methods deal with unreliable production lines considering single failure-machines. Or, in many practical cases, machines can be subject to different kinds of failures which occur with different frequencies and require different amounts of time to be repaired [3]. Levantesi [9] proposed an analytical method able to deal both with non-homogeneous lines and multiple failure modes for each machine considering a discrete flow of parts. Their method is based on the decomposition method introduced by Gershwin [8] which is an approximate evaluation of $K$-machine line based on the evaluation of $K-1$ buffered two-machine sub-lines. This method is also an extension of the decomposition technique based on a multiple failure approach previously proposed by Tolio et al. [15] for the analysis of exponential lines. Another method that considers performance evaluation of a tandem homogeneous production line with machines having multiple failure modes has been studied by Belmansour and Nourelfath [2,3]. This approach consists of an aggregation technique. The aggregation algorithm is a recursive approach that replaces each dipole (two-machine on buffer model) by a single machine. Based on a comparative study with the simple aggregation

approach, the authors have established that the performance evaluation is more accurate by distinguishing among different failure modes of each machine.

The remainder of this paper is organized as follows: Section 2 introduces the problem studied with the different assumptions and notations. This is followed by describing a transformation. In Section 4, we detail the equivalent machine method to evaluate the system throughput. Several numerical experiments are presented and discussed in Section 5. Finally, Section 6 summarizes the guidelines of our contribution.

## 2 Problem Description and Assumptions

The production line considered in this paper consists of $K$ unreliable machines or stations separated by $(K-1)$ intermediate buffers (see Figure 1). The products flow continuously in a fixed sequence from the first machine to the last one. It is assumed that there are always available products at the input of the system and unlimited space for material storage at the output of the last machine. In other words, the first machine cannot be starved and the last one cannot be blocked. The failures are operation-dependent, a machine can not fail while it is starved, blocked or idle. All times to failure and times to repair are independent and exponentially distributed. If no products are available in the upstream buffer $B_{j-1}$ the machine $M_i$ will be starved and when the downstream buffer $B_j$ is full, it will be blocked.

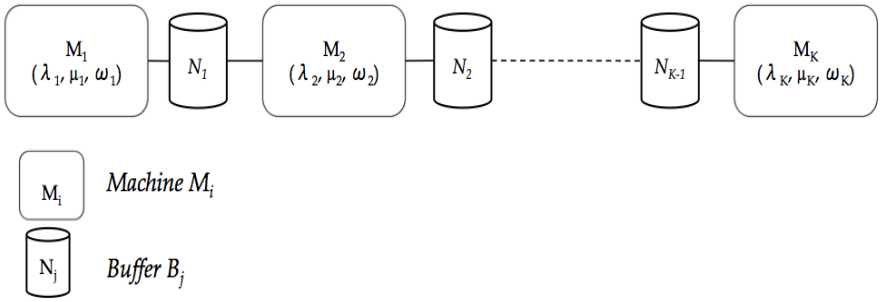



**Fig. 1.** A $K$-machine $(K-1)$-buffer serial production line

Each machine $M_i$ has $F_i$ failure modes. It can be operational, starved, blocked or failed in a certain mode. The processing times of the different machines are not necessarily identical. $\lambda_{im}$ and $\mu_{im}$ are, respectively, the failure and repair rate of machine $M_i$ in mode $m$. We assume also that an operational machine can fail in only one of its failure modes.

The different notations utilized in this paper are defined in Table 1.

**Table 1.** Notation

| | |
|---|---|
| $\lambda_i$ | Equivalent failure rate of the machine $M_i$ |
| $\mu_i$ | Equivalent repair rate of the machine $M_i$ |
| $\omega_i$ | Processing rate of the machine $M_i$ |
| $N_j, j \in \{1...K-1\}$ | Capacity of the intermediate buffer $B_j$ |
| $\xi_i, i \in \{1...K-1\}$ | Availability of the intermediate buffer $B_j$ regarding machine $M_i$ |
| $\alpha_j, j \in \{1...K-1\}$ | Processing rates fraction related to the buffer $B_j$ |
| $\rho_i, i \in \{1...K\}$ | Equivalent production rate of the machine $M_i$ |
| $P_j^s, s \in \{0...N_j\}$ | Steady probability to have $s$ products in the buffer $B_j$ |
| $\psi$ | Production line throughput |

## 3 Single-failure Mode Transformation

In this section, we present the transformation proposed by Belmansour and Nourelfath [2,3]. Each machine $M_i$ can be defined as a single failure mode machine with an equivalent failure rate and an equivalent repair rate.

The equivalent failure rate is defined as the sum of the different rates of the different failure modes (see Equation (1)).

$$\forall i = 1...K, \lambda_i = \sum_{m=1}^{F_i} \lambda_{im} \tag{1}$$

Based on the assumption that the equivalent machine should have the same availability, the equivalent repair rate is given by Equation (2).

$$\forall i = 1...K, \mu_i = \frac{1}{\sum_{m=1}^{F_i} \frac{\lambda_i}{\lambda_{im}} \times \frac{1}{\mu_{im}}} \tag{2}$$

**Case of a machine with two failure modes:**

If we consider a machine $M_i$ with two failure modes represented by $\lambda_{i1}, \lambda_{i2}$ and $\mu_{i1}, \mu_{i2}$, this machine corresponds to a single failure mode machine with failure $\lambda_i$ and repair rates $\mu_i$ such as both machines have the same availability as shown in Equation (3).

$$\frac{1}{1+\frac{\lambda_i}{\mu_i}} = \frac{1}{1+\frac{\lambda_{i1}}{\mu_{i1}}+\frac{\lambda_{i2}}{\mu_{i2}}} \tag{3}$$

Based on Equation (3) we obtain:

$$\mu_i = \frac{1}{(\frac{\lambda_{i1}}{\lambda_{i1}+\lambda_{i2}} \times \frac{1}{\mu_{i1}}) + (\frac{\lambda_{i2}}{\lambda_{i1}+\lambda_{i2}} \times \frac{1}{\mu_{i2}})} \tag{4}$$

Or, if we consider that $\lambda_i = \lambda_{i1} + \lambda_{i2}$, the equation above can be re-written as follows.

$$\mu_i = \frac{1}{(\frac{\lambda_{i1}}{\lambda_i} \times \frac{1}{\mu_{i1}}) + (\frac{\lambda_{i2}}{\lambda_i} \times \frac{1}{\mu_{i2}})} \tag{5}$$

These relationships can be easily established by considering the machine with the two failure modes as two serial machines with single failure mode for each machine.

If we take into account the assumption that the two failure modes cannot occur simultaneously the equivalent failure rate will be the following.

$$\lambda_i = \lambda_{i1} + \lambda_{i2} - \lambda_{i1} \times \lambda_{i2} \tag{6}$$

The equivalent repair rate will be changed consequently.

$$\mu_i = \frac{1}{(\frac{\lambda_{i1}}{\lambda_{i1}+\lambda_{i2}-\lambda_{i1}\times\lambda_{i2}} \times \frac{1}{\mu_{i1}}) + (\frac{\lambda_{i2}}{\lambda_{i1}+\lambda_{i2}-\lambda_{i1}\times\lambda_{i2}} \times \frac{1}{\mu_{i2}})} \tag{7}$$

## 4 Equivalent Machine Method Formulation

The main idea of the proposed approach is to replace each machine by an equivalent one that has only up and down states. The blockage and starvation are integrated in the up state of the machine. This formulation is a generalization of two-machine-one-buffer production line proposed by Ouazene et al. [13,11]. In this paper, we adapt the equivalent machine method proposed by the authors [12] to deal with discrete flow of material to the continuous case.

Based on the analysis of the buffers steady states using birth-death Markov processes, we determine the probabilities of starvation and blockage of each buffer. Then, these probabilities are used to model and analyze the system behavior in its steady state.

### 4.1 Two-machine-one-buffer Model

Before detailing the general formulation, we introduce the simple system which consists of two machines separated by one buffer. This model is used as a building block to construct the general model. To analyze the steady states of the buffer, we consider a birth-death Markov process with $(N+1)$ states $\{0, 1...N\}$ such as $N$ is the capacity of the intermediate buffer and $\omega_1$ and $\omega_2$ are, respectively, the birth and death transition rates.

The differential equations for the probability that the system is in state $j$ at time $t$ are:

$$\begin{cases} \frac{\partial P_0(t)}{\partial t} = -\omega_1 \times P_0(t) + \omega_2 \times P_1(t) \\ \frac{\partial P_j(t)}{\partial t} = \omega_1 \times P_{j-1}(t) - (\omega_1 + \omega_2) \times P_j(t) + \omega_2 \times P_{j+1}(t) \\ \frac{\partial P_N(t)}{\partial t} = \omega_1 \times P_{N-1}(t) + \omega_2 \times P_N(t) \end{cases} \tag{8}$$

In the steady state of the system, all the differential terms are equal to zero (see Equation (9)).

$$\begin{cases} 0 = -\omega_1 \times P_0 + \omega_2 \times P_1 \\ 0 = \omega_1 \times P_{j-1} - (\omega_1 + \omega_2) \times P_j + \omega_2 \times P_{j+1} \\ 0 = \omega_1 \times P_{N-1} + \omega_2 \times P_N \end{cases} \tag{9}$$

So, by simplifying the system above and considering the normalization equation: $\sum_{j=0}^{N} P_j = 1$ we obtain the steady probabilities of each buffer state (10).

$$P_j = \begin{cases} \frac{\alpha^j \times (1-\alpha)}{1-\alpha^{N+1}} \text{ if } \alpha \neq 1 \\ \frac{1}{N+1} \quad \text{if } \alpha = 1 \end{cases} \tag{10}$$

We are especially interested in the starvation and blockage probabilities, respectively, represented by empty and full buffer states given by the following equations:

$$P_0 = \begin{cases} \frac{1-\alpha}{1-\alpha^{N+1}} \text{ if } \alpha \neq 1 \\ \frac{1}{N+1} \quad \text{if } \alpha = 1 \end{cases} \tag{11}$$

$$P_N = \begin{cases} \frac{\alpha^N \times (1-\alpha)}{1-\alpha^{N+1}} \text{ if } \alpha \neq 1 \\ \frac{1}{N+1} \quad \text{if } \alpha = 1 \end{cases} \tag{12}$$

Based on these two probabilities, the effective production rate of each work station is defined as function of machine processing rate, machine and the buffer availabilities (13).

$$\rho_i = \omega_i \times \frac{\mu_i \times \xi_i}{\mu_i + \xi_i \times \lambda_i} \tag{13}$$

Such as: $\xi_1 = 1 - P_N$ and $\xi_2 = 1 - P_0$.
The system throughput $\psi$ is defined as the bottleneck between the two effective production rates $\rho_1$ and $\rho_2$.

$$\psi = min \left\{ \omega_1 \times \frac{\mu_1 \times (1 - P_N)}{\mu_1 + \lambda_1 \times (1 - P_N)}, \omega_2 \times \frac{\mu_2 \times (1 - P_0)}{\mu_2 + \lambda_2 \times (1 - P_0)} \right\} \tag{14}$$

After some transformations, the system throughput can be re-written as follows:

$$\psi = \begin{cases} \omega_1 \times \min\{\frac{\mu_1 \times (1-\alpha^N)}{\mu_1 \times (1-\alpha^{N+1}) + \lambda_1 \times (1-\alpha^N)}; \\ \frac{\mu_2 \times (1-\alpha^N)}{\mu_2 \times (1-\alpha^{N+1}) + \lambda_2 \times \alpha \times (1-\alpha^N)}\} \\ \text{if } \alpha \neq 1 \\ \omega \times \min\{\frac{N \times \mu_1}{N \times \lambda_1 + (N+1) \times \mu_1}, \frac{N \times \mu_2}{N \times \lambda_2 + (N+1) \times \mu_2}\} \\ \text{if } \alpha = 1 \end{cases} \tag{15}$$

More implicitly:

$$\psi = \begin{cases} \omega_1 \times \min\{\frac{1}{\frac{1-\alpha^{N+1}}{1-\alpha^N}+\frac{\lambda_1}{\mu_1}}; \frac{1}{\frac{1-\alpha^{N+1}}{1-\alpha^N}+\alpha\times\frac{\lambda_2}{\mu_2}}\} \\ \text{if } \alpha \neq 1 \\ \omega \times \min\{\frac{1}{\frac{N}{N+1}+\frac{\lambda_1}{\mu_1}}; \frac{1}{\frac{N}{N+1}+\frac{\lambda_2}{\mu_2}}\} \\ \text{if } \alpha = 1 \end{cases} \tag{16}$$

### 4.2 General Model with $K$ Machines and $(K-1)$ Buffers

The two-machine-one-buffer presented above is used as a building block to analyse larger production lines. Therefore, the states of each intermediate buffer $B_j$ are analyzed using a dedicated birth-death Markov process. These Markov processes differ in terms of number of states because the buffers are not identical. They also differ in terms of birth and death transition rates because each buffer $B_j$ is differently influenced by the machines and the other buffers (see Figure 2).

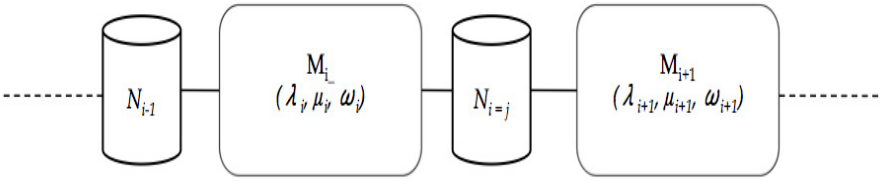



**Fig. 2.** Sub-system of the original production line

In the simple case of two machines and one buffer, each available machine processes $\omega_i$ products per time unit. For this reason, $\omega_1$ and $\omega_2$ are, respectively, the birth and death transition rates. But in the general case, the machines $M_i$ and $M_{i+1}$ related to the buffer $B_j$ are subject to starvation and blockage. So their effective processing rates are affected by the availabilities of the buffers $B_{j-1}$ and $B_{j+1}$. The upstream machine $M_i$ can process products if the buffer $B_{i-1}$ is not empty and the downstream machine $M_{i+1}$ can process products when the buffer $B_{j+1}$ is not full.

The first buffer and the last one should be considered as particular cases because the first machine cannot be starved and the last machine cannot be blocked. The birth-death Markov process, related to the buffer $B_j$, is represented in Figure 3.

The different states of the $(K-1)$ buffers are modeled by $(K-1)$ related but different birth-death Markov processes. Each stochastic process is defined by its processing rates ratio $\alpha_j$. The different ratios $\alpha_j$ are defined by the following equations:

$$\alpha_1 = \frac{\omega_1 \times (1 - P_{N_2}^2)}{\omega_2} \,. \tag{17}$$

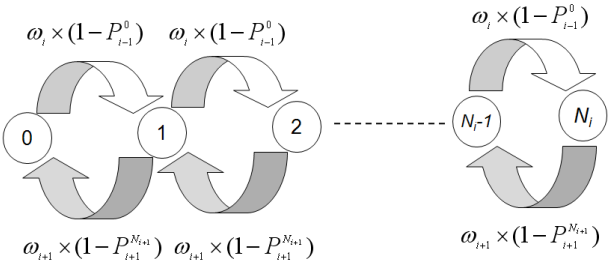



**Fig. 3.** Birth-death Markov process related to the buffer $B_j$

$$\alpha_{K-1} = \frac{\omega_{K-1} \times (1 - P_0^{K-1})}{\omega_K} \,. \tag{18}$$

$$\forall j = 2...K-2, \alpha_j = \frac{\omega_i \times (1 - P_0^{j-1})}{\omega_{i+1} \times (1 - P_{N_{j+1}}^{j+1})} \,. \tag{19}$$

Equations (17) and (18) consider, respectively, the particular case of the first and the last buffer because the first machine cannot be starved and the last one cannot be blocked.

Based on the analysis presented in Section 2, the probabilities of empty and full states of each buffer are calculated using Equations (20) and (21).

$$P_0^j = \begin{cases} \frac{1-\alpha_j}{1-\alpha_j^{N_j+1}} & \text{if } \alpha_j \neq 1 \\ \\ \frac{1}{N_j+1} & \text{if } \alpha_j = 1 \end{cases} \tag{20}$$

$$P_{N_i}^j = \begin{cases} \frac{\alpha_j^{N_j} \times (1-\alpha_j)}{1-\alpha_j^{N_j+1}} & \text{if } \alpha_j \neq 1 \\ \\ \frac{1}{N_j+1} & \text{if } \alpha_j = 1 \end{cases} \tag{21}$$

The resolution of Equations (17) to (21) allows the determination of the processing rates ratio $\alpha_j$ and empty and full states probabilities $(P_0^j, P_{N_j}^j)$ of each buffer. So, based on this information and the two-machine-one-buffer model presented above, we can calculate the effective production rate of each machine considering the influence of the buffers availabilities using Equation (22).

$$\forall i = 1...K, \rho_i = \omega_i \times \frac{\mu_i \times \xi_j}{\mu_i + \xi_j \times \lambda_i} \tag{22}$$

Such as:

$$\begin{cases} \xi_1 = 1 - P_{N_1}^1 \\ \xi_K = 1 - P_0^{K-1} \\ \forall j = 2...K-1, \xi_j = (1 - P_0^{j-1}) \times (1 - P_{N_{j+1}}^{j+1}) \end{cases} \tag{23}$$

Similarly to the two-machine-one-buffer model, the throughput of the production line $\psi$ is defined as the bottleneck between the effective production rates of all machines:

$$\psi = \min\{\omega_i \times \frac{\mu_i \times \xi_i}{\mu_i + \xi_i \times \lambda_i}\}, i = 1...K \ . \tag{24}$$

The equivalent machine method proposed in this paper to evaluate the system throughput of a buffered serial production line is summarized by the non-linear programming algorithm given below (see Algorithm 1). This algorithm can be solved by LINGO software.

**Algorithm 1.** Aggregated Equivalent Machine Method

| | | |
|---|---|---|
| | $K$ | Number of machines |
| | $K-1$ | Number of buffers |
| | $M_i$ | Number of failure modes of machine $M_i$ |
| **Require:** | $\lambda_{im}$ | Failure rate of machine $M_i$ in mode $m$ |
| | $\mu_{im}$ | Repair rate of machine $M_i$ in mode $m$ |
| | $\omega_i$ | Processing rate of machine $M_i$ |
| | $N_j$ | Capacity of buffer $B_j$ |

**for all** each machine $M_i$ **do**
  $\lambda_i \leftarrow \sum_{m=1}^{M(i)} \lambda_{im}$
  $\mu_i \leftarrow \frac{1}{\sum_{m=1}^{M(i)} \frac{\lambda_i}{\lambda_{im}} \times \frac{1}{\mu_{im}}}$
**end for**
**for all** each buffer $B_j$ **do**
  **for all** each machine $M_i$ **do**
    $P_j^0 \leftarrow$ the steady probability that the buffer $j$ is empty
    $P_j^{N_j} \leftarrow$ the steady probability that the buffer $j$ is full
    $\alpha_j \leftarrow$ the processing rate ratio related to the buffer $j$
    $\rho_i \leftarrow$ the equivalent throughput of the machine $i$
  **end for**
**end for**
**return** $\psi = \min\{\rho_i; i = 1...K\}$

## 5 Numerical Results

The proposed method has been validated by a comparative study based on both a simulation and an aggregation approach proposed by Belmansour and Nourelfath

[2,3]. Different production line configurations are considered with five, ten and fifteen machines, respectively. Each machine in these examples has two failure modes. The different machine's parameters (failure, repair and processing rates) of each example are reported in Tables 2, 3 and 4. The comparison, for each example, is made up by calculating the relative difference between the system throughput obtained using the analytical approach with the simulation results based on Equation (25).

$$gap(\%) = 100 \times |\frac{\psi_{Analyticalmodel} - \psi_{Simulation}}{\psi_{Simulation}}| \,. \tag{25}$$

**Table 2.** Data for the test example 1 with five machines [2]

| $M_i$ | $\lambda_{i1}$ | $\mu_{i1}$ | $\lambda_{i2}$ | $\mu_{i2}$ | $\omega_i$ | $N_i$ |
|---|---|---|---|---|---|---|
| Machine 1 | 0.0120 | 0.2200 | 0.0050 | 0.0400 | 1.1000 | 55 |
| Machine 2 | 0.0100 | 0.0400 | 0.0800 | 0.1500 | 1.1000 | 40 |
| Machine 3 | 0.1000 | 0.2000 | 0.0400 | 0.0900 | 1.1000 | 40 |
| Machine 4 | 0.0100 | 0.0870 | 0.1160 | 0.2971 | 1.1000 | 55 |
| Machine 5 | 0.1000 | 0.2500 | 0.0500 | 0.0800 | 1.1000 | - |

**Table 3.** Data for the test example 2 with ten machines [2]

| $M_i$ | $\lambda_{i1}$ | $\mu_{i1}$ | $\lambda_{i2}$ | $\mu_{i2}$ | $\omega_i$ | $N_i$ |
|---|---|---|---|---|---|---|
| Machine 1 | 0.0120 | 0.2200 | 0.0050 | 0.0400 | 1.1000 | 55 |
| Machine 2 | 0.0100 | 0.0400 | 0.0800 | 0.1500 | 1.1000 | 40 |
| Machine 3 | 0.1000 | 0.2000 | 0.0400 | 0.0900 | 1.1000 | 40 |
| Machine 4 | 0.0100 | 0.0870 | 0.1160 | 0.2971 | 1.1000 | 55 |
| Machine 5 | 0.1000 | 0.2500 | 0.0500 | 0.0800 | 1.1000 | 50 |
| Machine 6 | 0.0141 | 0.2323 | 0.0059 | 0.0422 | 1.1000 | 50 |
| Machine 7 | 0.0011 | 0.0348 | 0.0089 | 0.1306 | 1.1000 | 65 |
| Machine 8 | 0.0143 | 0.1350 | 0.0057 | 0.0607 | 1.1000 | 55 |
| Machine 9 | 0.0008 | 0.0349 | 0.0092 | 0.1192 | 1.1000 | 35 |
| Machine 10 | 0.0133 | 0.1709 | 0.0067 | 0.0546 | 1.1000 | - |

The obtained results for the three examples are reported in Table 5. Note that the proposed approach outperforms the aggregation methods already developed in the literature. In fact, the largest absolute error of the proposed method is equal to 2.96% while for the aggregation method it is 4.46%.

Based on these numerical results, we can state that the proposed method has been proven to be efficient and accurate to evaluate the system throughput of a serial production line with machines having different failure modes.

**Table 4.** Data for the test example 3 with fifteen machines [2]

| $M_i$ | $\lambda_{i1}$ | $\mu_{i1}$ | $\lambda_{i2}$ | $\mu_{i2}$ | $\omega_i$ | $N_i$ |
|---|---|---|---|---|---|---|
| Machine 1 | 0.0120 | 0.2200 | 0.0050 | 0.0400 | 1.3000 | 55 |
| Machine 2 | 0.0100 | 0.0400 | 0.0800 | 0.1500 | 1.3000 | 40 |
| Machine 3 | 0.1000 | 0.2000 | 0.0400 | 0.0900 | 1.3000 | 40 |
| Machine 4 | 0.0100 | 0.0870 | 0.1160 | 0.2971 | 1.3000 | 55 |
| Machine 5 | 0.1000 | 0.2500 | 0.0500 | 0.0800 | 1.3000 | 50 |
| Machine 6 | 0.0141 | 0.2323 | 0.0059 | 0.0422 | 1.3000 | 50 |
| Machine 7 | 0.0011 | 0.0348 | 0.0089 | 0.1306 | 1.3000 | 65 |
| Machine 8 | 0.0143 | 0.1350 | 0.0057 | 0.0607 | 1.3000 | 55 |
| Machine 9 | 0.0008 | 0.0349 | 0.0092 | 0.1192 | 1.3000 | 35 |
| Machine 10 | 0.0133 | 0.1709 | 0.0067 | 0.0546 | 1.3000 | 40 |
| Machine 11 | 0.0296 | 0.2323 | 0.0124 | 0.0422 | 1.3000 | 55 |
| Machine 12 | 0.0074 | 0.0696 | 0.0596 | 0.2611 | 1.3000 | 40 |
| Machine 13 | 0.0429 | 0.2024 | 0.0171 | 0.0910 | 1.3000 | 50 |
| Machine 14 | 0.0037 | 0.0698 | 0.0423 | 0.2383 | 1.3000 | 55 |
| Machine 15 | 0.0080 | 0.4273 | 0.0040 | 0.1366 | 1.3000 | - |

**Table 5.** Numerical results for the test examples

| Method | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| Simulation | 0.56508 | 0.52239 | 0.60723 |
| Existing aggregation method | 0.52748 | 0.48617 | 0.56691 |
| ***Gap 1 (%)*** | **-6.65%** | **-6.93%** | **-6,64%** |
| Aggregation method proposed by Belmansour and Nourelfath [3] | 0.54350 | 0.49911 | 0.58498 |
| ***Gap 2 (%)*** | **-3.82%** | **-4.46%** | **-3.66%** |
| Proposed method | 0.56838 | 0.53259 | 0.62519 |
| ***Gap 3 (%)*** | **0.58%** | **1.95%** | **2.96%** |

## 6 Conclusion

This paper addresses a new method to evaluate the system throughput of a buffered production line with machines having multiple failure modes. This method is based on the transformation of each machine to a single-failure machine. Then, the system throughput is evaluated based on the equivalent machine method adapted to the context of transfer lines. The approach proposed in the paper has been coded in an algorithm that has proven to be fast and accurate. A comparative study based on both simulation experiments and existing aggregation methods has shown the accuracy of the proposed approach.

The originality of this method is its reduced number of variables because it considers only the empty and full states of each buffer and the processing rates ratio related to each buffer. Therefore, to evaluate the throughput of a $K$-machine $(K-1)$-buffer production line we have to solve $(4K-3)$ equations.

The computational complexity is less important than other approaches based on the generation of buffer states transition probabilities matrix. In other words, the computational complexity of the proposed method in this paper is independent from the buffers' sizes.

Future extension of this work may be the adaptation of this approach to the analysis of more complex production systems such as series-parallel structures or split and merge systems. Ouazene et al. [14] have established the efficiency of the equivalent machine method in the evaluation of the performance of a buffered merge production system which consists of two parallel machines on the first stage and one machine on the second one.

**Acknowledgments.** The authors would like to thank the editors and anonymous reviewers for their valuable remarks, comments and suggestions that helped to improve this paper.

## References

1. Ambani, S.M.: Analytical Estimation of Throughput Distribution for Serial Manufacturing Systems with Multi-state Machines and its Application. Phd-Thesis Mechanical Engineering, University of Michigan, USA (2011)
2. Belmansour, A.T., Nourelfath, M.: An aggregation method for performance evaluation of a tandem homogenous production line with machines having multiple failure modes. CIRRELT-2008-53 (2008)
3. Belmansour, A.T., Nourelfath, M.: An aggregation method for performance evaluation of a tandem homogenous production line with machines having multiple failure modes. Reliability Engineering & System Safety 95(11), 1193–1201 (2010)
4. Buzacott, J.A.: Automatic transfer line with buffer stocks. International Journal of Production Research 5(3), 183–200 (1967)
5. Dallery, Y., David, R., Xie, X.L.: An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. IIE Transactions 20(3), 280–283 (1988)
6. Dallery, Y., David, R., Xie, X.L.: Approximate analysis of transfer lines with unreliable machines and finite buffers. IEEE Transactions on Automatic Control 34(9), 943–953 (1989)
7. De Koster, M.B.M.: Estimation of line efficiency by aggregation. International Journal of Production Research 25(4), 615–625 (1987)
8. Gershwin, S.B.: An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. Operations Research 35(2), 291–305 (1987)
9. Levantesi, R., Matta, A., Tolio, T.: Performance evaluation of continuous production lines with machines having different processing times and multiple failure modes. Performance Evaluation 51(2-4), 247–268 (2003)
10. Li, J., Blumenfeld, D.E., Huang, N., Alden, J.M.: Throughput analysis of production systems: recent advances and future topics. International Journal of Production Research 47(14), 3823–3851 (2009)
11. Ouazene, Y., Chehade, H., Yalaoui, A.: New restricted enumeration method for production line design optimization. In: INCOM 2012, 14th IFAC Symposium on Information Control Problems in Manufacturing, pp. 1347–1352. Elsevier Ltd. (2012) ISBN 978-3-902661-98-2

12. Ouazene, Y., Chehade, H., Yalaoui, A., Yalaoui, F.: Equivalent machine method for approximate evaluation of buffered unreliable production lines. In: 2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS), Singapore, pp. 33–39 (April 2013)
13. Ouazene, Y., Yalaoui, A., Chehade, H.: Throughput analysis of two-machine-one buffer line model: a comparative study. In: Proceeding of the 10th International FLINS Conference on Uncertainty Modeling in Knowledge Engineering and Decision Making, pp. 1281–1286. World Scientific (2012) ISBN 978-981-4417-73-0
14. Ouazene, Y., Yalaoui, A., Chehade, H., Yalaoui, F.: A buffered three-station merge system: performance analysis and comparative study. In: 2013 International Conference on Control, Decision and Information Technologies, Hammamet (May 2013)
15. Tolio, T., Matta, A., Jovane, F.: A method for performance evaluation of automated flow lines. CIRP Annals-Manufacturing Technology 47(1), 373–376 (1998)

# Towards Real-Time Data Acquisition for Simulation of Logistics Service Systems

Stefan Mutke[1], Martin Roth[1], André Ludwig[1], and Bogdan Franczyk[1,2]

[1] University of Leipzig, Information Systems Institute, Grimmaische Straße 12, 04109 Leipzig, Germany
{mutke,roth,ludwig,franczyk}@wifa.uni-leipzig.de
[2] University of Economics, Institute for Business Informatics, Komandorska 118/120, 53-345 Wrocław, Poland
{bogdan.franczyk}@ue.wroc.pl

**Abstract.** Driven by rising competition pressure companies began to outsource at least parts of their logistics functions to specialized logistics providers in order to concentrate on the core competences. Hence, new business models emerged like the fourth party logistics provider who acts like a coordinator of arising logistics networks. One of the main tasks of the provider is the planning of such logistics networks, which have a very collaborative and dynamic character. In this paper an efficient way to integrate process modeling and simulation as part of the planning phase is introduced. Furthermore, an integrated approach is introduced for supporting the planning by a better data acquisition in order to provide reliable results at an affordable effort using simulation techniques. Therefore, complex event processing is used to gather real-time data and provides the data as service profiles for simulation.

**Keywords:** Fourth Party Logistics Provider, Simulation, CEP, Data Acquisition.

## 1 Introduction

Logistics is defined as the management of the flow of goods and information between point of origin and point of destination meeting the requirements of shippers and recipients. The main objective of logistics is the delivery of the right goods, at the right point of time, to the right place, with the right quality and quantity and to the right costs (6Rs, [1] and see [2, 3] for a general overview).

In recent years, the logistics industry remarkably changed such that the planning and monitoring of logistics functions is no longer a task performed by customers of logistics providers (e.g. vendors, manufacturers) but by a number of so-called value-added logistics or fourth party logistics service providers (4PL) [4, 5]. Outsourced logistics functions encompass basic services such as transportation, handling and storage of goods but also services like packaging, finishing or clearing of goods. Due to specific requirements of each company (amount of demanded services or integration level) and due to industry-specific requirements (security or tracking issues) each requested logistics service is individual in scope. Thus, value-added

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 242–256, 2013.

logistics service providers need to provide highly individual service offerings to their clients. Within a network of affiliated logistics providers a 4PL selects matching providers to the needed services and integrates them to meet customer's requirements. Moreover, a 4PL governs the overall process, optimize it, and acts as prime contractor for the customer. They are the main contact person, coordinators of the involved logistics providers, and have the responsibility for the overall process and its quality of service. To determine the quality of service, the 4PL has to monitor and to measure the performance of each participating partner. Fig. 1 illustrates a 4PL's service lifecycle which consists of the phases: analysis, design, implementation, operation and retirement [6]. The lifecycle is outlined in more detail in Section 6.

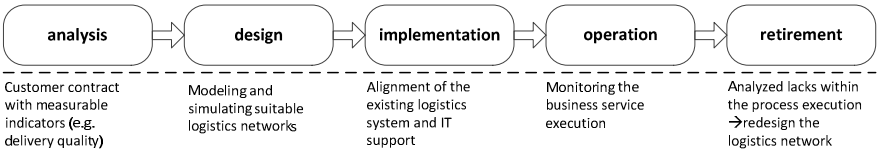



**Fig. 1.** 4PL service lifecycle

In this contribution we focus at first on the design phase and, thus, one of the main activities of a 4PL to be competitive – the multi-step planning process for its customers. Depending on specific requirements this planning process includes e.g. process modeling, selecting providers, defining the needed services or building long-term forecasts in order to assure a viable and robust logistics systems (see Fig. 1). Although we can accommodate several steps of planning a complete logistics system, the results of the various steps are still isolated (i.e. we still have to provide same information in multiple systems). Hence, high costs and an error-prone overall planning process are the results. Therefore, it is shown in [7] that the integration of planning steps, especially the further use of process models for developing simulation models, is essential to avoid this and to reduce the overall time and effort.

Another problem we focus in this paper concerns the data acquisition. Nowadays, companies are overwhelmed with data. On the one hand, this can lead to a more robust, and thus, more precise simulation because the available database is bigger than ever. On the other hand, it is more difficult to select the needed data in the right quality, quantity and granularity. This flood of data must be processed in a suitable manner to meet the requirements of the presented 4PL simulation approach.

In this contribution we, therefore, present an approach for integrating simulation in the overall planning process of a 4PL. Firstly, we discuss the prerequisites and requirements our planning approach is designed for, followed by an overview of adjacent approaches and fields of study. Then we show how to build a simulation model from an already modeled process and service profiles which contain information concerning the performance of each provider and their offered services. After that, we present a method on how to deal with the increasing velocity, variety, value and volume of data and how to analyze and process this data for further use. Afterwards, we show how to apply this method in order to use the acquired data for simulation in form of service profiles. And finally we end with a conclusion and future work of the approach.

## 2 Planning Prerequisites and Requirements

A core competence and an important task of a 4PL is the planning, orchestration and choreography of complex logistics services integrating various subsidiary logistics service providers [8]. Therefore, different IT-systems are used. Within the design phase, relevant services have to be identified and liable providers have to be chosen and coordinated in the overall process. Moreover, the entire structure of the logistics network with regard to their temporal dependencies has to be validated. Appropriate instruments for this purpose are, for instance, process modeling languages (e.g. Business Process Model and Notation (BPMN), Event-driven Process Chain (EPC)). Davenport defines a process as "*a specific ordering of work activities across time and place, with a beginning, an end, and clearly identifies input and outputs: a structure for action*" [9] Processes can be also described as follows: a process is defined as a coherent, self-contained sequence of activities required to fulfill an operational task in order to produce a specific service or product [10]. Similar to this is the description in [11] in which a process is described as a collaboration between process roles which perform tasks on concrete artifacts. Though processes have been widely used within a company, the definition above also allows making use of processes in an inter-company context with the same purpose. Thus, process modeling as an activity for the definition and description of a process combines executive (organizational units), design objects (information objects) and tasks (activities) and connects them via different control flows regardless of organizational boundaries. Fields of application and purpose of process modeling are, for example, the documentation, preparing for automation or optimization. As processes are described as a structure for action, process modeling languages represent the static structure of business processes but the dynamic aspects are not considered. This leads to a growing use of simulation in logistics [12]. Therefore, process models are in many cases the basis for building simulation models [13].

Simulation allows the current behavior of a system to be analyzed and understood. *"Simulation is the imitation of the operation of a real-world process or system over time. […] Simulation is an indispensable problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyze the behavior of a system, ask what-if questions about the real system, and aid in the design of real systems. Both existing and conceptual systems can be modeled with simulation."* [14] In logistics, simulation methodology is becoming increasingly important for securing the planning, management and monitoring of material, personnel and information flows [12]. As complex logistics services are established for a long period of time radical changes during the operation phase are very expensive and often consume an enormous amount of time [15]. Thus, it is necessary to anticipate the future behavior of a logistics system prior to its implementation. Hence, simulation models of logistics networks can be used to improve the decision-making process in the planning phase. Especially discrete-event simulation (DES) is appropriate to enhance decision support by analyzing several system configurations, which differ in structure and behavior [16].

The use of simulation also leads to a number of problems. Building simulation models requires special training and experience to avoid errors, because it is a methodology that is learned over time. Furthermore, building simulation models and their analysis is expensive and consume an enormous amount of time. This can lead to a non-profitable use of simulation [14]. The use of different models within the

planning process (process model, simulation model) leads to another problem. Each time a model is slightly modified, any of the other models must also be revised. This also increases the modeling effort. One of the main problems with the use of simulation is the availability of valid input data in the right quality, quantity and granularity [17]. These are an essential precondition for high-value results [18]. In addition, the used data must be available and visible as fast as possible. Conventional approaches using data stored in data warehouses and are request-driven [19]. Thereby a simulation works on retrospective data.

For these problems, the following requirements are derived. The effort for the development of simulation models must be reduced. Especially as in the planning of logistics systems several models come to use. These models build upon one another and have dependencies among each other. A change in a model also leads to changes in subsequent models. Therefore, the use of simulation techniques has to be integrated in the planning process [7]. It must be ensured that the created process models within the planning process, based on a separate description of each logistics service, can be transferred automatically into a simulation model. However, the different fluctuations (e.g. fluctuations in demand, sales trend and seasonal fluctuations) of the entire logistics system, which can potentially arise, should be considered. On the one hand, this requirement aims to minimize the planning effort of a 4PL. On the other hand, manual errors in the creation of a simulation model should be avoided. Furthermore, the need for special training and special experience in simulation model building is reduced. Another requirement concerns the information acquisition. As a result of the information overload the investment in simulation projects for information acquisition is almost 50 % of the total project time. This leads to the need of an efficient approach for gathering information to support the logistics planner in all planning activities. To gain a robust simulation result the used data have to describe the current state of all logistics networks.

## 3 Related Work

**Simulation:** Simulation approaches are widely used in logistics in order to plan logistics systems. Ingalls discusses the benefits of simulation as a method to study the behavior of logistics networks [20]. Additionally, advantages and disadvantages are presented for analyzing supply chains with the use of simulation models in general. A concrete simulation approach is not provided. In [21] a commonly applicable simulation framework for modeling supply chains is presented. Instead of [20] they focus on a more technical perspective as they show an overview of event-discrete simulation environments in terms of domains of applicability, types of libraries, input-output functionalities, animation functionalities, etc. Cimino et al. also show how and when to use certain programming languages as a viable alternative for such environments. A modeling approach and a simulation model for supporting supply chain management is presented by Longo and Mirabelli in [22]. In addition, they provide a decision making tool for supply chain management and, therefore, develop a discrete event simulation tool for a supply chain simulation using Plant Simulation[1] including a modeling approach. All these approaches are relevant for developing an

[1] (based on eM-Plant) `http://www.siemens.com/plm/PlantSimulation`

integrated planning and simulation approach. However, all these approaches satisfy the 4PL's specific requirements (Section 2) only partially. The development of simulation models based on process models is insufficiently considered.

**Model Transformations:** For integrating simulation in the planning the use of transformation approaches for defining transformation models as a mediator between process and simulation models is interesting for our approach. In both approaches of [23, 24] a transformation model is used in an additional step in order to derive a simulation model from an already existing process model. Both approaches take the fact that process models are independently defined from simulation requirements. In practice, process models serve to foster transparency or documentation and to analyze the requirements for the introduction or implementation of new information systems. However, both approaches assume that a process model is defined using a specific modeling language (EPC).

**Data Acquisition for Simulation:** Bernhard et al. gives at first a theoretical overview of research results including theoretical definitions of terms like information, data, and knowledge. Based on this a process-oriented procedure model for information acquisition in a superordinate procedure model for simulation according to VDI 3633 [16] is presented. Furthermore, different taxonomies of methods from data acquisition, statistics and visualization and their utilization were analyzed and classified. In contrast, [25] propose the separation of the steps of information and acquisition from the modeling process. Therefore, the procedure model for simulation was extended by a separate handling of the model and the data. So a continuous and iterative verification and validation process should be provided. Kuhn et al. argue that a distinguished consideration of data collection and preparation is missing and fill this gap by another procedure model extended by a chain of sub-processes within information acquisition [26]. The paper proposes a procedure model of information acquisition which is more a task- and user-oriented approach. All these contributions have one thing in common: they assume that simulation projects are isolated from an overall integrated planning procedure and the development and analysis of simulation models is a project for its own. In our approach, simulation is part of an integrated planning process and an overall approach for a 4PL.

**Complex Event Processing:** Roth and Donath point out the use and advantages of complex event processing (CEP) for the 4PL business model for monitoring business processes and collecting real-time data [6]. Yao et al. analyze the application of CEP in hospitals by using RFID. They introduce a framework and provide a possible solution to improve patient safety, whereby the power of CEP is shown. Some parts of the approach can be partly adopted, but the framework is too abstract and not suitable for the presented application area [27]. Buchmann et al. investigate event processing in production, logistics and transportation. It is described how service-orientation and event processing can be combined and how the application area can benefit from real-time data. This paper covers the logistics area and discusses CEP as a powerful approach without going into detail [17]. Because of the high level consideration the paper only provides partial input for the work presented in this paper, but the discussion underpins the thoughts of the authors. Anymore these approaches do not meet the requirements of a 4PL business model, more precisely

for the simulation. In this contribution we present how CEP can be used for gathering and processing real-time data for simulation.

## 4 Simulation Approach

In Section 2 it is described that different models (e.g. process models, simulation models) are used for planning logistics processes. This section specifies how the transformation of process models into simulation models is implemented prototypically as part of an integrated planning process for a 4PL.

Process models describe functional or structural aspects relevant for a process. In the scope of a 4PL's planning process these components represent the different partial logistics services as part of the overall process. Within this research BPMN is used as modeling language for creating process models. Therefore, we presume that tasks of BPMN represent logistics services. In [28] an approach for formal and semantic description of services in the logistics domain using concepts of service-orientation and Semantic Web technologies is presented. The approach also categorizes and describes modular logistics services such as transport, handling, storage, value-added services, etc. using a logistics ontology. Concepts of this ontology are used in this research paper to refer from BPMN tasks to the description of specific logistics services. Thus, each BPMN task is assigned to a specific logistics service type. So, the result is a BPMN process model including all logistics services necessary to meet customer's requirements. Fig. 2 illustrates the prototypical implementation of logistics annotations (e.g. service types) within a BPMN-Modeller.

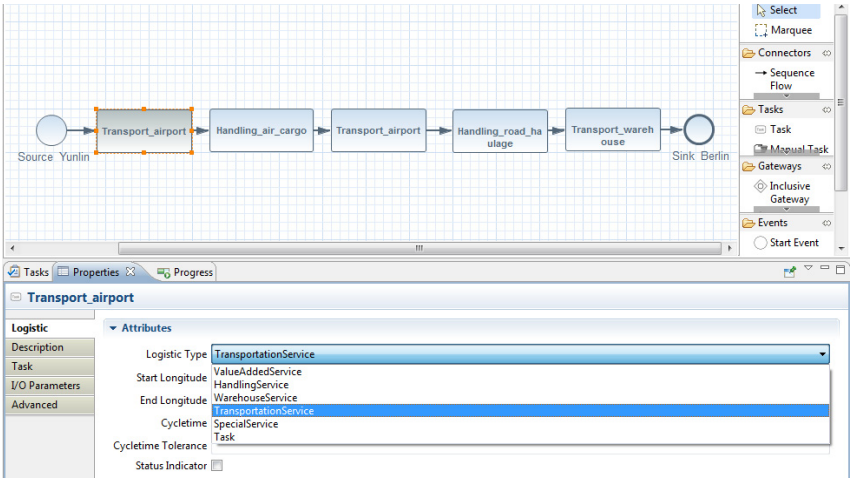



**Fig. 2.** BPMN-Task assigned to logistics service type

Despite having a process model and using this model as the basis for creating a simulation model, additional information such as the visualization of the processes, is necessary. Therefore, it was analyzed what information are additionally required to create a simulation model. The following elements are the basic concepts of simulation models in general and, therefore, also the basic concepts of DES environments [14]:

**Table 1.** Essential simulation concepts

| | |
|---|---|
| **Entities** | An entity represents real-world objects. Entities that move through the system (e.g. products, customers) are dynamic and entities that serve other entities (e.g. conveyors, machines, warehouse) are static. |
| **Events** | An event is an occurrence that changes the state of the simulation system and is the beginning and ending of an activity or delay (e.g. freight is loaded). |
| **Attributes** | An attribute describes the characteristics of an entity (e.g. time of arrival, due date priority, color). A set of entities may have the same attribute slots but different values for different entities, so the attribute value is tied to a specific entity. |
| **Activities** | An activity represents a specific period of time. The duration of this time period is known prior and can be a constant, a random value from a statistical distribution or input from a file, etc. (e.g. processing time of a machine). |
| **Delays** | A delay is an indefinite period of time. The duration is caused by some combination of system conditions (e.g. the freight is waiting for loading). |

The fundamental goal of simulation in logistics is the study of transport volumes and capacities of the partial logistics services over time to ensure that customers' demand can be met. So it is possible to analyze the flows of goods through the logistics system with regard to the capacity in order to identify bottlenecks early on. To create simulation models of a specific domain (e.g. logistics) primarily application-oriented modeling concepts are used [14]. Typical in logistics is the use of "modular concepts". These provide topological, organizational and / or informational elements - appropriately aggregated, predefined and parameterized from an application perspective - for a specific application domain [29]. Two simulation tools which are widely used in the logistics domain, which realize more or less the application-oriented modeling concept (Enterprise Dynamics (ED)[2] and Arena[3]), have been used to create different examples of simulation models to study transport volumes and capacities [30]. These tool-dependent models have been analyzed and compared in terms of the used modeling concepts and the required data. The common basic concepts were consolidated and used to create the metamodel shown in Fig. 6.

A simulation model basically consists of the following concepts. A *source* generates goods at predefined time periods. A *sink* is the concept where goods leave a model. The purpose of an *activity* is to handle goods. Therefore, *goods* enter an activity and remain there for a certain time. Moreover, an activity is assigned to a *service* type. All *time* periods can be described by a set of *distribution* functions. Regarding the service type, a *capacity* is an additional characteristic of an activity. For instance, an activity with the service type "warehouse service" is restricted by a maximum capacity and has a certain queuing strategy. The connecting elements between the activities are represented by two different *relations*. On the one hand, relations can be simple, i.e., without specific characteristics. On the other hand, a

[2] http://www.incontrolsim.com/
[3] http://www.arenasimulation.com

connection between activities can be represented by *conditional relations* with specific characteristics. Depending on certain conditions or probabilities one or the other path is used. The model itself is marked as a *ServiceAspect* and the elements representing characteristics are marked as *ServiceDescriptionElements*. So these marked elements represent the connection to other models [31].

Now, the question arises of where to get the information for building a simulation model. As already mentioned, modeling business processes including different partial logistics services using BPMN is part of the 4PL's planning process. These process models can be regarded as given in a service repository [31]. So we can use the structure (start, end, tasks, relations, gateways) of the process to derive the structure (source, sink, activities and relations) for the simulation model. Gateways, for example, in process models are represented by conditional relations. Further information is available in service profiles (see Section 6). So these profiles contain the specific information required to characterize the activities in the simulation model, e.g. time, quality or capacities. The method of collecting this information is described in Section 5 in more detail.

To combine the process model with the provider information, the process editor is extended by a provider selection. Based on the process model for each partial logistics services represented as BPMN tasks a suitable provider and the required information are selected. With this information and the underlying simulation metamodel (see Fig. 6), a 4PL can automatically generate a simulation model for a specific simulation tool. This requires that for each simulation tool transformation rules have to be defined only once. This approach enables a 4PL to make use of the advantages of simulation for securing the planning process and to improve decision-making without any special training and special experience in the creation of simulation models. Simulation models can be created in an easy and efficient way and the effort for comparing a set of different logistics network configurations is reduced. Furthermore, the simulation results serve to improve the planned process in form of a planning cycle (see Fig. 3).

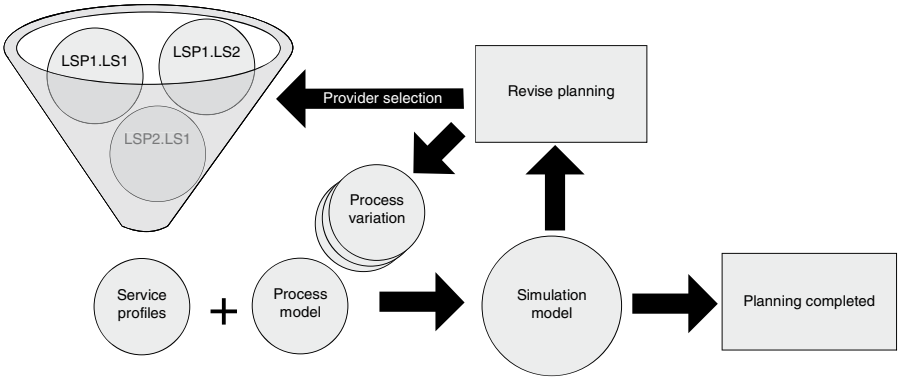



**Fig. 3.** Automated planning cycle within the design phase

## 5 Data Acquisition Using CEP

In this section the data acquisition for simulation using CEP is introduced, whereby the consideration is on an abstract level without going into detail of a particular tool or technique.

Complex event processing (CEP) is designed to deal with the increasing velocity, variety, value and volume of data, known as Big Data. CEP is defined as a set of tools and techniques for analyzing and controlling the complex series of interrelated events. Thereby the events are processed as they happen, thus, continuously and in a timely manner [19, 32]. An event is the central aspect of CEP and is defined as "*anything that happens, or is contemplated as happening (change of state)*" [33], e.g. a RFID-enabled good is recognized by a RFID reader. If an event summarizes, represents or denotes a set of other events, it is a so-called "complex event", e.g. a good left the issuing area [33]. In this paper it is assumed that CEP is already used to monitor an instantiated logistics network [6]. The next paragraph exemplifies this and emphasizes the adequacy of applying CEP in the area of a 4PL.

The outsourced service between the 4PL and the customer as well as between the 4PL and the service providers is contractually secured. A contract records the agreed upon obligations and responsibilities of contractual parties in terms of business process conditions [34]. These conditions are often expressed as goals which must be achieved by each party. The goals can be extracted from the customer needs or from legal regulations and are known as Service Level Objectives (SLOs), which define measurable indicators like delivery quality, delivery reliability or delivery flexibility. The contract must exist in a formalized form, whereby the CEP engine is capable to work with. This contract describes the target state of each logistics service (LS) realized by the participants of the network and acts like a pattern. As soon as the process execution is started (and thus instantiated) the 4PL has to ensure the fulfillment of the defined SLOs. To achieve this, internal (e.g. good left the issuing area) and external (e.g. traffic jam) data regarding to the good will be pushed to the 4PL. By doing this the 4PL can ensure that possible penalties (e.g. delayed or damaged good) will be hand out to the "faulty" participant of the network. If it is not traceable which participant of the network is the flaw, a logistics network would not be robust and sustainable over a longer period. Furthermore, the use of CEP allows to forecast, whether an instantiated process will meet the SLOs in the future or not [6]. The incoming data which describe the actual state is squared with the SLOs which describe the target state. This comparison takes place within the CEP engine. All data will be processed to evaluate the process execution of every logistics network partner and build up service profiles. The service profiles include key performance indicators which benchmark the logistics service providers (LSP) and their services. In contrast to current instruments, this evaluation takes place during the process run-time and not at the expiration (retirement, see Fig. 1) of a process. If delays are identified, an alarm will be triggered and data about the failure will be raised, e.g. the duration or reasons of a delivery delay. The following example and explanation should briefly describe the suitability of CEP in the area of the 4PL business model at a more detailed level.

Fig. 4 illustrates a possible material and data flow of a specific logistics network. As seen in the material flow layer, three LSP take part in the logistics network to accomplish the contract between the 4PL and the customer. The squares emblematize

the responsibilities for each LSP. Beside the actual transportation of the good, the data regarding to the good must be processed as well. The lower part of the data flow layer exemplifies the data sources within a logistics service (ERP, RFID, Barcode). These examples should clarify that there are a multitude of data sources with their own characteristics, which must be linked with the 4PL system. By using CEP it is possible to link nearly every data source in a short space of time, because CEP is loosely coupled. Thereby the 4PL gain situational awareness, because a high variety of data sources – internal and external (e.g. traffic systems) - can be linked rapidly. Moreover, CEP can handle the rising velocity of data while processing them in real-time, which will lead to a better availability and visibility of data. Using e.g. RFID leads directly to the challenge that a flood of data is generated. Moreover, companies are only interested in information with a high value. Therefore, it is necessary that a dispatcher does not receive messages such as "good_1 was read at reader_1 at 12:45 UTC". According to that, CEP provides filtering mechanism so that all redundant messages will be percolated, which will reduce the volume of data. The result is that only one message is received by the dispatcher. Moreover, it can be stated that the message "good_1 was read at reader_1 at 12:45 UTC" does not have a high information value. CEP offers the opportunity to aggregate data to obtain a higher information value. By using these mechanisms it is possible to aggregate technical data (e.g. RFID reads) to business processes, whereby the message is transformed to "good_1 for Mr. X left the warehouse at gate 2. The delivery is delayed for 45 minutes". This message is used to evaluate the performance of the logistics service (see e.g. LSP1.LS2 in Fig. 4) in form of service profiles. The profiles of each offered and operated logistics service are aggregated again to achieve an overall performance profile of every logistics service provider (provider profile). Furthermore, these benchmarks can be aggregated again to achieve a profile for the whole network (network profile). All of these profiles represent an essential input for other tasks like provider selection or simulation of newly planned logistics systems.

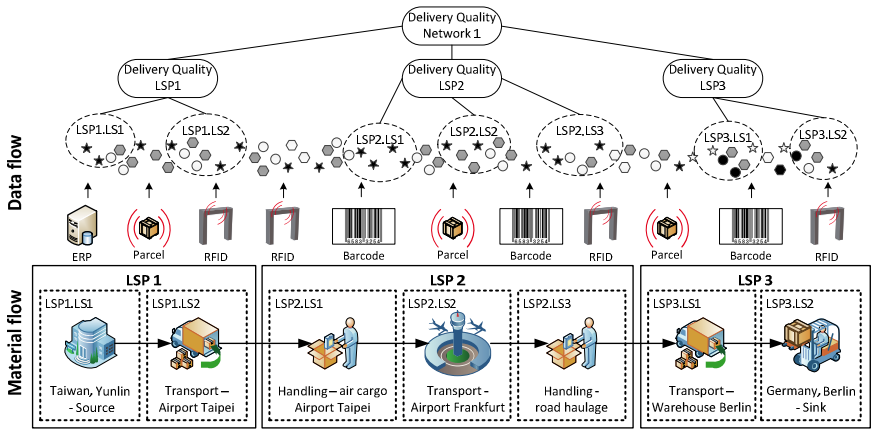



**Fig. 4.** Exemplified material and data flow of a logistics service instance

CEP provides powerful approaches to process data, transform them to information and link them to business processes. By doing so, CEP is a suitable technique to provide actual data at a desired granularity level in a timely manner. Hence, CEP is a suitable approach to monitor logistics networks and to support simulation with latest data constituted by service profiles. This leads to a better database, whereby simulations have not to rely on experience or outdated data. The available data includes current performance profile built up from internal and external data and can be combined as desired. This data can be used to simulate logistics networks at an early point of creation and generates more reliable predictions.

## 6 Service Profiles as Data Source for Simulation

The previous section explains how the logistics network including the different logistics services are monitored with the use of CEP during the operation phase.

The approach allows to process data and transforms them to information. This way of data acquisition is performed over all running processes in all logistics networks including their logistics services provided by the different service providers. With the use of CEP it is possible to create and update service profiles dynamically and at run-time. These profiles are provided in real-time at an appropriate granularity level and, thus, meet the requirements of simulation in the 4PL business model. Because CEP facilitates situational awareness by using external data and, thus, observes the environment, it is feasible to prioritize profile parameters far easier. There can be a multitude of reasons why a LSP does not comply with the delivery period, e.g. traffic jam or just driving too slow, which are interesting for creating profiles and causes different ratings. The profiles also include the positive and negative deviation from the agreed conditions. All these aspects are attached to the corresponding events and allow a detailed evaluation of a specific logistics service and its provider. So for each single logistics service a profile with parameter like service level, lead time, etc. is created and is updated continuously. The services provided by a single logistics provider can be aggregated to provider profiles. This evaluation is used for the provider selection and for simulation (see Fig. 1). In summary, a closer look at the service lifecycle in conjunction with the presented approaches is given (see Fig. 5).

In the *analysis* phase the 4PL establish a pool of LSP with a description of their capabilities. Furthermore, a contract between the customer and the 4PL with some SLOs (e.g. delivery time) is drawn up.

The 4PL models a logistics network regarding to the above mentioned constraints. This model is illustrated in a formalized way in form of a process model. The process model describes the static structure of a business process. To analyze the dynamic behavior of a system a simulation model of the logistics network based on the process model is used. In order to minimize the modeling effort and the manual errors creating a simulation model, the simulation approach is integrated in the planning process [7]. The *design* phase ends with this step.

In the *implementation* phase the 4PL has to encompass the designed process by integrating the LSPs and the alignment of the existing logistics information systems and IT support.

During the *operation* phase, the 4PL monitors the logistics service execution (process instance) realized by the LSPs. On that account, the 4PL processes internal

(e.g. temperature) and external (e.g. traffic information) data regarding to the good by using CEP. The fulfillment of a logistics service and the resulting service profiles are created and updated continuously (see Section 5). If it is predictable that an ongoing contract and the regarding SLOs cannot be fulfilled, the 4PL will be timely informed to adopt compensating measures. In this case, the service lifecycle will be restarted. The gathered real-time data is an important input for the analysis and design phase for new logistics services, because the gap between physical and virtual can be bridged.

At the end of the service lifecycle the contract expires and the service terminates (*retirement*). The process execution is analyzed and possible lacks will be eradicated.

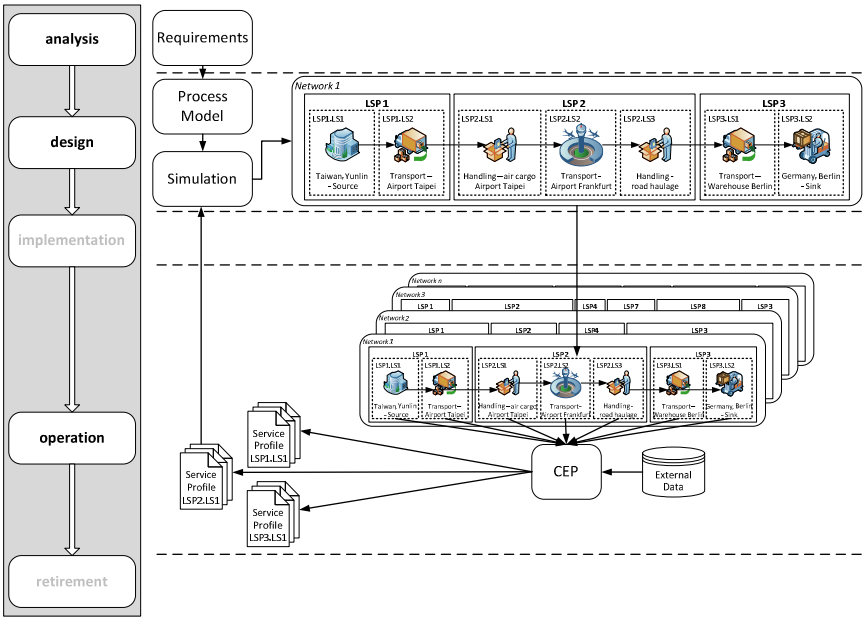



**Fig. 5.** Service lifecycle focused on using CEP for acquiring real-time data for simulation

## 7 Conclusion and Future Work

From the 4PL point of view a major issue is an integrated planning process with which a multitude of service providers are orchestrated in order to realize a customer's logistics network. Therefore, we clarified the requirements of our approach in general, in that we presented under which circumstances our approach can be applied. In this context, we presented an integrated planning approach based on a process transformation into a simulation in order to make sure, that the modeled process is robust, cost efficient and meets the customer's requirements. If the result of the simulation is satisfying and we can determine a valid combination of services and service providers respectively, the planning process can be closed at this point. Essential to this approach is the acquisition of real-time data for simulation. In [6] it is shown, that CEP is suitable to monitor an instantiated logistics network within the area of the 4PL business model. In contrast to that paper, we developed the approach

further in that we discuss the use of CEP to acquire data for planning new complex logistics services, especially for simulating them. To prepare these data we propose the development of service profiles which can be aggregated to provider profiles. Finally, we illustrated the overall approach and the interaction between simulation, CEP and service profiles focused on the service lifecycle.

Subject of the future work is the development of service profile patterns. Depending on the service type (e.g. transportation, storage) a service profile consists of different KPI's. In addition, a set of indicators have to be worked out to create expressive provider profiles for further planning steps (e.g. provider selection). Furthermore, the prioritization of delays must be investigated.

**Acknowledgments.** The work presented in this paper was funded by the German Federal Ministry of Education and Research under the project LSEM (BMBF 03IPT504X) and LogiLeit (BMBF 03IPT504A).

## References

1. ten Hompel, M., Schmidt, T., Nagel, L.: Materialflusssysteme: Förder- und Lagertechnik. Springer, Heidelberg (2007)
2. Arnold, D., Furmans, K., Isermann, H., Kuhn, A., Tempelmeier, H.: Handbuch Logistik. Springer, Heidelberg (2008)
3. Gudehus, T., Kotzab, H.: Comprehensive logistics. Springer, New York (2012)
4. Schmitt, A.: 4PL-Providing-TM als strategische Option für Kontraktlogistikdienstleister: eine konzeptionell-empirische Betrachtung. Dt. Univ.-Verl., Wiesbaden (2006)
5. Nissen, V., Bothe, M.: Fourth Party Logistics. Logistik Management 4, 16–26 (2002)
6. Roth, M., Donath, S.: Applying Complex Event Processing towards Monitoring of Multi-party Contracts and Services for Logistics – A Discussion. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBIP, vol. 99, pp. 458–463. Springer, Heidelberg (2012)
7. Mutke, S., Klinkmüller, C., Ludwig, A., Franczyk, B.: Towards an Integrated Simulation Approach for Planning Logistics Service Systems. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBIP, vol. 99, pp. 306–317. Springer, Heidelberg (2012)
8. Thiell, M., Hernandez, S.: Logistics services in the 21st century: supply chain integration and service architecture. In: Service Science and Logistics Informatics, vol. 2010, pp. 359–378. Business Science Reference, Hershey (2010)
9. Davenport, T.H.: Process innovation: reengineering work through information technology. Harvard Business School Press (1993)
10. Staud, J.L.: Geschäftsprozessanalyse: Ereignisgesteuerte Prozessketten und objektorientierte Geschäftsprozessmodellierung für Betriebswirtschaftliche Standardsoftware. Springer, Heidelberg (2006)
11. Münch, J., Armbrust, O., Kowalczyk, M., Soto, M.: Software Process Definition and Management. Springer, Heidelberg (2012)
12. Terzi, S., Cavalieri, S.: Simulation in the supply chain context: a survey. Computers in Industry 53, 3–16 (2004)
13. Rosemann, M., Schwegmann, A., Delfmann, P.: Vorbereitung der Prozessmodellierung. In: Becker, J., Kugeler, M., Rosemann, M. (eds.) Prozessmanagement: Ein Leitfaden zur Prozessorientierten Organisationsgestaltung, pp. 45–103. Springer, Heidelberg (2005)
14. Banks, J.: Handbook of simulation principles, methodology, advances, applications, and practice. Wiley Co-published by Engineering & Management Press, New York (1998)

15. Alves, G., Roßmann, J., Wischnewski, R.: A discrete-event-simulation approach for logistic systems with real time resource routing and VR integration. In: International Conference on Computational Systems Engineering (ICCSE 2009), World Academy of Science, Engineering and Technology, WASET, Venedig, Italy, pp. 476–481 (2009)
16. VDI-Richtlinie: 3633, Blatt 1: Simulation von Logistik-, Materialfluß- und Produktionssystemen. Beuth, Berlin (2010)
17. Buchmann, A., Pfohl, H.-C., Appel, S., Freudenreich, T., Frischbier, S., Petrov, I., Zuber, C.: Event-Driven Services: Integrating Production, Logistics and Transportation. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010 Workshops. LNCS, vol. 6568, pp. 237–241. Springer, Heidelberg (2011)
18. Bernhard, J., Wenzel, S.: Information Acquisition for Model-based Analysis of Large Logistics Networks. In: Proceedings of SCS-ESM (2005)
19. Etzion, O., Niblett, P.: Event processing in action. Manning, Greenwich (2011)
20. Ingalls, R.G.: The value of simulation in modeling supply chains, pp. 1371–1376. IEEE Computer Society Press (1998)
21. Cimino, A., Longo, F., Mirabelli, G.: A general simulation framework for supply chain modeling: state of the art and case study. Arxiv (2010)
22. Longo, F., Mirabelli, G.: An advanced supply chain management tool based on modeling and simulation. Comput. Ind. Eng. 54, 570–588 (2008)
23. Petsch, M., Schorcht, H., Nissen, V., Himmelreich, K.: Ein Transformationsmodell zur Überführung von Prozessmodellen in eine Simulationsumgebung. In: Loos, P., Nüttgens, M., Turowski, K., Werth, D. (eds.) Modellierung betrieblicher Informationssysteme - Modellierung zwischen SOA und Compliance Management, Saarbrücken, Germany, pp. 209–219 (2008)
24. Kloos, O., Schorcht, H., Petsch, M., Nissen, V.: Dienstleistungsmodellierung als Grundlage für eine Simulation. In: Dienstleistungsmodellierung 2010, pp. 86–106 (2010)
25. Rabe, M., Spieckermann, S., Wenzel, S.: Verification and validation activities within a new procedure model for v&v in production and logistics simulation. In: Proceedings of the 2009 Winter Simulation Conference (WSC), pp. 2509–2519 (2009)
26. Kuhnt, S., Wenzel, S.: Information acquisition for modelling and simulation of logistics networks. Journal of Simulation 4, 109–115 (2010)
27. Yao, W., Chu, C.H., Li, Z.: Leveraging complex event processing for smart hospitals using RFID. J. Netw. Comput. Appl. 34, 799–810 (2011)
28. Hoxha, J., Scheuermann, A., Bloehdorn, S.: An Approach to Formal and Semantic Representation of Logistics Services. In: Workshop on Artificial Intelligence and Logistics (AILog), pp. 73–78 (2010)
29. Kuhn, A., Wenzel, S.: Simulation logistischer Systeme. In: Arnold, D., Furmans, K., Isermann, H., Kuhn, A., Tempelmeier, H. (eds.) Handbuch der Logistik. Springer, Heidelberg (2008)
30. Motta, M., Wagenitz, A., Hellingrath, B., Weller, R.: Gestaltung logistischer Netzwerke. In: Advances in Simulation for Production and Logistics Applications, pp. 21–30 (2008)
31. Augenstein, C., Mutke, S., Ludwig, A.: Integration von Planungssystemen in der Logistik–Ansatz und Anwendung. In: 11. Internationale Tagung Wirtschaftsinformatik, pp. 1391–1405 (2013)
32. Luckham, D.C.: The power of events: an introduction to complex event processing in distributed enterprise systems. Addison-Wesley, Boston (2002)
33. Luckham, D., Schulte, R.: EPTS Event Processing Glossary v2.0. Event Processing Technical Society (2011)
34. Weigand, H., Xu, L.: Contracts in E-Commerce. In: Meersman, R., Aberer, K., Dillon, T. (eds.) Semantic Issues in E-Commerce Systems. IFIP, vol. 111, pp. 3–17. Springer, Boston (2003)

# Appendix

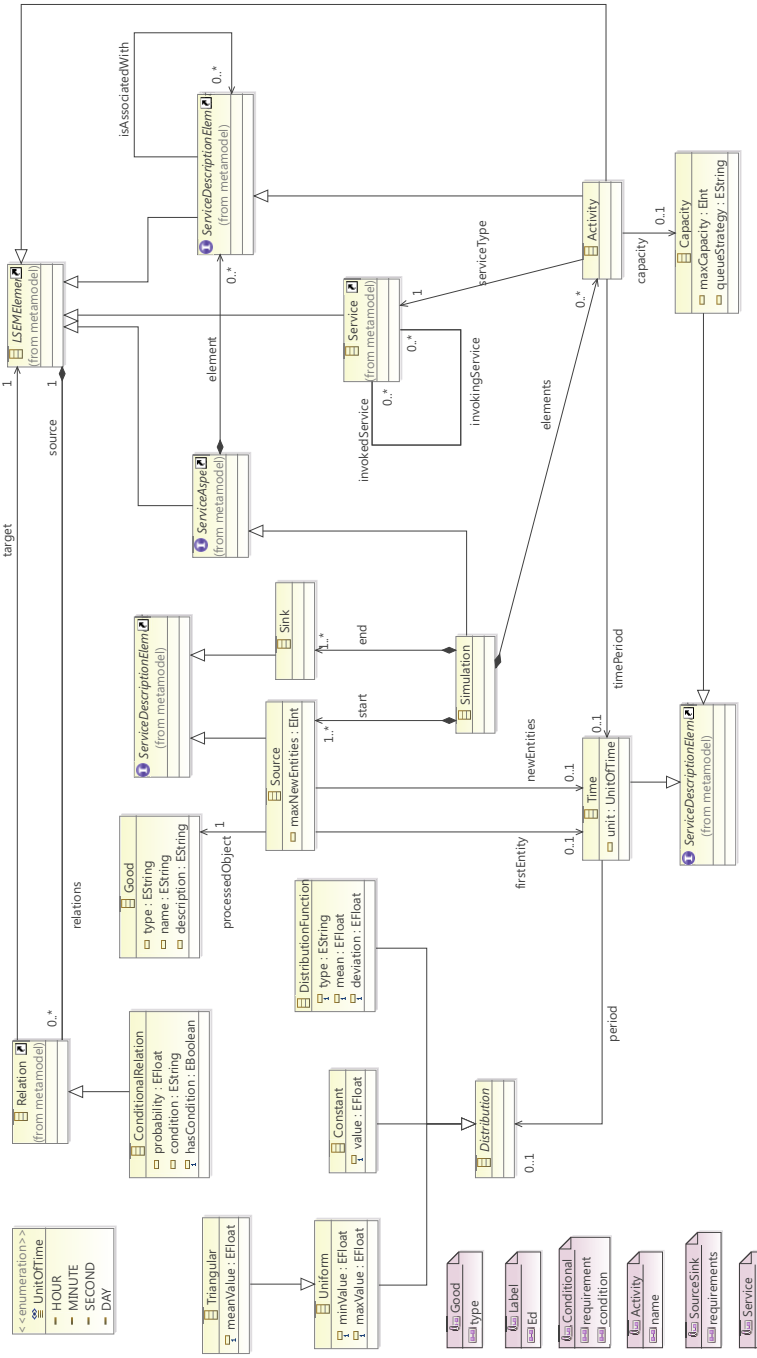



**Fig. 6.** Generic simulation metamodel

# Column Generation Based Heuristic for the Three Dimensional Vehicle Loading Problem

Batoul Mahvash[1], Anjali Awasthi[1], and Satyaveer Chauhan[2]

[1] Mechanical Engineering Department, Concordia Institute for Information Systems Engineering
{b_mahvas@encs.concordia.ca,awasthi@ciise.concordia.ca}
[2] John Molson School of Business,
Concordia University, Montreal, Quebec, Canada
satyaveer.chauhan@concordia.ca

**Abstract.** In this work we present a vehicle loading problem that consists of packing a given set of items into a minimum number of vehicles for delivery. A heuristic approach based on Dantzig-Wolfe decomposition is developed and tested on a widely available data set. A greedy search procedure is developed to speed up the column generation approach. A problem specific branching technique is used to generate integer solutions in a reasonable time. Numerical experimentation is done to compare the performance of the developed approach with the available results in the literature.

**Keywords:** Three-dimensional vehicle loading problem, Column Generation, Heuristics.

## 1 Introduction

Loading, packing and distribution activities have considerable cost share in overall logistics costs and as supply chains are becoming lean and competitive the pressure to deliver goods economically is mounting and thus these problems have generated considerable interest among academicians and practitioners. Furthermore, loading and distribution operations vary across industry and thus several variants of such problems exist and as these problems are combinatorial in nature, each variant may require new insights and approaches.

In this work we address a three-dimensional vehicle loading problem (3DVLP) that consists of packing goods into a minimum number of vehicles. In 3DVLP goods are expressed in terms of cubic items. Items have to be loaded orthogonally into vehicles without overlapping. Items may have fixed orientation, that is, items cannot be rotated around all axes. A number of vehicles of fixed sizes, specified with length, width, and height exists with rectangular loading surface accessed only from one side.

To solve the strong NP-hard 3DVLP, we apply a column generation (CG) technique embedded in a branch-and-bound method which is also popularly known as the branch-and-price method. The 3DVLP is formulated as a set-covering model based on Dantzig-Wolfe decomposition. The associated set-covering model is split into a

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 257–268, 2013.

master problem that is a linear relaxation of the model and a sub-problem. The master problem deals with assignments of items to vehicles, while the sub-problem deals with the accommodation of items within those vehicles. The master problem deals with a large number of variables and to handle this complexity a CG technique is used, which considers just a subset of variables. A set of heuristic pricing problems are solved to identify the new variables (feasible columns) with negative reduced cost. To check the feasibility of a column, a modification of the extreme point-based heuristic method [1] is used. At the termination of the column generation step (when no new entering column/variable can be identified) if the solution to the master problem is non-integer, a set of branching rules in a heuristic framework is applied.

Extensive computational results on test instances indicate that solutions obtained by the CG technique are comparable to those produced by a two-level tabu search meta-heuristic ($TS^2PACK$) developed by Crainic et al. [2]. The performance of the CG technique is only compared with $TS^2PACK$, since this method outperforms other techniques for solving the 3DVLP.

We assume that $I$ is the set of all cubic items. Each item $i \epsilon I$ is characterized by length $l_i$, width $w_i$ and height $h_i$. There is an unlimited number of vehicles of fixed sizes, specified with length $L$, width $W$, and height $H$, with rectangular loading surface accessed only from one side. Without loss of generality, it is assumed that the length and width of vehicles are along the x axis and y axis, respectively. The coordinates of back-left-bottom (BLB) corner of the vehicle is fixed at origin.

## 2 Literature Review

In the literature, vehicle loading or container loading problems are referred to as bin-packing problems. Pisinger [3] pointed out that there are different variants of packing problems depending on the objective function and restrictions available in the problem. There is a huge literature on multi-dimensional packing problems. Iori and Martello [4] have broken down the multiple-dimensional packing problem into the two-dimensional bin packing problem (2BPP), two-dimensional strip packing problem (2SPP), three-dimensional bin packing problem (3BPP) and three-dimensional strip packing problem (3SPP). They argued that genetic algorithms and tabu search are often used as heuristics to solve multi-dimensional bin packing problems.

3BPP deals with packing a given set of rectangular boxes into a minimum number of large identical three-dimensional bins. Most researches focus on the 3BPP with identical bins and boxes with fixed orientation. Martello et al. [5] were the first to develop a two-level branch & bound exact algorithm to the 3BPP based on the concept of corner points (CPs). The CPs are points within the residual space of the vehicle where a new item can be accommodated. They tested their exact method on instances with up to 200 items.

Den Boef et al. [6], Fekete and Schepers [7], Martello et al. [5] and Boschetti [8] discussed and proposed the lower bounds for the 3BPP. The lower bounds proposed by Boschetti dominate all bounds by the other authors.

Pisinger [3] proposed a heuristic based on the wall-building approach to decompose the 3BPP into a number of layers split into a number of strips. Gehring and Bortfeldt [9] presented a genetic algorithm for loading a set of heterogeneous cartons into a single container [10].

A new heuristic called height first–area second (HA) was presented by Lodi et al. [10]. In this heuristic, items as layers are packed into bins by solving a 1D-BP problem. To obtain layers and in order to have better solutions, items are clustered twice, once by height and once by decreasing area of their base. The HA heuristic can obtain the best results on benchmark tests.

To better utilize the container's volume, a new concept of extreme points (EPs), an extension of the CPs was introduced by Crainic et al. [1]. Using this concept the authors designed new practical heuristic methods based on the first fit decreasing and the best fit decreasing. These heuristic methods yield better results compared to all heuristics and complex methods for the 3D-BP in the literature.

In the following works in the literature the packing problem was formulated as set covering model and solved by a column generation technique.

Gilmore and Gomory [11] [12] were the first to formulate a packing or cutting problem as a set covering problem and performed column generation for the one dimensional and two-dimensional cutting stock problem. Oliveira and Ferreira [13] extended the work of Gilmore and Gomory to three-stage and general multi-stage cutting stock problems and applied fast greedy heuristic to generate columns. If the heuristic fails to generat a column, a slower exact algorithm was applied. A set covering formulation for bin-packing problems was also presented by Monaci and Toth [14]. They applied fast greedy heuristics to generate columns and a lagrangian-based heuristic algorithm to solve the associated set-covering formulation.

A hybrid algorithm for the two-dimensional bin packing problem based on column-generation has been proposed by Pisinger and Sigurd [15]. They solved the corresponding pricing problem through a constraint-satisfaction problem (CSP) generating valid inequalities in a branch-and-cut method. They developed a branch-and-price-and-cut algorithm capable of solving to optimality several instances with up to n = 100 rectangles and obtained efficient lower bounds in reasonable time. Puchinger and Raidl [16] presented a branch-and-price algorithm based on a set covering formulation for the two-dimensional bin packing problem. They applied four proposed pricing methods to quickly generate columns.

## 3 The Set-Covering Formulation

The key idea of the set-covering formulation is to enumerate all feasible loadings of the problem. Let $\mathcal{L}$ be the set of all feasible loadings associated to a single vehicle. For each feasible loading $l \in \mathcal{L}$, a binary variable $x_l$ is defined equal to 1 if loading $l \in \mathcal{L}$ appears in the solution, otherwise it is 0. The binary parameter $a_i^l$ is equal to 1 if feasible loading $l$ contains item $i \epsilon\, I$, otherwise it is 0. The set-covering formulation of 3DVLP is:

$$Min \sum_{l \in \mathcal{L}} x_l \tag{1}$$

s.t.:

$$\sum_{l \in \mathcal{L}} a_i^l x_l \geq 1 \qquad \forall\, i \in I \tag{2}$$

$$x_l \in \{0,1\} \qquad \forall\, l \in \mathcal{L} \tag{3}$$

Objective function (1) determines a minimum number of vehicles. Constraints (2) ensure that each item must be covered by at least a feasible loading selected in the solution.

## 4 Column Generation (CG) Technique

A linear programming (LP) relaxation of the set-covering formulation, without integrality constraints on $x$ variables, is called master problem (MP). Since MP may contain a very large number of variables, the CG technique, which works with a limited number of variables, is used. The corresponding LP relaxation model to the partial set of loadings is called the restricted master problem (RMP). In fact the CG technique is an efficient approach capable of solving problems with large number of variables. Using CG, the majority of the loadings are not explicitly considered and RMP is solved for a partial set of loadings. Based on a solution to RMP in each iteration of the CG technique more loadings with minimum reduced cost are generated and added to the RMP step by step to reach the optimal solution to MP.

Suppose that $\pi_i$, $i \epsilon\, I$ be the set of dual variables associated with the current solution $x_l$ to the RMP. $c_l^{\pi} = 1 - \sum_{i \in I} a_i^l \pi_i$ is defined as reduced cost associated to loading $l \in \mathcal{L}$. Only the feasible loadings with the negative reduced cost can improve the current solution to the RMP. In each iteration of the CG technique, we must solve the pricing problem that consists of finding a feasible loading $l \in \mathcal{L}$ with minimum negative reduced cost. If no feasible loading with negative reduced cost exists, then there is no column (loading) to be added to RMP, and, therefore, the current solution is optimal to the RMP and MP.

### Heuristic Pricing

To guarantee feasible loadings with the negative reduced cost a modification of the algorithm of Crainic et al. [1] is applied for different items sorting rules. In fact, items are sorted according to non-increasing values of $\pi_i$, $\frac{\pi_i}{h_i.w_i.l_i}$, $\frac{\pi_i}{h_i+w_i+l_i}$, $\frac{\pi_i}{h_i}$, $\frac{\pi_i}{l_i}$, $\frac{\pi_i}{w_i}$. These sorting rules help us to find feasible loadings with negative reduced cost.

For each of the sorted lists of items by the mentioned rules, items are inserted into a vehicle one after the other using a modified algorithm until the residual capacity of the vehicle is sufficient to accommodate one more item. If the reduced cost associated to this feasible loading is negative, it is added to RMP as new column. This method

usually gives a promising loading with negative reduced cost, but not necessarily minimal. To reduce the total number of iterations and the overall computing time, all profitable columns found in each iteration of the CG technique are inserted into the RMP. If no further columns with negative reduced costs can be found by this procedure, the column generation stops.

## 5 Branching

After terminating column generation if the current solution to the RMP is integer, it would be a near-optimal or incidentally an optimal solution to the 3DVLP. Otherwise, an integer solution has to be determined through branch-and-price.

In this work we consider two branching strategies. The first one is a depth-first heuristic (D_FH) branching in which variables $x_l$ with fractional values more than a given specific value, are fixed to one. If there are no variables with fractional value more than the specific value, the variable with the largest fractional value is fixed to one. After fixing variables, more columns are generated using the heuristic pricing algorithm until there is little or no change in the RMP solution. Repeating this branching strategy leads to an integer solution or infeasible solution.

The other branching rule applied in this work is a general rule for the set-partitioning problem originally suggested by Ryan and Foster [18]. Given a fractional LP solution to a set-covering problem, we choose a pair of items to branch on. Two selection rules are applied: 1- select the pair (of items) *i* and *j* with maximum volume from a loading with high fractional value in the solution; 2-select the pair (items) *i* and *j* with maximum volume from a loading whose corresponding value is closest to 0.5. In one branch we force that the pair of items *i* and *j* has to be assigned to the same vehicle and to different vehicles on the other branch. The first branch is performed by adding constraint $\sum_{l\in\mathcal{L}} a_i^l a_j^l x_l = 1$ to the model, and the second branch by adding two constraints $\sum_{l\in\mathcal{L}}(1 - a_i^l)a_j^l x_l = 0$ and $\sum_{l\in\mathcal{L}} a_i^l(1 - a_j^l)x_l = 0$ to the model.

In the branch where the pair *i, j* must be in the same vehicle, we eliminate any loading from the RMP that contains *i* or *j* individually. On the other branch, where the pair of items *i* and *j* must not be in the same vehicle, any loading that contains both items *i* and *j* is eliminated from RMP.

We choose first the branch where a pair of items must be in the same vehicle and perform a depth first tree search. In this tree search in each branching, the pair of items in the same vehicle is fixed. The algorithm stops when a good integer solution is found. Backtracking is performed in a depth first manner when necessary. Unfortunately, this branching rule is very time-consuming.

## 6 Generating an Initial Solution

The CG technique in principle can start with the identity matrix as initial columns. The computational time of the CG technique can be decreased significantly by

initializing the algorithm with a good feasible solution. In this work an initial solution is obtained by a modification of the algorithm of Crainic et al. [1]. Given a list of sorted items specified by a sorting rule, items are loaded into the vehicle one after the other based on the concept of Extreme Points (EPs) [1]. When an item cannot be loaded into the current vehicle, a new vehicle is created. Placing an item into a vehicle splits vehicle's space into new volumes. EPs provide information about all possible empty positions where new items may be loaded.

The basic process of generating EPs is as follows. Suppose that an item $i$ with dimensions $(l_i, w_i, h_i)$ is added to a given loading in position $(x_i, y_i, z_i)$. This placement produces a set of EPs that are projections of point $(x_i + l_i, y_i, z_i)$ along the $y$ and $z$ axes, point $(x_i, y_i + w_i, z_i)$ along the $x$ and $z$ axes, and point $(x_i, y_i, z_i + h_i)$ along the $x$ and $y$ axes (see Figure 1). In each direction, if there are some items between item $i$ and the wall of the vehicle, a point is projected on the nearest item.

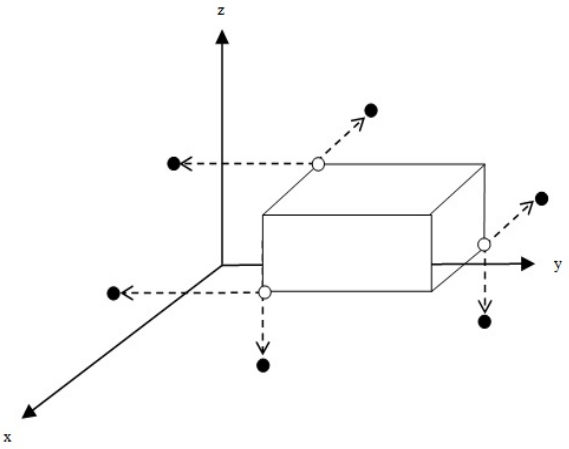



**Fig. 1.** EPs (black circles) generated by placing an item into a vehicle

If the vehicle is empty, placing an item in position (0,0,0) generates three EPs in points $(l_i, 0,0)$, $(0, w_i, 0)$ and $(0,0, h_i)$.

### 6.1 Introducing New EPs

Suppose that there is a loading of two items as shown in Figure 2. All EPs related to this loading are illustrated by black circles. To add a new item $k$ with dimensions $(l, w, h)$ into the current loading, all EPs are considered, but no one is appropriate. The only place where item $k$ can be accommodated is the point specified by the white circle, but this point is not in the set of extreme points.

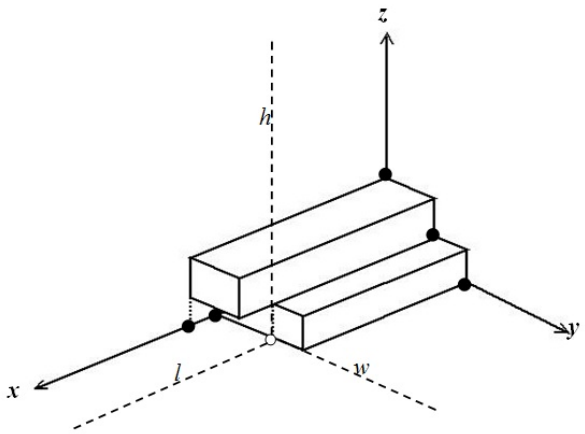



**Fig. 2.** An example of new EP (white circle)

To generate new extreme points we apply the following process. Suppose that an item $i$ with dimensions $(l_i, w_i, h_i)$ is added to a packing in position $(x_i, y_i, z_i)$. Let $p_i$ be the projection of point $(x_i + l_i, y_i + w_i, z_i)$ along the $z$ axes (see Figure 3). In this direction, if there are some items between item $i$ and the floor of the vehicle, the projection is on the nearest item. The new EPs are projections of point $p_i$ along the $x$ and $y$ axes. In these directions, if there are some items between point $p_i$ and the wall of the vehicle, the projection is on the nearest item.

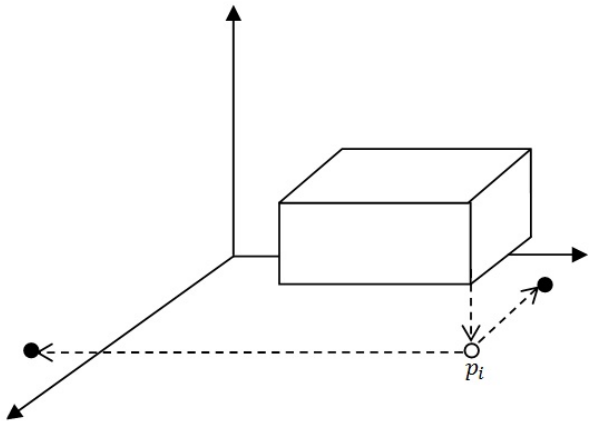



**Fig. 3.** New EPs (black circles) after loading an item

## 6.2 Sorting Rules

As mentioned by Crainic et al. [1], there are several item ordering rules from which different versions of heuristics can be developed. They presented a list of rules

yielding the best experimental results. In the present work, we apply two sorting rules Clustered Area-height and Clustered Height-area [1]. In the Clustered Area-height, for a given $\alpha \in [1,100]$ a set of intervals $A_{k,\alpha} = [\frac{(k-1)LW}{100}\alpha, \frac{kLW}{100}\alpha]$ is defined based on the vehicle area $L \times W$. Items whose base areas are in the same interval are assigned into a group. Groups are ordered according to $k$ and items of a group are sorted by decreasing values of their height. In the Clustered Height-area, for a given $\alpha \in [1,100]$ a set of intervals $h_{k,\alpha} = [\frac{(k-1)H}{100}\alpha, \frac{kH}{100}\alpha]$ is defined based on the vehicle height $H$. Items whose height is in the same interval are assigned in a group. Groups are ordered according to $k$ and items of a group are sorted by decreasing values of their base area.

### 6.3 Pseudocode for Generating an Initial Solution

All items are loaded into vehicles by the `Loading(SI,(L,W,H))` procedure in which SI is a list of sorted items resulted by rules Clustered Area-height or Clustered Height-area and (L,W,H) are the dimensions of the vehicles. As long as SI is not empty, one vehicle after the other is filled. Let LEPs be the set of EPs initialized by point (0, 0, 0). For each item $i \in SI$, `PutItemInVehicle` procedure scans list LEPs from first to end to identify an EP $p$ on where an item $i$ can be placed. It returns true, if item $i$ can be loaded, otherwise false. In fact, an item can be placed on an EP, if placement of its left-back-bottom corner on EP does not overlap with the other items available in the vehicle. If item $i$ can be placed on more than one EP, the EP with the lowest z, y, x coordinates in order is selected. `Update_LEPs` procedure generates and inserts new EPs, generated by placing item $i$ to LEPs. The residual spaces of EPs are updated by the `UpdateResidualSpace` procedure. Items are added to a vehicle as long as possible. If no items fit into a vehicle, the vehicle is closed, and a new vehicle is initialized and so on. This process is repeated until all items are loaded into vehicles. Note that procedures `Update_LEP` and `UpdateResidualSpace` are Algorithms 1 and 2 in Crainic et al. [2], respectively. After placement of an item into a vehicle the total set of EPs is updated by the `UpdateResidualSpace` procedure.

```
Loading(SI,(L,W,H))
  while SI is not empty do
    Initialize new vehicle v with dimensions (L,W,H)
    LEPs= {(0, 0, 0)}.
    for each item i • SI do
      if PutItemInVehicle(i, LEPs) then
        Update_3DEPL(i,p,LEPs)
        UpdateResidualSpace(i,LNVs)
        Remove item i from SI
      Endif
    Endfor
    Close vehicle v
  Endwhile
```

## 7 Computational Experiments

The heuristic CG algorithm is coded in C++ and run on a Dell 2.20 GHz. The algorithm is tested on a set of 320 instances generated by Martello et al. [5] for the three-dimensional packing problem. These instances are classified into eight classes, each of which contains four different types of items. The number of items, in each instance, could be equal to either 50, 100, 150, or 200. For each class and instance size, ten different problem instances based on different random seeds are generated. For classes 1 to 5, the bin size is $W = H = L = 100$ and the following five types of items are considered:

- Type 1: $w_i$ uniformly random in $[1, \frac{1}{2}W]$, $h_i$ uniformly random in $[\frac{2}{3}H, H]$, and $l_i$ uniformly random in $[\frac{2}{3}L, L]$.
- Type 2: $w_i$ uniformly random in $[\frac{2}{3}W, W]$, $h_i$ uniformly random in $[1, \frac{1}{2}H]$, and $l_i$ uniformly random in $[\frac{2}{3}L, L]$.
- Type 3: $w_i$ uniformly random in $[\frac{2}{3}W, W]$, $h_i$ uniformly random in $[\frac{2}{3}H, H]$, and $l_i$ uniformly random in $[1, \frac{1}{2}L]$.
- Type 4: $w_i$ uniformly random in $[\frac{1}{2}W, W]$, $h_i$ uniformly random in $[\frac{1}{2}H, H]$, and $l_i$ uniformly random in $[\frac{1}{2}L, L]$.
- Type 5: $w_i$ uniformly random in $[1, \frac{1}{2}W]$, $h_i$ uniformly random in $[1, \frac{1}{2}H]$, and $l_i$ uniformly random in $[1, \frac{1}{2}L]$.
- Class 1: type 1 with probability 60%, type 2, 3, 4, 5 with probability 10% each.
- Class 2: type 2 with probability 60%, type 1, 3,4, 5 with probability 10% each.
- Class 3: type 3 with probability 60%, type 1, 2, 4, 5 with probability 10% each.
- Class 4: type 1 with probability 60%, type 1,2, 3, 5 with probability 10% each.
- Class 5: type 2 with probability 60%, type 1, 2,3,4,with probability 10% each.

Classes 6 to 7 are produced according to instances presented by Berkey and Wang [19]:

- Class 6: $w_i$ , $h_i$ and $l_i$ uniformly random in [1,10] and $W = H = L = 10$.
- Class 7: $w_i$ , $h_i$ and $l_i$ uniformly random in [1,35] and $W = H = L = 40$.
- Class 8: $w_i$ , $h_i$ and $l_i$ uniformly random in [1,100] and $W = H = L = 100$.

This set of instances can be reproduced by the instance generator available at www.diku.dk/~pisinger/codes.html.

Since the solution to RMP for all of instances are non-integer, we apply branching to attain integer solution. D_FH with limit value 0.6 leads to good integer solutions for most of the instances, but infeasible solutions for maximum of four out of ten problem instances, in each class and instance size. For the instances with an infeasible solution, all fixed variables in the branching are released and the current master problem is solved to integrality by the CPLEX solver. The total execution time for each

**Table 1.** Results for the CG technique and TS$^2$PACK

| Class | Vehicle size | Number of items | CG | TS$^2$Pack (1000 seconds) | LB | CPU time |
|---|---|---|---|---|---|---|
| 1 | 100×100 | 50 | 13.4 | 13.4 | 12.9 | 15 |
| | | 100 | 26.6 | 26.7 | 25.6 | 100 |
| | | 150 | 36.3 | 37 | 35.8 | 500 |
| | | 200 | 50.7 | 51.1 | 49.7 | 700 |
| 2 | 100×100 | 50 | 13.8 | - | 13 | 10 |
| | | 100 | 25.5 | - | 25.1 | 70 |
| | | 150 | 36.4 | - | 35.7 | 500 |
| | | 200 | 49.3 | - | 48.1 | 800 |
| 3 | 100×100 | 50 | 13.3 | - | 12.7 | 20 |
| | | 100 | 25.9 | - | 24.7 | 60 |
| | | 150 | 37.4 | - | 36.4 | 500 |
| | | 200 | 49.7 | - | 48.6 | 800 |
| 4 | 100×100 | 50 | 29.4 | 29.4 | 29 | 4 |
| | | 100 | 59 | 58.9 | 58.5 | 11 |
| | | 150 | 86.8 | 86.8 | 86.4 | 30 |
| | | 200 | 118.88 | 118.88 | 118.3 | 60 |
| 5 | 100×100 | 50 | 8.3 | 8.3 | 7.6 | 11 |
| | | 100 | 15 | 15.2 | 14 | 50 |
| | | 150 | 20 | 20.1 | 18.8 | 500 |
| | | 200 | 27.1 | 27.4 | 26 | 500 |
| 6 | 10×10 | 50 | 9.9 | 9.8 | 9.4 | 10 |
| | | 100 | 19 | 19.1 | 18.4 | 150 |
| | | 150 | 29.2 | 29.2 | 28.5 | 300 |
| | | 200 | 37.4 | 37.7 | 36.7 | 1000 |
| 7 | 40×40 | 50 | 7.4 | 7.4 | 6.8 | 10 |
| | | 100 | 12.4 | 12.3 | 11.5 | 100 |
| | | 150 | 15.5 | 15.8 | 14.4 | 1000 |
| | | 200 | 23.5 | 23.5 | 22.7 | 1000 |
| 8 | 100×100 | 50 | 9.2 | 9.2 | 8.7 | 12 |
| | | 100 | 18.9 | 18.8 | 18.4 | 120 |
| | | 150 | 23.5 | 23.9 | 22.5 | 1000 |
| | | 200 | 29.4 | 30 | 28.2 | 1000 |
| Total Vehicles | | Classes1,4-8 | 726.78 | 729.8 | 708.8 | |

instance including column generation time, branching or CPLEX solver time is less than 1000 seconds. The branching rules based on Ryan and Foster [18] are not only time-consuming for the large-sized instances, but also the solutions are not superior to those resulting from D_FH branching and the CPLEX solver. For the small-sized instances the solution quality is similar. Therefore, we just present results produced by D_FH branching and the CPLEX solver.

We compare the heuristic CG technique with the two-level metaheuristic tabu search ($TS^2PACK$) developed by Crainic et al. [2]. The authors mentioned that the $TS^2PACK$ metaheuristic outperforms other techniques for solving the three-dimensional bin packing problem. Computational results are summarized in Table1. The class of instance, the vehicle size and the number of items are presented in columns 1, 2 and 3, respectively. Columns CG and $TS^2PACK$ correspond to the mean number of vehicles over ten instances resulting from the CG technique and the $TS^2PACK$ heuristic. It should be noted that $TS^2PACK$ is stopped after 1000 seconds for each instance. Since the properties of classes 2 and 3 are the same as class 1, the results for these classes are not considered for the $TS^2PACK$ heuristic in Crainic et al. [2]. Column LB shows the lower bound for the mean number of vehicles over ten instances resulted by Boschetti [5]. Since there are ten different instances for each class and instance size, the last column shows the maximum execution time in seconds for each instance. Note that the execution time for some instances is much less than this value. It is seen that the maximum computational time for each instance is 1000 seconds.

A comparison of results in column 3 with column 4 indicates that the quality of the solutions obtained by the CG technique is the same or better than those produced by the $TS^2PACK$ heuristic with a maximum computational time of 1000 seconds for each instance in both algorithms. The performance of the CG technique is better than the $TS^2PACK$ heuristic for the large-sized instances. The overall results show a total gap of 3% between CG and $TS^2PACK$.

## 8 Conclusion

In this work, we presented a column generation based heuristic algorithm for the three-dimensional vehicle loading problem. To reduce the overall execution time of the algorithm, heuristic pricing and branching are applied. The computational results show that solutions obtained through this algorithm are better than any solution obtained by meta-heuristics proposed in the literature. This method as a flexible technique can solve quite large instances better than other methods in a reasonable time. A second contribution of this work is modifying the definition of extreme points used in the best existing constructive heuristic for the 3D-BPP.

## References


1. Crainic, T., Perboli, G., Tadei, R.: Extreme Point-Based Heuristics for Three-Dimensional Bin Packing. INFORMS Journal on Computing 20(3), 368–384 (2008)
2. Crainic, T., Perboli, G., Tadei, R.: TS 2 PACK: A two-level tabu search for the three-dimensional bin packing problem. European Journal of Operational Research 195, 744–760 (2009)
3. Pisinger, D.: Heuristics for the container loading problem. European Journal of Operational Research 141, 382–392 (2002)
4. Iori, M., Martello, S.: Routing problems with loading constraints. Top 18, 4–27 (2010)
5. Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Operations Research 48, 256–267 (2000)
6. Den Boef, E., Korst, J., Martello, S., Pisinger, D., Vigo, D.: Erratum to "The three-dimensional bin packing problem": Robot-packable and orthogonal variants of packing problems. Operations Research 53(4), 735–736 (2005)
7. Fekete, S., Schepers, J.: A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: Burkard, R.E., Woeginger, G.J. (eds.) ESA 1997. LNCS, vol. 1284, pp. 144–156. Springer, Heidelberg (1997)
8. Boschetti, M.: New lower bounds for the finite three-dimensional bin packing problem. Discrete Applied Mathematics 140, 241–258 (2004)
9. Gehring, H., Bortfeldt, A.: A genetic algorithm for solving the container loading problem. International Transactions in Operational Research 4, 401–418 (1997)
10. Lodi, A., Martello, S., Vigo, D.: Heuristic algorithms for the three-dimensional bin packing problem. European Journal of Operational Research 141, 410–420 (2002)
11. Gilmore, P., Gomory, R.: A linear programming approch to the cutting-stock problem (part I). Operations Research 9, 849–859 (1961)
12. Gilmore, P., Gomory, R.: Multistage cutting-stock problems of two and more dimensions. Operations Research 13, 90–120 (1965)
13. Oliveira, J., Ferreira, J.: A faster variant of the Gilmore and Gomory technique for cutting stock problems. JORBEL- Belgium Journal of Operations Research, Statistics and Computer Science 34(1), 23–38 (1994)
14. Monaci, M., Toth, P.: A set covering based heuristic approach for bin-packing problems. INFORMS Journal on Computing 18, 71–85 (2006)
15. Pisinger, D., Sigurd, M.: Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. INFORMS Journal on Computing 19, 36–51 (2007)
16. Puchinger, J., Raidl, G.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research 183, 1304–1327 (2007)
17. Chen, C., Lee, S., Shen, Q.: An analytical model for the container loading problem. European Journal of Operational Research 80, 68–76 (1995)
18. Ryan, D., Foster, B.: An integer programming approach to scheduling. In: Wren, A. (ed.) Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling, Amsterdam, The Netherlands, pp. 269–280 (1981)
19. Berkey, J., Wang, P.: Two dimensional finite bin packing algorithms. Journal of the Operational Research Society 38, 423–429 (1987)

# Author Index