Integrating Workforce Scheduling Aspects into Daily Operations at Trailer Terminals with a Special Focus on Loading and Unloading of Trucks

Giorgi Tadumadze

Doctoral thesis
submitted in fulfilment of the requirements for the
degree of doctor rerum politicarum (Dr. rer. pol.)

at the Faculty Rechts- und Wirtschaftswissenschaften
of the Technische Universität Darmstadt

First assessor: Prof. Dr. Simon Emde
Second assessor: Prof. Dr. Christoph Glock
Darmstadt, 2020

Tadumadze, Giorgi
Integrating Workforce Scheduling Aspects into Daily Operations at Trailer Terminals with a Special Focus on Loading and Unloading of Trucks
Darmstadt, Technische Universität Darmstadt,
Year thesis published in TUprints: 2021
Date of the viva voce: 15.12.2020

Acknowledgments

This dissertation was written at the Chair of Management Science & Operations Research of the Technische Universität Darmstadt. During this long but exciting journey, I have received tremendous assistance from various people, without whom I would not be able to conduct this work. I would like to take a moment to thank all of them for their valuable support.

First of all, I would like to thank my supervisor, Prof. Dr. Simon Emde, who gave me the opportunity to be a team member of his newly-established chair and the privilege to be his first PhD student. His continuous support during my entire doctoral study is invaluable. His highly competent guidance helped me at every stage of the research and writing process. I have learned a lot from him during this time not only on an academic and professional level but also personal. Furthermore, I would like to express my deepest gratitude to Prof. Dr. Christoph Glock for serving as the second reviewer of my dissertation and for all his valuable advice especially at the final stage of this journey.

My sincere thanks go to Prof. Dr. Nils Boysen. It is a great honor for me to have worked together on the joint publications in academic journals, which are parts of this dissertation. Besides, it is worth noting that exactly his lectures at the Friedrich-Schiller-Universität brought me to the path of operations research and inspired me to pursue my PhD in this field. I also owe many thanks to Dr. Felix Weidinger and Heiko Diefenbach for the fruitful collaboration and their immense contribution to our publications.

I also thank the members of my PhD committee, Prof. Dr. Michael Neugart, Prof. Dr. Ralf Elbert, and Prof. Dr. Alexander Kock as well as Jan Phillipp Müller for their time and insightful comments on the dissertation defense.

Likewise, I am thankful to my (former) colleagues at the chair of Management Science & Operations Research, Martin Wirth and Lukas Polten for their endless support, as well as the fruitful discussions on academic and non-academic topics. My special thanks

go to Dr. Fabian Beck for his valuable feedback on the dissertation defense talk the day before the actual defense and for all the fun-times during these years.

Finally, I am very grateful to my family and friends, who always stood by me throughout these years. My special thanks go to my cousin Avto, and his wife, Mariam, for their inestimable help in the final stage. Last but not least, I want to thank my beloved Itzi for her unconditional love and understanding, which helped me through the dark times. Without her being next to me, I am sure, I would not have made it through the PhD degree.

German abstract

Die vorliegende kumulative Dissertation besteht aus vier wissenschaftlichen Artikeln, von denen drei bereits in wissenschaftlichen Zeitschriften veröffentlicht wurden und einer sich in dem Begutachtungsprozess für Veröffentlichung in einer weiteren internationalen Fachzeitschrift befindet. Alle vier Artikel beschäftigen sich mit den kombinatorischen Optimierungsproblemen, die durch Integration der Personalfaktoren in verschiedenen operativen Planungsproblemen im Bereich der Umschlaglogistik entstehen. Insbesondere sind alle betrachteten Planungsprobleme mit Ent- und Beladeprozessen der Lastkraftwagen (LKWs) an riesigen Umschlagterminals verbunden, die wichtige Knotenpunkte für viele moderne Distributionsnetzwerke darstellen. Typischerweise werden an solchen riesigen Umschlagterminals täglich hunderte LKWs koordiniert, indem die Güter von ankommenden LKWs entladen, umsortiert, möglicherweise zwischengelagert und in ausgehende LKWs geladen werden. Dabei verfügt ein Terminalgebäude i. d. R. über mehrere Ladetore, an denen LKWs angedockt werden, während die Waren aus einem Anhänger eines einkommenden LKWs entladen bzw. in einen Anhänger eines ausgehenden LKWs geladen werden. Beispiele für solche Umschlagterminals sind große Distributionszentren oder sogenannte Cross-Docks (z. B. in der Lebensmittel- oder Automobilindustrie), Hub-Terminals in der Postindustrie usw. Eines der am intensivsten untersuchten Probleme bei der Betriebsplanung solcher Umschlagterminals bezeichnet man als Truck Scheduling Problem, welches die LKWs den Toren und jeweiligen Abfertigungszeitfenstern zuweist.

Obwohl die Ent- und Beladevorgänge an den Laderampen i. d. R. eine Beteiligung menschlicher Arbeitskräfte voraussetzen, hat die Berücksichtigung der Personalplanungsaspekte in den Truck Scheduling Problemen bisher nur geringe Aufmerksamkeit in der wissenschaftlichen Literatur gefunden. Der Schwerpunkt dieser Dissertation liegt auf der Integration der Personalaspekte in Truck Scheduling Probleme. Die in unterschiedlichen Artikeln vorgestellten Optimierungsprobleme unterscheiden sich in räumlichen Aspekten, d. h. in den Örtlichkeiten im Umschlagterminal, in denen die jeweiligen Aktivitäten stattfinden. Sie umfassen Aktivitäten an den Laderampen sowohl im Wareneingang als

auch im Warenausgang sowie Vorgänge innerhalb des Hofs des Terminals (d. h. außerhalb des Terminalgebäudes). Darüber hinaus unterscheiden sich die vier Aufsätze auch im Hinblick auf die Methoden, die zur Lösung der Planungsprobleme eingesetzt werden.

Das in dem ersten Artikel betrachtete Planungsproblem beschäftigt sich mit dem Truck Scheduling Problem im Wareneingang. Insbesondere wird ein neuartiges integriertes Personal- und LKW-Planungsproblem untersucht, in dem ankommende LKWs mit beschränkten Zeitfenstern (definiert durch fixe Ankunfts- und Abfahrtszeiten) an Terminaltoren von einer variierenden Anzahl an Logistikmitarbeitern entladen werden können. Dabei hängt die Dauer eines Entladevorgangs nicht nur von der spezifischen LKW-Ladung ab, sondern auch von der Anzahl der Mitarbeiter, die die Waren aus dem LKW-Anhänger gemeinsam entladen. Der Großteil bestehender Ansätze in der Literatur ignoriert Personalplanungsaspekte in dem Truck Scheduling Problem; es wird von festen Abfertigungszeiten für jeden LKW ausgegangen und somit angenommen, dass jeder Laderampe eine feste Anzahl an Mitarbeitern im Voraus zugewiesen wird. Es besteht jedoch ein zusätzlicher Freiheitsgrad darin, je nach Dringlichkeit dem jeweiligen Ladevorgang eine variierende Anzahl der Mitarbeiter zuzuordnen. Eine höhere Anzahl von Mitarbeitern beschleunigt die Abfertigung des Vorganges, sodass es sinnvoll ist, Personal-Zuordnung und Truck Scheduling Problem ganzheitlich zu lösen. Das sich daraus ergebende ganzheitliche Planungsproblem wird für zwei repräsentative Anwendungsfälle des Truck Scheduling Problems untersucht: für ein konventionelles Distributionszentrum und für eine Cross-Dock Umgebung, die sich voneinander in ihren Zielsetzungen unterscheiden. Bei Distributionszentren ist die Zielsetzung des betrachteten Optimierungsproblems, die gesamte Flow-Time (d. h. die gesamte Zeit, die ein LKW am Terminal verbringt, inklusive der Wartezeit vor dem tatsächlichen Entladevorgang) zu minimieren. Im Kontrast dazu versucht das Optimierungsproblem in der Cross-Docking Umgebung, die ankommenden LKWs an den Entladetoren so zu planen, dass die entladene Ware in entsprechende ausgehende LKWs möglichst termingerecht umgeladen wird.

Nach der formellen Darstellung der entstandenen Planungsprobleme wird für jede Problemvariante ein gemischt-ganzzahliges lineares Optimierungsmodell entwickelt. Darüber hinaus werden die Optimierungsprobleme als sogenanntes Intervall Scheduling Problem umformuliert und darauf basierend unterschiedliche heuristische Lösungsverfahren entworfen, die sich in der Erzeugung der Intervalle in der Modellaufbereitungsphase unterscheiden. In einer rechnergestützten Studie wird die Rechenleistung der vorgeschlagenen Lösungsansätze auf zufällig generierten Probleminstanzen getestet. Dabei erweist sich eines der Verfahren zur Intervallerzeugung als sehr erfolgreich, welches Probleminstanzen

mit realistischen Größenordnungen in angemessener Zeit erfolgreich löst. Des Weiteren wird der neuartige kombinierte Ansatz der Personal- und LKW-Planung mit bestehenden sukzessiven Ansätzen verglichen, in denen die Personal-Zuordnung und das Truck Scheduling nacheinander separat gelöst werden. Die Ergebnisse zeigen, dass im Vergleich zum sukzessiven Ansatz der kombinierte Personalplanungsansatz die Leistung des Truck Scheduling Problems für beide repräsentativen Anwendungsfälle erheblich verbessert.

Der Fokus des zweiten Artikels liegt auf den Aktivitäten im Warenausgang und beschäftigt sich mit der Beladung und Ablaufplanung ausgehender LKWs. Insbesondere befasst sich das in dem zweiten Aufsatz vorgestellte Planungsproblem mit folgenden Entscheidungen: Zum einen müssen die Sendungen mit variierender Größe in die ausgehenden LKWs so verladen werden, dass die begrenzte Kapazität der jeweiligen LKW-Anhänger nicht überschritten wird (LKW-Verladeproblem); zum anderen sind die einzelnen LKWs mit gewissen Ankunft- und Abfahrtzeiten zu den Ladetoren so zu planen, dass begrenzte Terminalressourcen für Ladevorgänge (Ladetore und menschlichen Arbeitskräfte) in Betracht gezogen werden (Truck Scheduling Problem). Ein solches Planungsproblem entsteht typischerweise im Tagesgeschäft der Versandlager in der Produktionsindustrie (z. B. in der Automobilindustrie), da von solchen Versandlagern jeden Tag mehrere Lastwagen in Richtung desselben Zielortes abfahren. Deshalb muss neben dem eigentlichen Truck Scheduling Problem eine Entscheidung darüber getroffen werden, welche Ladeeinheiten in welche LKWs geladen werden. In diesem Umfeld besteht der Hauptzweck solcher Versandlager darin, eine Just-in-Time oder manchmal sogar Just-in-Sequence Lieferung der fertigen Montageteile an die Produktionslinie zu ermöglichen. Daher ist die Zielsetzung des vorgeschlagenen Optimierungsproblems, die bestellten Ladeeinheiten möglichst spät, aber nicht später als zur angeforderten Zeit zu versenden.

Das entstandene Planungsproblem wird als mathematisches Optimierungsproblem formalisiert, für welches zwei verschiedene gemischt-ganzzahlige lineare Optimierungsmodelle entworfen werden. Des Weiteren wird das Optimierungsproblem als bekanntes generalisiertes Set Covering Problem umformuliert, welches als Grundlage der Spaltengenerierung für einen entworfenen exakten Branch-and-Price Algorithmus dient. In einer Reihe von numerischen Tests wird die algorithmische Leistung der vorgeschlagenen Lösungsverfahren mit neu generierten Probleminstanzen getestet und miteinander verglichen. Dabei zeigt der neuartige exakte Branch-and-Price Algorithmus eine hervorragende Leistung und ist den beiden gemischt-ganzzahligen linearen Optimierungsmodellen unter Verwendung eines kommerziellen Standardsolvers überlegen, da er in der Lage ist, den Großteil der betrachteten Probleminstanzen in kurzer Zeit exakt zu lösen. Darüber

hinaus werden einige betriebswirtschaftliche Erkenntnisse untersucht, etwa der Einfluss von Zeitfenstermanagement und Personalbestand auf die Pünktlichkeit der Lieferungen, welche als Grundlage für die mittelfristige Planung eines geeigneten Zeitfenstermanagements und Personalbestands genutzt werden können.

Der dritte Artikel beschäftigt sich mit Aktivitäten auf dem Hof eines Trailer-Terminals, d. h. außerhalb des Terminalgebäudes. Auf viele große Umschlagterminals werden die LKW-Sattelanhänger innerhalb des Hofs von speziellen Terminaltraktoren transportiert (in der englischsprachigen Literatur ist so ein Terminaltraktor oft unter dem Namen Spotter bekannt). Dadurch entsteht ein neues Planungsproblem, nämlich die Bewegung der LKW-Anhänger innerhalb des Hofs unter mehreren Spottern effizient zu koordinieren. Im dritten Artikel wird ein neuartiges Planungsproblem von Spottern betrachtet, welches in der Planungshierarchie nach der Festlegung der Ablaufpläne einzelner LKWs an den Toren (Truck Scheduling) gelöst werden muss. Eine Reihe von gegebenen Transportaufträgen ist zu einzelnen Spottern so zuzuordnen, dass gegebene LKW-Pläne zeitlich realisiert werden können. Dabei ist die Zielsetzung des betrachteten Optimierungsproblems, einen möglichst robusten Ablaufplan von Spottern zu finden, der die zeitliche Realisierung der geplanten LKW-Ladungen an den Toren auch dann ermöglicht, wenn unerwartete Verspätungen im ursprünglichen LKW-Plan entstehen. Dafür wird zwischen jedem Paar aufeinanderfolgender Transportaufträge eine Pufferzeit eingesetzt, welche möglichst groß zu halten ist, um das System vor einer Ausbreitung kleiner Verspätungen zu schützen. Es werden zwei Problemvarianten von robuster Spotter-Planung betrachtet, die sich in ihrer Zielsetzung unterscheiden. Die erste Problemvariante zielt darauf, die gewichtete Summe aller Pufferzeiten zu maximieren, wohingegen die Zielsetzung der zweiten Problemvariante die Maximierung der minimalen gewichteten Pufferzeit ist.

Für jede der beiden Problemvarianten wird ein exakter Algorithmus entworfen, welcher das entsprechende Optimierungsproblem in polynomieller Zeit optimal lösen kann. In einer Reihe an Rechenstudien erbringen beide Lösungsansätze eine sehr gute algorithmische Leistung, sodass Instanzen mit tausenden von Transportaufträgen und hunderten von Spottern in wenigen Minuten optimal gelöst werden. Darüber hinaus wird die Robustheit der vorgeschlagenen Ansätze in einer Simulationsstudie evaluiert, in der gewisse Verspätungen der LKW-Ankünfte sowie Verzögerungen von deren Ladevorgängen an den Ladetoren simuliert werden, um zu untersuchen, ob die gefundenen Lösungen tatsächlich robust gegenüber solchen Verspätungen sind. Die Ergebnisse der Simulationsstudie zeigen, dass die Anwendung eines geeigneten Robustheitsmaßes die Ausbreitung unvorhergesehener Verzögerungen erheblich reduzieren kann. Des Weiteren wird der Einfluss

der Spotter-Flottengröße auf die Leistung des Hofs untersucht. Die Ergebnisse können als Grundlage für ein mittelfristiges Planungsproblem genutzt werden, in dem ein gewisses Serviceniveau zu erreichen ist und dafür nach einer optimalen Anzahl an Spottern gesucht wird.

Im Unterschied zu den drei vorherigen Artikeln, die jeweils konkrete Anwendungsprobleme in Betracht ziehen, liegt der Schwerpunkt des vierten Artikels eher in der Algorithmik. So ist der Hauptbeitrag des vierten Artikels der Entwurf eines neuartigen exakten Algorithmus, welcher auf dem sogenannten Ansatz der kombinatorischen (oder logic-based) Benders Dekomposition basiert und zur optimalen Lösung einer allgemeineren Klasse der Maschinenplanungsprobleme angewendet werden kann. Insbesondere besteht das grundlegende Optimierungsproblem darin, eine Reihe von Aufträgen (Jobs) mit beschränkten Zeitfenstern auf eine Menge nicht zusammenhängender paralleler Maschinen zu planen und dabei eine gewisse Minimax-Zielfunktion zu minimieren. Solche Arten von Maschinenplanungsproblemen haben eine breite Anwendung für verschiedene Planungsprobleme in der Produktion und Logistik, unter anderem auch für Truck Scheduling Probleme. In diesem Zusammenhang ist ein LKW (mit Ankunfts- und Abfahrtszeit) einem Job (mit Fertigstellungs- und Fälligkeitszeit) gleichzusetzen und ein Ladetor einer Maschine, welche zu jedem Zeitpunkt höchstens einen Job verarbeiten kann. Neben dem exakten Benders Dekomposition Algorithmus wird auch ein heuristisches Lösungsverfahren entworfen, welches auf der Umformulierung des Optimierungsproblems als bekanntes generalisiertes Set Partitioning Problem basiert. Beide entwickelten Algorithmen werden für drei unterschiedliche Minimax-Zielfunktionen entworfen, nämlich für die Minimierung der Fertigstellungszeit des letzten Jobs (d. h. makespan), der größten gewichteten Abfertigungszeit (d. h. maximal weighted flow time) und der größten gewichteten Verspätung (d. h. maximal weighted lateness). Darüber hinaus werden sie für eine Reihe von problemrelevanten Erweiterungen angepasst. Die algorithmische Leistung der neuartigen exakten und heuristischen Lösungsansätze wird in einer umfangreichen rechnerischen Studie sowohl auf neu generierten Probleminstanzen als auch in der Literatur existierenden Datensätzen getestet. Der Benders-Dekomposition-Ansatz schneidet ziemlich gut ab, da er die meisten Probleminstanzen innerhalb kurzer Rechenzeit optimal löst. Auch die Heuristik leistet eine gute Arbeit, insbesondere während der Lösung der schwierigen Instanzen mit engen Zeitfenstern der Jobs.

Abstract

The cumulative dissertation at hand consists of four research papers, three of which have already been published in scientific journals. The fourth is under review to be published in another international journal. All four articles deal with the combinatorial optimization problems that arise through integrating personnel scheduling factors into operative planning at cargo handling facilities. More precisely, the considered planning problems are related to trucks' unloading and loading processes at massive transshipment terminals, crucial nodes for many modern transportation networks. Examples for such large trailer terminals can be distribution centers or cross-docks (e.g., in the food or automotive industry), hub terminals in the postal sector, etc. At such cargo facilities, hundreds of trucks are coordinated daily. In particular, goods are unloaded from incoming trucks, possibly stored for a while, consolidated, and loaded into outgoing trucks. Typically, a terminal building of such cargo facility consists of several gates (or dock doors), where trucks are docked while goods are unloaded or loaded into the trailer. One crucial planning problem at such facilities is to schedule trucks' loading and unloading processes at the terminal gates. The planning problem that determines on at which door and in which time interval each truck has to be (un-)loaded is called truck scheduling problem.

Although trailers' (un-)loading activities at the gates usually require worker's involvement, the integration of personnel planning aspects into truck scheduling problems has only received little attention in the scientific literature. The focus of this thesis is to integrate workforce planning into truck scheduling problems. The planning problems proposed in the presented articles differ in spatial aspects where the respective operations take place. They include activities at the gates in both the receiving and the shipping areas of the facility as well as activities within the terminal yard, i.e., outside the terminal building. Furthermore, as discussed below, these four papers apply different methodologies to solve the corresponding optimization problems.

The first paper focuses on inbound operations, i.e., in the receiving area of the cargo facility. We introduce a novel integrated workforce and truck scheduling problem, where

incoming trucks with strict arrival and departure times are unloaded at the dock doors by a varying number of terminal employees. The duration of the trailer's unloading depends not only on the specific truckload but also on the number of workers jointly unloading it. It is worth mentioning that most existing models widely ignore the latter phenomenon by assuming constant handling times for each truck. That represents the case where a fixed number of employees are preassigned to each dock door in advance. In practice, however, terminal managers have additional flexibility in assigning a diverse number of employees to each truck, depending on the degree of its actual urgency. Since a higher number of employees speeds up the handling time, it is reasonable to solve the personnel assignment and truck scheduling problem holistically. We examine the integrated planning approach for two representative truck scheduling settings: a conventional distribution center and a cross-docking terminal. The resulting two optimization problems differ from each other in their objectives. The former's objective is to minimize the total flow-time (i.e., the total time a truck spends at the terminal, including the waiting time before the actual unloading process). In the cross-docking environment, the goods are directly moved towards the corresponding outbound trucks without any (extended) intermediate storage in the terminal. Hence, the latter problem aims to schedule the incoming trucks at the inbound gates in such a way that the unloaded goods can be transferred to the corresponding outbound trucks on time. More precisely, the truck scheduling problem in the cross-docking setting minimizes the number of delayed shipments that cannot reach the dedicated outbound trucks before the scheduled departure time.

After formalizing the planning problems as mathematical optimization problems, we develop a mixed-integer linear optimization programming model for each setting. Moreover, we provide an alternative problem reformulation as the interval scheduling problem. Based on this, we design various heuristic solution procedures, which are different in terms of the interval generation process in the model preparation phase. We test the algorithmic performance of the proposed solution approaches using newly generated problem instances. Thereby, one of the proposed strategies proves to be very successful, obtaining (near-to-)optimal solutions in a reasonable computational time for the problem instances of realistic size. Furthermore, we compare the novel integrated truck and workforce scheduling approach with existing successive methods that solve the personnel assignment and the truck scheduling problem sequentially. The results show that compared to the existing approaches, the novel integrated planning approach can significantly improve the terminal's performance for both representative settings.

The second article focuses on operations in the shipping area and deals with the loading

and scheduling of outbound trucks. More precisely, the planning problem deals with the following decisions: assigning shipments with varying sizes to outgoing trucks considering the trailers' limited loading capacity, and scheduling the trucks' loading processes at the gates considering the given time windows and limited terminal workforce—required to load the shipments into trucks. Such a planning problem arises at cargo facilities where the goods are consolidated on the outbound side. An important example of such facilities is dispatch warehouses of part suppliers (e.g., in the automotive industry), from which multiple trucks depart towards the same destination each day. Such dispatch warehouses are primarily implemented to deliver readily usable parts to the assembly plant in a just-in-time or sometimes even just-in-sequence manner. Therefore, the objective of the proposed optimization problem is to ship the ordered parts as late as possible, but not later than their deadlines at the assembly line.

We formalize the resulting planning problem as a mathematical optimization problem and design two different mixed-integer linear programming models. Furthermore, we reformulate the optimization problem as the well-known generalized set covering problem, which we use as the basis for the column generation method, and design an exact branch-and-price algorithm. In series of numerical tests, we observe the computational performance of the proposed three solution approaches (i.e., two MILP models using the standard solver, and the exact branch-and-price algorithm) on newly generated large problem instances. Our results show that the branch-and-price algorithm outperforms both MILP models, obtaining optimal solutions for most of the considered problem instances in a short computational time. Moreover, we explore some managerial insights, such as the influence of the time window management and workforce amount on delivery punctuality. Our findings can be useful for terminal managers when designing a suitable time window management system or seeking the proper workforce amount at the tactical planning level.

The third article covers the activities within the trailer yard, i.e., outside the terminal building. Many large trailer terminals use specialized yard tractors (often called spotters or switchers) to transport semitrailers within the yard. This way, a new planning problem arises, associated with coordinating the intra-yard trailer movement among multiple spotters. In the third paper, we introduce a novel spotter scheduling problem that has to be solved in the planning hierarchy after defining the particular gate and the docking time for each trailer (i.e., after solving the truck scheduling problem). Our robust spotter scheduling problem assigns a set of given trailer movements to spotters such that the predefined truck schedules can be realized on time—even if unexpected delays occur in

the trailer yard. We insert a buffer time between any pair of consecutive transportation requests and aim to keep it as large as possible to protect the system from propagating small delays. We propose two problem variants of the robust spotter scheduling problem, which differ in their objectives. The first problem version maximizes the weighted sum of buffer times (max-sum), and the second one the minimum weighted buffer time (max-min).

For each problem variant, we develop an exact algorithm that finds the optimal solution in polynomial time for the related optimization problem. Our approaches show an outstanding algorithmic performance in computational tests, optimally solving instances with thousands of transport requests and hundreds of spotters in short computational time. Besides, we evaluate the robustness of the proposed approaches in a simulation study. Specifically, we simulate some uncertain events at the terminal, such as delays in truck arrivals and handling processes at the loading gates, and examine whether the solutions found by the proposed approaches are indeed robust. The results show that applying a suitable robustness measure (i.e., max-min) can significantly reduce the propagation of unforeseen delays. Finally, we also observe the interrelation between the two objectives and explore the impact of the spotter fleet size on the yard performance. The latter study's findings can be a useful insight for mid-term planning when seeking an appropriate number of spotters to achieve the desired service level.

In contrast to the three previous papers, each dealing with the specific application problems, the fourth paper primarily focuses on algorithmics. Its main contribution is a novel exact algorithm, based on the combinatorial (or logic-based) Benders decomposition scheme, that can be applied to optimally solve a broader class of scheduling problems. In particular, the fundamental optimization problem is to schedule a set of jobs with release dates and deadlines on a set of unrelated parallel machines minimizing some min-max objective. Such types of machine scheduling problems have a wide range of applications for various scheduling problems in production and logistics, including truck scheduling problems. In the truck scheduling context, a job (with the release date and the deadline) corresponds to a truck (with the arrival and the departure time), and a machine to a gate that can process at most one job at a time. Apart from the exact Benders decomposition algorithm, we propose a heuristic algorithm that relies on a reformulation of the optimization problem as the well-known generalized set partitioning problem. We design the algorithms for three different minimax objective functions: minimizing the completion time of the last job (i.e., makespan), the maximum weighted flow time, and the maximum weighted lateness. Besides, we reveal how to modify the

proposed algorithms for several problem-relevant extensions. We test the computational performance of the provided solution techniques in an extensive computational study on both newly generated random problem instances and existing instance sets available in the literature. The novel exact Benders decomposition approach performs quite well, solving most problem instances to proven optimality within short computational time. The proposed heuristic also does a good job, successfully providing near-optimal solutions in a reasonable time for particularly tricky problem instances with tight time windows.

Contents

List of Figures

List of Tables

List of Abbreviations

AWL	Average weighted lateness
B&BC	Branch & Benders cut
B&P	Branch & Price
BAP	Berth allocation problem
BDP	Bounded dynamic programming
CD	Cross-docking
CP	Constraint programming
CPLEX	IBM ILOG CPLEX Optimizer
CPU	Central processing unit
DC	Distribution center
DRL	Dock-related lateness
EDD	Earliest due date
ERD	Earliest release date
EU	European Union
GB	Gigabyte
GHz	Gigahertz
GPS	Global Positioning System
GSC	Generalized set covering

GSPP	Generalized set partitioning problem
h	Hour
ILP	Integer linear programming
ITWS	Integrated truck and workforce scheduling
ITWS-CD	Integrated truck and workforce scheduling in cross-docking context
ITWS-CD-I	Interval scheduling for integrated truck and workforce scheduling in cross-docking context
ITWS-DC	Integrated truck and workforce scheduling in distribution center context
ITWS-DC-I	Interval scheduling for integrated truck and workforce scheduling in distribution center context
JIT	Just-in-time
km/h	Kilometer per hour
L	Large
LB	Lower bound
LP	Linear programming
m	Meter
M	Medium
m/s	Meter per second
MILP	Mixed integer linear programming
MILP-LO	Linear ordering mixed-integer linear programming
MILP-TI	Time-indexed mixed-integer linear programming
min	Minute
MIP	Mixed integer programming

MIP-OP	Original mixed-integer programming
NP	Nondeterministic polynomial time
NTC	National Transport Commission Australia
NTI	Nova Technology International
OEM	Original equipment manufacturer
OTSLP	Outbound truck scheduling and loading problem
PC	Personal computer
RAM	Random-Access Memory
RMP	Restricted master problem
RSSP	Robust spotter scheduling problem
s	Second
S	Small
SKC	Selective k-colorability problem
TRL	Truck-related lateness
TSP	Traveling salesman problem
UB	Upper bound
UNCTAD	United Nations Conference on Trade and Development
VB.NET	Visual Basic.NET
VRS	Verkehrsrundschau (a german professional journal)
XS	Extra small

Chapter A.

Introduction and Overview

A.1. Introduction

Apart from the tremendous increase in freight transport in many regions of the world during the past decades (e.g., in the western countries, see Blanquart and Burmeister, 2009), the road transport remains the most widespread transportation mode among the three inland transport modes—road, rail, and inland waterways. According to Eurostat (2020), the road transport performance (measured in tonne-kilometres) accounted for 75.3% of the total EU inland freight transport in 2018. The same proportion in Germany is reported to be 71.4% in 2019, which is expected to increase further by 2023 (Statista, 2020).

The above-mentioned growth of the road traffic has led to congestion not only at the transport network links (i.e., roads), but also at its nodes (i.e., cargo facilities), where freight is unloaded from incoming and loaded into outgoing trucks. Examples of such cargo handling facilities may be a source and a sink node of the distribution network (e.g., massive manufacturing plants in the automotive industry, see Battini et al., 2013), or intermediate nodes, which we discuss below in more detail.

Such intermediate nodes differ in their types and purposes in the transportation network. While goods are temporarily stored for some time at conventional distribution centers, the relatively modern cross-docking concept does not consider any (extended) intermediate cargo storage in a terminal. Instead, after unloading the inbound trucks, the shipments are directly moved towards the outbound trucks and loaded into their trailers. This way, they leave the terminal promptly to the next destination in the distribution chain (Boysen and Fliedner, 2010). The absence of the storage buffer inside the

cross-docking terminal enables a significant reduction in handling and inventory costs. On the downside, cross-docking creates new decision problems related to the precise synchronization of the inbound and outbound operations, which is a complex decision task faced by many terminal managers (Buijs et al., 2014). The cross-docking is part of many modern supply chain networks in diverse industries, such as the automotive industry (Battini et al., 2013, Berghman and Leus, 2015), retail grocery (Buijs et al., 2016), food, and fresh produce industry (Boysen, 2010, Zaerpour et al., 2015, Bodnar et al., 2017) etc. For the state-of-the-art of cross-docking, addressing a wide range of related decision problems, we refer to Van Belle et al. (2012). Besides, several other surveys with various foci have been published on planning problems at cross-docks (e.g., Boysen and Fliedner, 2010, Stephan and Boysen, 2011a, Shuib and Fatthi, 2012, Buijs et al., 2014, Ladier and Alpan, 2016).

Buijs et al. (2014) review synchronization problems in cross-docking networks and classify three typical network configurations with a single cross-dock, which define its type and purpose in the distribution chain:

- A *many-to-few configuration* is common for the manufacturing industry (e.g., in the automotive industry) and is dedicated to the consolidation of shipments from multiple suppliers to one or a few customers. Hence, consolidation mainly takes place on the outbound side. In this configuration, a cross-dock facility is often allocated close to the receiver(s) (e.g., in the supplier park of a manufacturer, see Wang and Chen, 2019)), enabling timely delivery of readily usable parts for the final assembly at the manufacturing plant.

- A *few-to-many configuration* is usually encountered in retail distribution, where large loads from a few distribution centers are split into smaller shipments for plenty of retail stores. In this setting, cross-docking is often implemented in combination with the more conventional warehousing strategy. Such a network mainly intends to improve delivery service levels and reduce transportation costs (Buijs et al., 2016).

- A *many-to-many configuration* can be observed at supply chains with a wide variety of assortment and abundant suppliers and receivers, e.g., in the postal service industry (e.g., Boysen et al., 2017, Fedtke and Boysen, 2017). At parcel distribution centers, small shipments from multiple incoming trucks are consolidated in full truckloads to be jointly transported with fully loaded outgoing trucks later on. Hence, economies in transportation are gained at both the inbound and outbound

sides.

Regardless of their type and purpose in the transportation network, most trailer transshipment terminals have many common characteristics and share plenty of decision problems during their life cycle. In the following, we give a short overview of some in the literature intensively discussed decision problems that arise at such terminals:

A critical strategic decision, which has to be made on a long-term planning horizon, is to locate the facility in the transportation network, often referred to as *location planning* (Van Belle et al., 2012). The construction costs play a vital role in the choice of the facility's location (Domschke et al., 2018). Therefore, large trailer terminals are often located in rural areas, where the land price is not high. The reason for this is that such trailer terminals typically need a lot of space. A facility typically consists of a spacious terminal building with a storage area and multiple gates (or dock doors), where trailers are docked during the (un-)loading operations, and a large parking lot, where trailers are parked before or after the processing at the gates. For a general overview of the location planning problems, we refer to Brandeau and Chiu (1989). More specific location planning problems for intermediate network nodes, i.e., hub terminals, warehouses, cross-docks, or other kinds of cargo facilities, are reviewed by Klose and Drexl (2005) and Campbell and O'Kelly (2012).

Another essential class of strategic planning problems, consisting of several crucial decisions, is *layout planning*. A critical layout planning problem for many cargo handling facilities—especially for the cross-docking terminals—is related to the shape of the terminal building. Bartholdi and Gue (2004) study this problem and analyze the performance of different shaped cross-docking buildings (e.g., I, L, T, X-shaped), depending on the number of the terminal dock doors. In an investigation into the layout planning problem for rectangular unit-load cross-docks, Carlo and Bozer (2011) derived the optimal ratio between the terminal's length and width, minimizing the expected rectilinear travel distance. Layout planning also covers intra-terminal layout decisions, such as designing staging areas inside the terminal building (Vis and Roodbergen, 2011) and organizing sortation conveyors at parcel distribution centers (Fedtke and Boysen, 2017).

An important decision in this regard is the arrangement of inbound and outbound areas within the terminal building. At this point, it is necessary to distinguish between the different service modes, classified by Boysen and Fliedner (2010), which defines the degree of freedom in assigning inbound and outbound trucks to the gates:

- *Exclusive-mode* considers that each gate is exclusively dedicated to either inbound or outbound operations. The dock doors that serve the inbound trucks are also called *strip doors*, and the ones exclusively reserved for the outbound trucks are referred to as *stack doors* (Van Belle et al., 2012). For practical purposes, all strip doors are usually located along one side of the terminal and all stack doors on the other side. As a result, the terminal building has the spatially separated *receiving* and *shipping* areas (Van Belle et al., 2012). Stephan and Boysen (2011b) denote such a setting as "vis-à-vis policy" and compare it with a more flexible "mixed policy", where the strip and stack doors can be facultatively assigned all around the cross-dock.

- *Mixed-mode* considers that dock doors can serve both inbound and outbound operations. Despite its flexibility, this mode is less popular in the industry practice and scientific literature (Ladier and Alpan, 2016). However, recently the mixed-mode has been gaining attention in academia (e.g., Bodnar et al., 2017, Rijal et al., 2019).

- *Combination-mode* occurs when one subset of gates operates in exclusive-mode and the other in mixed-mode.

One of the most significant challenges in the daily operations at large trailer terminals is allocating trucks to the gates, where they are loaded or unloaded. In this regard, two main research branches emerge from the studies discussed so far: *door assignment* and *truck scheduling*. The distinction between these two areas is often somewhat ambiguous, and no standard notation has been used to classify different optimization problems (Ladier and Alpan, 2016).

Door assignment is sometimes referred to as *truck-to-dock assignment* (Ladier and Alpan, 2016), *dock door assignment* (Shuib and Fatthi, 2012), *cross-dock planning* (Buijs et al., 2014), or *destination assignment*, in case of considering only outbound trucks with fixed destinations (Boysen and Fliedner, 2010). The door assignment decides on the arrangement of trucks at the gates such that some performance measure is optimized. A typical objective of the door assignment problem is to minimize the total travel distance of the transferred goods inside the terminal building. The first door assignment problem was proposed by Tsui and Chang (1990), who then designed an exact branch and bound algorithm for this problem in the following study (Tsui and Chang, 1992). Since then, there has been published a large number of studies (e.g., Bartholdi and Gue, 2000, Oh

et al., 2006, Yu et al., 2008, Cohen and Keren, 2009) that deal with this issue (see Shuib and Fatthi, 2012, for a review on door assignment problems in cross-docking).

Boysen and Fliedner (2010) draw our attention to distinctive categories of door assignment problems, denoted as *destination assignment problem*, where the shipping doors are exclusively dedicated to particular destinations. In this setting, the inbound and outbound operations are usually planned successively. The destination assignment problem (i.e., outbound operations) is typically scheduled for a mid-term planning level (one to several months). In contrast, the inbound operations are planned at the operational planning level over a shorter planning horizon (typically one day or even one working shift). Zenker and Boysen (2018) observe a destination assignment problem at parcel distribution centers considering more destinations than the shipping doors. Hence, they analyze the decision problem on which destinations should share the gates.

In their review, Van Belle et al. (2012) differentiate door assignment and truck scheduling problems with the following criteria: *door assignment* considers that the number of trucks is equal to the number of doors. Hence, each door receives one truck and vice versa, so that the time horizon is not taken into account. If there are more trucks than the gates, time aspects have to be considered, and the corresponding problem becomes *truck scheduling*. In other words, in contrast to the door assignment problem, the truck scheduling problem takes time dimensions into account. Therefore, a truck scheduling problem is usually more complex than a door assignment problem. Truck scheduling decides on *where* (i.e., at which door) and *when* (i.e., in which time interval) to process each truck. The common objective of the truck scheduling problem is related to the timely processing of all trucks. The first truck scheduling problem was proposed for a parcel hub terminal by McWilliams et al. (2005). Since then, a considerable amount of literature has been published on truck scheduling problems (see Boysen and Fliedner, 2010, for an exhaustive review of truck scheduling problems at cross-docks).

Ladier and Alpan (2016) categorize other problem classes related to the allocation of trucks at the doors, which can be classified somewhere between the door assignment and the truck scheduling problems. Specifically, they explicitly distinguish the *truck-to-door sequencing* problem, which determines the sequence of trucks processed at each door without taking any time aspects into account. Moreover, the authors distinguish *truck sequencing* problems, which do not take space dimensions between the doors into account and only determine the order in which the trucks arrive at the gates.

Despite the intensive study in this field, very little attention has been paid to the role

of workforce scheduling into truck scheduling and related problems. Terminal employees play a crucial role in trucks' loading and unloading activities from both sides of the gate. Inside the terminal building, workers execute several manual tasks when loading and unloading trailers at the gate. Outside the terminal building, the intra-yard trailer transportation is also often conducted by terminal employees. This thesis focuses on integrating workforce-related aspects into daily operations at trailer terminals. More precisely, we introduce some novel scheduling problems associated with the trucks' loading and unloading operations, considering personnel scheduling aspects. The proposed scheduling problems cover inbound and outbound operations inside the terminal building as well as activities within the yard. After introducing the new planning problems, we formalize them as mathematical optimization problems and design appropriate solution methods.

The rest of the thesis is organized as follows: in this chapter, we first describe the relevant daily operations at typical large trailer terminals (Section A.2). We then give a brief overview of the proposed novel planning problems and the applied solution techniques (Section A.3). Later, the following Chapters B, C, D and E discuss each planning problem one by one in more detail. Finally, Chapter F concludes the thesis.

A.2. Daily operations at trailer terminals

This section describes typical daily operations at trailer terminals associated with the trailer loading and unloading transactions. Note that due to the organizational and technical diversities of different cargo facilities, the particular process steps vary from terminal to terminal. In the following, we discuss a typical process flow, which covers the essential operations at most trailer terminals.

On the inbound side, fully or partially loaded incoming trucks arrive at the terminal, delivering freight directly from a supplier or another intermediate node (e.g., other depot or hub terminal, freight airport, seaport, freight yard, etc.). Typically, the terminal management is aware of the trucks' arrival time by the beginning of the workday, since the trucks' processing time windows are usually booked beforehand (Karaenke et al., 2019). Besides, the trucks' locations can be continuously tracked via a GPS navigation system. When a truck arrives at the terminal, its arrival is registered at the entry gate, e.g., by a yard operator, who assigns the trailer to a particular parking space in the yard. Then, depending on whether the intra-yard trailer transportation is executed by local

shunters or delivery road truckers, there can be two alternative process flows:

- If road truck drivers are responsible for the intra-yard trailer transportation, the trucker first parks the trailer at the assigned parking position. The trailer stays there until it is called up by a yard dispatcher, informing the gate number and the processing time window, e.g., via pager handed over at the terminal entry during registration (Berghman and Leus, 2015). After that, the trucker transports the trailer to the dedicated gate and docks it, precisely backward-maneuvering until the trailer reaches the gate's loading platform. In this scenario, the trucker stays at the terminal, waiting until the unloading completes. Once the trailer's processing is completed, the trucker can depart to the next destination in the logistics chain.

- If the terminal uses the specialized terminal tractors (also called spotters or switchers, see Yano et al., 1998, Berghman and Leus, 2015) for the intra-yard trailer transportation, the delivery truck is uncoupled from the semitrailer at a particular location in the yard, instructed by a yard operator, e.g., at the parking position. The uncoupling step is registered in the yard information system by a yard operator, who can monitor the parking lot via a video surveillance system. From that moment, the detached semitrailer is available to be picked up by a spotter to bring it to the appropriate terminal gate. The delivery trucker is released after the uncoupling step and does not have to wait to complete the trailer's further processing (Yano et al., 1998). Thus, the road trucker may directly leave the terminal or be coupled with an other (empty or full) trailer to transport it towards the next destination in the distribution chain.

In both settings explained above, the docking step at the gate is followed by the trailer unloading stage. In particular, the truckload is unloaded from the trailer and moved inside the terminal building to a (temporary) storage area. Once the trailer is docked at the gate, a terminal employee operating at that door prepares the trailer unloading step. This step consists of manual activities, such as breaking the trailer seal, opening the trailer door, removing the load restraining belt, etc. After that, the actual unloading step starts, which may vary from facility to facility depending on the cargo type and the applied material handling equipment. We can differentiate between the following unloading modes:

- If the freight is palletized, terminal workers usually utilize a forklift truck or a manual pallet jack. This way, they move entire pallets from the trailer to an

intermediate storage area inside the building, e.g., a staging area directly next to the dock door (Bartholdi and Gue, 2000).

- In parcel distribution centers, where the truckload consists of smaller loose goods, a telescopic conveyor may be applied (Fedtke and Boysen, 2017). A logistics worker pulls the telescopic conveyor directly in front (and later into) the trailer and puts parcels individually onto the conveyor belt. The telescopic conveyor can be extended or retracted to the desired level. As a result, it enables workers to always directly place each package onto the conveyor belt, without extra walking and carrying the shipments inside the trailer (Boysen et al., 2017).

- In recent years, novel automatic cargo batch loading and unloading systems have been developed. These are fully-automated platforms, installed at the dock, that can elevate the complete truckload in a single effort. The application of such a system can effectively reduce the trucks' processing time at gates and prevents accidents during the (un-)loading transactions (Lee et al., 2014, Kang et al., 2017). However, it is noteworthy that installing such platforms requires colossal investments and due to their infancy, they cannot be yet widely used for a broad range of trailer and truckload types.

Once the goods have been unloaded from the trailer, a terminal employee accomplishes subsequent follow-up manual tasks, such as closing the trailer door, putting a seal, etc. After that, the trailer is ready to be undocked from the gate.

The unloaded goods are transshipped through the facility—to a temporal storage area or directly at the staging area in front of the corresponding shipping door. Intra-terminal transportation can again diversify depending on the terminal and cargo type.

- Palletized goods are usually transported *manually*. Logistics workers use manual jacks to pull single or multiple pallets—coupled with a chain—through the terminal (Bartholdi and Gue, 2000). At some terminals, instead of manual pallet jacks, forklift trucks may be used for the same purposes.

- Smaller shipments are typically transported using an *automated* system, e.g., a network of sortation conveyor belts (Van Belle et al., 2012). The individual loads are verified via scanning their barcodes, and their target points in the terminal are determined. This way, the automated conveyor can move the boxes to the target destination. An extensive survey on operational planning problems using

automated sortation conveyors is provided by Boysen et al. (2019).

- Note that a *combination* of both aforementioned transportation modes, i.e., *manual* and *automated*, is also possible (Ladier and Alpan, 2016).

The operations on the outbound side are similar but executed in the reverse order. After fetching the shipping goods to the outbound gate, the logistics workers load them into empty (or in case of the milk run distribution—partially loaded, see Hosseini et al., 2014) trailers. Note that the loading stage faces additional difficulties and requires more time than the unloading process. The trailer loading post-processing step consists of manual tasks, like restraining the load (e.g., using a tie-down lashing strap, see NTC, 2018), closing the trailer door, putting the trailer seal, etc. Finally, the loaded semitrailer is picked up either by a spotter driver, who retrieves it to the parking lot, or by a road trucker, who can directly leave the terminal.

Some particular transactions may slightly differ from the above-explained flow at the cargo facilities in real-world applications. The described procedure should primarily serve as a generic representation of the trailer terminal's process steps, which are essential for the scheduling problems discussed in this thesis.

A.3. Overview of the novel planning problems

This section gives a short overview of the scheduling problems observed in the following chapters and outlines their interrelation. All four chapters deal with novel scheduling problems related to trucks' loading or unloading at the terminal gates, considering workforce involvement. The planning problems differ in terms of the considered area where the respective activities take place. They include operations in the receiving and shipping areas of the terminal as well as activities in the yard. We discuss these problems one by one in more detail in the following chapters (Chapters B, C, D and E). Note that each of these chapters is either already published in or submitted to a scientific journal as an independent article (see Table A.1 for more information).

One of the greatest challenges many terminal managers face is coordinating the loading and unloading activities of multiple trucks at dock doors while taking the limited terminal resources into account. Almost every paper that has been written on truck scheduling assumes that dock doors are the unique critical resources during trucks' processing.

Thesis		Paper			
Chapter	Focus	Authors	Title	Journal	Status
B	Inbound truck scheduling	G.Tadumadze N. Boysen S. Emde F. Weidinger	Integrated truck and workforce scheduling to accelerate the unloading of trucks	*European Journal of Operational Research*	Published
C	Outbound truck scheduling	G.Tadumadze S. Emde	Loading and scheduling outbound trucks at a dispatch warehouse	-	Under Review
D	Yard truck scheduling	G.Tadumadze N. Boysen S. Emde	Robust spotter scheduling in trailer yards	*OR Spectrum*	Published
E	General truck scheduling as parallel machine scheduling problem	G.Tadumadze S. Emde H. Diefenbach	Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines	*OR Spectrum*	Published

Table A.1.: Overview of the chapters of the thesis and the underlying papers

The doors are indeed often the bottleneck resources of the terminal, especially at peak hours. However, the unloading and loading operations of trucks require other labor-intensive resources, namely, terminal employees. Truck's processing at a dock door consists of several manual tasks, which can be executed only by humans. Examples of such manual actions include removing/putting a trailer seal, opening/closing a trailer, putting/removing safety belt, (un-)installing telescope conveyor, (un-)packing shipments in/from the trailer, driving industrial trucks and forklifts inside and outside the terminal buildings, etc. Therefore, the truck scheduling problem is heavily interdependent with the workforce scheduling problem, which is another complex issue (Boysen and Fliedner, 2010). Most of the existing works suppose that each truck has a fixed processing time at every door, presuming that the same numbers of the terminal workers operate at each gate. However, in practice, terminal managers can assign a varying number of (un-)loaders to each truck. They can assign more workers to the truck's unloading with critical timing and, this way, shorten its processing.

So far, only a few works have paid attention to the role of workforce amount on the truck's handling time. In their work on the multi-mode resource-constrained cross-dock scheduling problem, Hermel et al. (2016) consider that trucks' processing at the gate depends on the number of workers. Specifically, the authors allow two types of task processing: "regular processing" and "crash processing". The former refers to the circumstance when a single worker executes the task. The latter reduces the task's handling time when two workers perform it. Indeed, assigning two instead of a single worker can decrease the truck's handling time. However, there may be additional flexibility to increase the number of workers even further. Similarly, Konur and Golias (2017) propose an outbound truck scheduling problem, where they consider that the truck's loading time depends on the number of workers operating at the designated door. They allow assigning a varying number of workers to each gate, who remain there during the entire planning horizon. However, sometimes it may be meaningful to dynamically adapt the workforce amount at the doors during the planning horizon—depending on the particular docked truck's priority.

In Chapter B, we propose a novel inbound truck scheduling problem where each truck's processing time depends on the number of logistics workers concurrently unloading the goods from the trailer. Moreover, we consider that the workers may switch the doors during the planning horizon. Additional workers can only accelerate the trailer's unloading step—one of the five manual steps of the truck's processing at the gate. Thus, the resulting optimization problem decides on *where* (i.e., at which door), *when* (i.e.,

in which time interval), and *how fast* (i.e., by how many workers) each truck should be unloaded. The former two decisions surely belong to the truck scheduling problem, while the latter is rather a workforce scheduling task. Thus, our novel optimization problem combines the workforce scheduling problem and the truck scheduling problem into one holistic problem. We examine the resulting *integrated truck and workforce scheduling problem* (dubbed as ITWS) for two basic settings: for conventional distribution centers and cross-docks. In distribution centers, the unloaded goods are stored in the terminal for some time. The ITWS in the distribution center context aims to minimize the trucks' total flow time. In the cross-docking environment, the goods are directly moved onto outbound trucks without any intermediate storage in the terminal. Therefore, the goal of the ITWS in the cross-docking context is to schedule trucks such that the shipments from inbound trucks can reach the outbound trailers on time. Note that this setting is suited for cross-docks with the exclusive-mode (i.e., where each door is exclusively reserved either for inbound or outbound trucks), and the inbound and outbound operations are executed separately. More precisely, the proposed model observes the exclusively inbound truck scheduling problem for the given outbound truck schedules, whereas the *destination assignment* problem is assumed to be solved in the previous planning stage.

After formalizing both versions of ITWS problems as appropriate mathematical optimization problems, we design a mixed-integer linear programming (MILP) model for each problem variant. Furthermore, we reformulate them as a so-called interval scheduling problem and, based on this, design some heuristic solution methods to solve the proposed optimization problems in a reasonable time. We test the algorithmic performance of the proposed approaches in an extensive computational experiment using newly generated ITWS problem instances. Moreover, we compare the proposed integrated planning approach with the existing successive planning approaches. More precisely, in the first stage, we assign a particular (constant or varying) workforce amount to each dock. Subsequently, we solve the truck scheduling problem for the fixed workforce assignment at each door. We compare the integrated and successive planning approaches from two different perspectives. On the one hand, we fix the available workforce amount and observe the terminal performance improvements when the integrated instead of the successive planning approach is used. On the other hand, we explore the workforce saving potential of the integrated planning approach when a certain service level must be achieved. Chapter B has been published as Tadumadze, G., Boysen, N., Emde, S., Weidinger, F. (2019): Integrated truck and workforce scheduling to accelerate the unloading of trucks. *European Journal of Operational Research* 278 (1) 343–362. The definitive publisher authenticated

version is available online at `https://doi.org/10.1016/j.ejor.2019.04.024`.

The truck scheduling problem presented in Chapter B is suited for the inbound operations, i.e., unloading cargo from incoming trucks. In contrast to the inbound transactions, the activities at the outbound side may consist of additional issues that need to be solved adequately. In particular, shipments must be assigned to outbound trucks considering the trailer's limited loading capacity. Such a decision is especially critical for cargo handling facilities in the many-to-few configuration in the manufacturing industry, where the consolidation takes place on the outbound side (Buijs et al., 2014). In the automotive industry, the original equipment manufacturers (OEMs) often purchase labor-intensive parts (e.g., seats and wiring harnesses, etc.) from far-distant suppliers, promising low procurement prices (Schwerdfeger et al., 2018). The scarce storage space at the OEM's plant makes it often impossible to build large safety stocks. Seeing that the assembly line should run smoothly, the required parts have to be shipped to the manufacturing plant in a just-in-time or sometimes even just-in-sequence manner (Emde, 2011). Therefore, part suppliers often allocate dispatch warehouses in close vicinity to the targeted manufacturing plants in a so-called *just-in-time corridor* (Schwerdfeger et al., 2018). This way, readily usable components are stored at the dispatch warehouse until the very last moment, i.e., as long as the final assembly sequence is definite. Once the final production sequence is determined, the OEM orders the required components with a strict delivery deadline. On the one hand, the shipments cannot arrive at the manufacturing plant later than the deadline, seeing that the assembly line must never run out of the necessary parts. On the other hand, due to the scarce storage space at the OEM's plant, the pieces should be dispatched from the intermediate warehouse as late as possible.

In this regard, it is advisable to rule out the late delivery via hard constraints and penalize the earliness in an objective function. This kind of optimization problem, aiming to minimize the shipments' total earliness is studied by Boysen et al. (2016). More precisely, their just-in-time vehicle scheduling problem consists of the two following decisions: assigning a given set of items to the outbound trucks considering the trucks' limited loading capacity, and planning the trucks' departure times such that the early arrival of orders is minimized. Although their model consists of a decision on each truck's departure time, it does not precisely solve the outbound truck scheduling problem at the intermediate facility. Their model does not decide when each truck must be loaded under consideration of the terminal's limited resources and the trucks' restricted time windows.

Chapter C introduces a novel *outbound truck loading and scheduling problem* (dubbed as OTSLP), which combines the truck loading problem with the truck scheduling problem, as mentioned above. More precisely, OTSLP solves the following decision problems simultaneously:

- *Truck loading problem* assigns the orders with diverse sizes to the outbound trucks with varying loading capacities,
- *Truck scheduling problem* schedules trucks' processing within the given time windows, taking the terminal's limited resources (i.e., dock doors and terminal workforce) into consideration.

Loading an order consists of tasks like fetching the parts from the storage, sorting, packaging, labeling, and placing them inside the trailer. Depending on the order's size, its preparation may require a certain workforce amount, a limited resource at the terminal. Moreover, each order is associated with a shipping deadline and a penalty cost per time unit of earliness. The objective of OTSLP is to minimize the total weighted earliness of the shipments.

After formalizing the OTSLP as a mathematical optimization problem, we develop two alternative MILP model formulations. The first MILP model relies on the continuous-time framework and the second one on the discrete-time framework. Moreover, we reformulate the optimization problem as the well-known generalized set covering problem, which we use as the basis for a column generation scheme and develop an exact branch-and-price algorithm. We investigate the computational performance of both MILP models and the branch-and-price algorithm in appropriate numerical experiments. Moreover, we explore some managerial aspects, such as the impact of trucks' time window management and the workforce amount on delivery punctuality. Chapter B bases on the working paper Tadumadze and Emde (2020).

Chapters B and C focus on the operations at the dock door inside the terminal building. In contrast, Chapter D rather concentrates on the processes within the terminal yard. As mentioned above, intra-yard trailer transshipment can be executed either directly by road trucks or by spotters, driven by terminal employees. The latter setting leads to new planning problem, namely, which trailer has to be shunted by which spotter.

There has been no detailed investigation of spotter's involvement in the trailer yard's daily operations. To the best of the author's knowledge, the only works dealing with

spotters are proposed by Berghman et al. (2014) and Berghman and Leus (2015). They investigate a truck scheduling problem and consider that terminal's local shunters are responsible for intra-yard trailer transportation. The related optimization problem is modeled as a three-stage flow shop, where a semitrailer corresponds to a job that has to be processed in the following three steps:

- at first, the parked semitrailer has to be transported from the parking lot to the gate;
- then, it has to be loaded or unloaded at the dock door;
- finally, it has to be retrieved from the gate to the parking lot.

The first and the third stages are executed on a set of identical parallel machines, which represent a fleet of spotters. The second stage is processed on a separate set of identical parallel machines, representing the terminal gates. Berghman et al. (2014) formalize the resulting problem and provide mixed-integer linear programming models that can be solved using a commercial standard solver. Based on that work, Berghman and Leus (2015) propose several meta-heuristics for a slightly altered model and test their computational performance on larger problem instances, generated according to real-life data collected at Toyota Parts Center Europe.

The interdependence between the truck scheduling problem and the spotter scheduling problem is obvious, and it seems meaningful to solve them simultaneously. Nevertheless, the resulting holistic problem is too complex, and solving realistic problem instances in a reasonable time is technically challenging. Hence, some simplifying assumptions have to be made on both sides, i.e., the truck scheduling and the spotter scheduling. Berghman et al. (2014) and Berghman and Leus (2015), too, make several simplifying assumptions on both sides of the integrated problem. On the *truck scheduling* side, they model the gates as a set of identical parallel machines. By doing this, the authors assume that all dock doors are equipped with an equal amount of resources, leading to fixed (un-)loading times of the trucks. Also, on the *spotter scheduling* side, the model implies that the trailer transportation time lasts the same amount of time, neglecting the actual position of the parked trailer and the assigned gate. In reality, the trailer's transportation time strongly depends on the distance between its parking position and the dedicated dock door. Finally, the authors ignore spotter's deadheading times between successive transportation jobs. By doing that, the model assumes that spotters are always directly available at the start point of each task independent from the endpoint of the previous

transportation request.

To model the above-discussed points more precisely, keeping in mind that problem instances of realistic size must be solvable in an adequate time, we suggest to divide this holistic problem into two handier problems and solve them successively. This way, we can solve the truck (and workforce) scheduling problem in the first stage, ignoring the spotter scheduling part. Subsequently, we can solve the remaining spotter scheduling problem with the given truck schedules. The former problem can be solved, e.g., as described in Chapters B and C, while the latter's solution methods, i.e., scheduling spotters with the given dock door and docking time assignments, are proposed in Chapter D.

Chapter D deals with a *robust spotter scheduling problem* (dubbed as RSSP) that considers the truck schedules as given, solved in the previous planning step. In this regard, RSSP seeks such an assignment of transportation requests (or jobs) to the spotters, which allows timely transportation of trailers within the yard. A job corresponds to shunting a semitrailer from a parking space to a concrete dock door or vice versa. Each transportation request has a fixed completion time, by which the trailer has to be moved. Moreover, it has a determined processing time, which corresponds to the trailer's transportation time. Finally, we consider sequence-dependent setup times for each job pair, representing the spotter's deadheading time when executing these two jobs consecutively. We can derive all transportation requests and the related parameters according to the predefined truck schedules.

The goal of RSSP is to find robust spotter schedules, which allow the realization of the given truck schedules even if some unforeseen delays occur in the yard. Such delays can be caused by external factors, such as a traffic situation on the roads (leading to trucks' late arrival), or internal factors, like congestions of the terminal's material handling process (causing prolonged handling times at the gates). We aim to protect the system against the propagation of such small delays. For this, we insert buffer times between any pair of consecutive jobs and intend to keep them as large as possible. Sufficiently large buffer times can protect small delays from being propagated, so that each job can be executed on time by a spotter, even if its previous task completes later than estimated. We propose two variants of the RSSP, which differ in terms of the objective function. Specifically, the first problem variant maximizes the total weighted buffer time, and the second one the shortest weighted buffer time.

After formalizing the resulting optimization problems accordingly, we develop efficient exact algorithms, obtaining optimal solutions in polynomial time. We test the algo-

rithms' computational performance in an extensive computational study using newly generated large-scale problem instances. Moreover, we evaluate the robustness of the obtained solutions in a simulation study. In particular, we simulate some external and internal delays in the yard and explore how the found solutions can protect the system against such uncertainties. Moreover, we investigate the effect of the spotter fleet size on the yard's performance. Chapter D has been published as Tadumadze, G., Boysen, N., Emde, S. (2020): Robust spotter scheduling in trailer yards. *OR Spectrum* 42(4), 995–1021. The definitive publisher authenticated version is available online at `https://doi.org/10.1007/s00291-020-00599-5`.

While Chapters B, E and D deal with scheduling problems with a concrete application, the focus of Chapter E is primarily on the solution algorithms. Specifically, in Chapter E, we propose novel exact and heuristic algorithms to solve a more general class of parallel machine scheduling problems with several practical model extensions and different objectives. In particular, the baseline optimization problem is to schedule a set of jobs with time windows (i.e., release dates and deadlines) on a set of unrelated parallel machines, minimizing some min-max objective. This kind of optimization problem covers a broad class of truck scheduling (and other) problems.

In the truck scheduling context, we can equate a job with a truck, where the job's release date and deadline correspond to the truck's arrival and departure time, respectively. A machine corresponds to a dock door that can handle at most one job (i.e., truck) at a time. Depending on how the specific door assignment influences the truck's processing time, the set of parallel machines become unrelated, related, or identical. To cover a wide range of applications, we observe the most general setting with unrelated machines. By doing this, we address the extensive case, where the truck's processing time can vary from gate to gate and may depend on a specific storage location of the truckload and the specific resource amount at the dock door. Note that our general model also covers the special cases with identical and related (or uniform) parallel machines. They correspond to such kinds of truck scheduling problems where the trucks' processing is identical at every gate, or where it only depends on the varying amount of the assigned resources (e.g., number of logistics workers) at the door.

Besides the truck scheduling application, the described optimization problem has further practical applications. Another usage of the proposed problem is berth allocation at maritime container terminals, where vessels (i.e., jobs) are moored at specific berths (i.e., machines) alongside the quay wall. Also, in the most general setting of the berth

allocation problem, it is meaningful to model berths as unrelated machines. Firstly, each ship may have a so-called "optimal berthing point", which may depend on the storage location of its cargo in the container terminal. Secondly, the berths may differ in terms of the resources (e.g., quay cranes). Hence, modeling berths as unrelated parallel machines covers the most extensive setting, where each ship defines a berth-dependent processing time.

We investigate three different min-max objectives, which are relevant for both truck scheduling and berth allocation applications.

- For successive planning runs, it is highly desirable to clear the terminal quickly (Boysen and Fliedner, 2010). Thus, our first objective is to complete the latest job's processing as soon as possible, i.e., we minimize the *makespan*.
- For a better truck (or vessel) utilization, it is advisable to serve each customer (i.e., truck or ship) timely. Thus, it is rational to minimize their service times (or flow times), defined as the time lap between the customer's arrival and the completion time. This way, the transportation device can depart promptly to the next destination. Our second objective is to minimize the *maximum (weighted) flow time*, aiming to improve the worst-served customer's service time.
- In practice, it is not always possible to find such schedules that ensure to serve every customer before the given deadline. In such circumstances, a more reasonable objective function may be such that allows violating some deadline restrictions but penalizes those violations. In this context, our third objective is to minimize the *maximum (weighted) lateness*.

Machine scheduling problems have drawn enormous attention from scholars during the last four decades. For a historical overview of the research in the scheduling theory in general and its applications, we refer to Pinedo (2016). Surveys on the vast amount of literature on parallel machine scheduling are provided by (e.g., Cheng and Sin, 1990, Chen et al., 1998). Pfund et al. (2004) restrict the scope of their survey to the unrelated parallel-machine scheduling problems with several performance measures, including the makespan and the maximum tardiness, which are more related to the problem variants investigated in Chapter E.

To solve the above-described optimization problem, we develop an exact method using logic-based benders decomposition scheme, which we denote as "branch & Benders cut"

(dubbed as *B&BC*). Apart from that, we design a heuristic using the problem's reformulation as a generalized set partitioning problem, which we dub as *GSPP-heuristic*. This method bases on one of the heuristic interval selection techniques, which has shown the most reliable performance in Chapter E. To evaluate the proposed algorithms' computational performance, we compare them with commercial standard solver using the original MILP model formulations. The comparison is conducted on newly generated random problem instances. Moreover, we explore the algorithmic performance of B&BC and GSPP-heuristic on existing benchmark problem instance sets for both mentioned applications. In the truck scheduling context, we use the problem instances generated in Chapter B. In the berth allocation context, we use existing benchmark sets of problem instances, which are generated based on real-world traffic data at the Port of Gioia Tauro (Cordeau et al., 2005). Chapter E has been published as Tadumadze, G., Emde, S., Diefenbach, H. (2020): Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. *OR Spectrum* 42(2), 461–497. The definitive publisher authenticated version is available using the following DOI: `https://doi.org/10.1007/s00291-020-00586-w`.

The methodological approach taken in this thesis is a mixture of techniques from discrete and combinatorial optimization. It is noteworthy that each optimization problem discussed in Chapters B, C and E (i.e., apart from the RSSP from Chapter D) is NP-hard. For each of these problems, we provide a MILP model that can be solved using a commercial standard solver, e.g., IBM ILOG CPLEX Optimizer. However, due to their high complexity, the problem instances of the real-world size cannot be efficiently solved only using such standard model formulations. Therefore, there is a need to develop specialized solution methods that can find good (i.e., not necessarily optimal, but near to optimal) solutions in an adequate time.

We reformulate each of these optimization problems as a well-known generalized set partitioning (or set covering) problem. This kind of integer linear programming (ILP) model formulations benefit from some modeling advantages and the strong LP-relaxation. On the downside, the related ILP models suffer from many variables (columns). Therefore, in Chapters B and E, we propose new heuristic techniques, which enable us to decrease the computational time at the expense of solution quality. Instead of including all variables (or columns) in the ILP model, we first heuristically preselect a subset of variables in the model preprocessing phase. Subsequently, we solve the reduced problem with a subset of preselected columns within a shorter computational time. In Chapter C, we use the set covering model formulation as the foundation for an exact branch-and-price

algorithm, where we iteratively add the variables (or columns) to the model.

Unlike the other optimization problems, both alternatives of RSSP, discussed in Chapter D, are not NP-hard. Therefore, we do not develop appropriate MILP models for them. Instead, we design an exact polynomial-time algorithm for each problem variant to efficiently achieve optimal solutions in a short computational time. Moreover, we use a discrete-event simulation technique to simulate some uncertainties and assess the robustness of the proposed approaches.

Chapter B.

Inbound truck scheduling*

Abstract: Truck scheduling coordinates the loading and unloading processes of trucks competing for the timely processing at some terminal, e.g., a cross-docking terminal or distribution center. Existing research almost invariably assumes that the (un-)loading times of trucks are fixed. In the real world, however, terminal managers have the additional flexibility of adapting the workforce for processing critical trucks. For instance, two instead of one worker can be assigned to jointly unload the parcels of a trailer reaching a terminal of the postal service industry onto a conveyor, or an additional forklift can support the removal of pallets from a truck in a distribution center. Thus, workforce management influences the processing times of trucks and a simultaneous planning of both tasks seems advisable. For two representative truck scheduling settings, mixed-integer models integrating workforce planning and truck scheduling are presented. Furthermore, we re-formulate the problems as interval scheduling problems and develop heuristics based on these formulations. Finally, the holistic planning models are compared with alternative models assuming a fixed workforce. Our findings reveal that integrated planning can considerably increase the performance of truck scheduling in terms of total flow time and punctuality.

*This chapter has been published as: Tadumadze, G., Boysen, N., Emde, S., Weidinger, F. (2019): Integrated truck and workforce scheduling to accelerate the unloading of trucks. *European Journal of Operational Research* 278 (1), 343–362.

B.1. Introduction

Many supply chains – especially those having a versatile assortment and plenty of suppliers – rely on a consolidation of goods somewhere in their distribution networks, so that a lot of incoming and outgoing trucks need to be efficiently coordinated:

- Ladier and Alpan (2016) report on a huge cross-docking terminal of a less-than-truckload logistics provider for smaller shipments where an average of 320 trucks has to be processed each day.
- In the automotive industry, thousands of parts need to be delivered each day, so that plants are reported handling up to 400 (Battini et al., 2013) or even 480 (Berghman and Leus, 2015) inbound trucks.
- Similar numbers of trucks are related to hub terminals of the postal service industry where parcels need to be sorted by fully-automated sortation conveyors (Boysen et al., 2017).

In view of these examples, it is not astounding that truck scheduling has received a lot of attention among scientists and practitioners in the recent years (Boysen and Fliedner, 2010). Typically, the number of trucks with overlapping time windows (i.e., that are jointly waiting in a terminal yard and compete for a timely processing at some dock door) is greater than the number of dock doors available at the terminal, which forces some trucks to wait while other trucks are processed. Truck scheduling assigns each incoming and/or outgoing truck to a dock door and defines a processing time interval so that some objective function – mostly ensuring a timely delivery of goods – is optimized. Computerized truck scheduling procedures specifically tailored to the needs of the respective supply chains are, for instance, provided for air cargo terminals (Ou et al., 2010, Boysen et al., 2013), automobile plants (Berghman and Leus, 2015), distribution centers of food retailers (Boysen, 2010), cross-docking terminals (Yu and Egbelu, 2008, Boysen et al., 2010), and terminals of the postal service industry (McWilliams et al., 2005, Boysen et al., 2017). An in-depth survey on the broad literature on truck scheduling is provided by Boysen and Fliedner (2010).

Existing research, however, almost invariably presupposes a fixed workforce operating each dock door, so that a fixed processing time is assigned to each truck. This finding is also supported by the two recent survey papers of Ladier and Alpan (2016) and Buijs

et al. (2014) on cross-docking research, which both state the integration of workforce scheduling as an important future research task. Indeed, in the real world, terminal managers have additional ability to adapt the workforce and this way influence the trucks' processing times. By adding additional logistics workers to help with (un-)loading a critical truck, the processing time can be accelerated. As a result, the truck schedule may be improved. Simply dedicating a fixed given number of workers to each door foregoes this flexibility. There are only a few other studies integrating scarce resources inside a terminal into truck scheduling. Shakeri et al. (2012) explicitly model a scarce number of forklifts required to move shipments between inbound and outbound docks in a cross-docking terminal once shipments are unloaded but do not consider a varying (un-)loading workforce, instead assuming a dedicated worker operating each dock. Similarly, Konur and Golias (2017) allow varying numbers of workers per door but assume that workers never switch to another door during the planning horizon. Hermel et al. (2016) allow "crash processing", where two instead of one worker is assigned to a door. Günther and Nissen (2010, 2014) investigate the sub-daily staff scheduling problem for a logistics service provider and develop three (meta-)heuristic approaches: particle swarm optimization, an evolutionary strategy, and a constructive agent-based heuristic. Their problem deals with the assignment of staff with varying qualifications to work stations with given workload and required qualification over time. Ladier et al. (2014) develop an employee scheduling decision support tool for a logistics platform consisting of three MIP models covering the decision problems from weekly timetabling to daily employee rostering. The proposed three MIPs are solved sequentially so that the output of the previous model is the input for the next model. While the models of Ladier et al. (2014) are generalized for all warehouse-related tasks, Ladier and Alpan (2015) deals with the integration of the employee timetabling and rostering problem specifically into truck scheduling operations in a cross-docking platform. Since the integrated model would constitute a highly complex problem, Ladier and Alpan (2015) suggest a sequential and two iterative approaches applying existing models for employee timetabling and truck scheduling. To the best of our knowledge, the solution of both scheduling problems in a holistic manner while taking workforce-dependent processing times into account has not been attempted before this paper.

In industry practice, there are three alternatives for (un-)loading trucks:

- Palletized goods are moved by some forklift truck or manual pallet jack between trailer and intermediate storage, e.g., a staging area or a dragline (see Bartholdi and Gue, 2000).

- Loose goods directly stored in the trailer without additional loading equipment, e.g., parcels, are individually put onto a conveyor by logistics workers. Typically, telescope conveyors are applied, which can be extended or retracted during the progressive unloading or loading process, so that a worker can always directly place each parcel onto the conveyor without additional walking (e.g., Fedtke and Boysen, 2017).

- In the recent years, fully-automated loading platforms fixedly installed at a dock have been introduced. They allow to load and unload a complete truckload in a single batch just by moving a liftable platform (e.g., Lee et al., 2014).

The development of the latter solution is still in its infancy and struggles with the wide range of different trailer types. The former two solutions, however, are the most widespread and rely on human workers. Here, truck processing can be executed by a single logistics worker, but it is also possible to flexibly add personnel to accelerate the process. Clearly, the restricted space inside a trailer limits the number of workers concurrently processing a truck to two, at most three workers and obstructions among them suggest a subadditive performance increase. Nonetheless, additional workers can accelerate the processing of critical trucks, so that the trucks' processing times depend on the workforce scheduling of logistics workers. The paper on hand is dedicated to investigating the performance gains of an integrated truck and personnel scheduling compared to the fixed worker assignment that is a precondition for the fixed processing times assumed by previous research. In industry practice, large terminals handling hundreds of trucks apply information systems for their yard management. Most of these systems are just for monitoring incoming and outgoing trucks and to coordinate the availability of loading docks, but there are also systems containing sophisticated decision support tools based on operations research (INFORM, 2019). However, also these advanced tools do not integrate truck processing and workforce scheduling, so that we explore whether going beyond the current state-of-the-art is a worthwhile endeavour. This paper aims at a general conclusion whether an integrated planning leads to a considerable performance increase compared to a successive planning approach, where both problems are solved separately. However, truck scheduling is applied in a wide range of industries and a huge number of papers has been published on this topic (e.g., Buijs et al., 2014, Ladier and Alpan, 2016), so that it seems impossible to model an integrated planning approach suited to all applications. Instead, we try to identify two cases being representative for a wide range of settings, which are described in Section B.2.

The existence of two scarce resources, i.e., dock doors and logistics workers, which need to be coordinated, resembles the processing of container ships in harbors. Our dock doors correspond to the berthing space at the quay wall and our logistics workers to the quay cranes moving the containers to or from a vessel. In harbors, the quay cranes can flexibly be relocated along the quay wall, so that, here as well, processing can be accelerated by assigning multiple resources to a vessel. Optimization procedures integrating the allocation of berthing space and the quay crane assignment are summarized by the in-depth survey papers of Bierwirth and Meisel (2010, 2015). The main difference between both areas of application is that quay cranes share a rail track, so that non-crossing constraints are relevant when relocating cranes between berths. Human logistics workers do not face these restrictions.

Another stream of research closely related to our problem is (parallel) machine scheduling with additional resources. By assigning a scarce resource to the execution of a job, its processing time can be reduced, which is indicated by the term *controllable processing times*. Our workforce resource that can be added to truck processing represents a discrete and renewable resource. It is discrete because only one, two, or three workers can jointly (un-)load a truck. Human work is also a renewable resource because it cannot be used up and is available again in each period of the planning horizon. Survey papers on this field of research are provided by Błażewicz et al. (2007), Shabtay and Steiner (2007) and Edis et al. (2013). The main distinction to truck scheduling is twofold. First, machine scheduling with controllable processing times assumes that additional resources can be added during the complete processing interval. Truck processing, however, also requires maneuvering the trailer to or from the dock, so that additional workers can only be applied during the (un-)loading phase, which is just one part of the total processing time. Other related works, dealing with varying processing power, stem from the field of energy-efficient scheduling, where the processing time of a job on some *speed-scaling* machine additionally depends on the energy consumption rate of different speed levels (e.g., Batista Abikarram et al., 2019, Che et al., 2017, Kayvanfar et al., 2014, Wu and Che, 2019).

Our problem is also somewhat mathematically similar to multi-mode resource-constrained project scheduling problems (RCPSP, surveyed by Hartmann and Briskorn (2010)). However, RCPSP models generally do not take the particularities of cross-dock scheduling into account, such as transfer times between doors and the different stages of truck processing (docking, unloading, etc.). Since these steps cannot be preempted, must follow in immediate succession, and only some steps can be accelerated, they can only be

modelled in the RCPSP in a very round-about way, including artificial maximum time lags and precedence constraints (Bartusch et al., 1988). Moreover, our truck-related objectives are not commonly considered in the multi-mode RCPSP literature.

It can be concluded that existing research is not directly applicable and new solution concepts have to be developed. To do so, our paper is structured as follows. Section B.2 describes our integrated truck and personnel scheduling problem in detail. Solution procedures are developed in Section B.3, whose computational performance is evaluated in Section B.4. In Section B.5 we benchmark our holistic approach with two alternative truck scheduling approaches based on a fixedly assigned workforce to explore the performance gains promised by a variable workforce assignment. Finally, Section B.6 concludes the paper.

B.2. Problem description

B.2.1. Verbal description and basic assumptions

This paper investigates combined truck and workforce scheduling and, thus, integrates the following decisions:

- The truck scheduling part decides on the *where* and *when* of truck processing and assigns each truck a dock door and a processing time window.
- Furthermore, we specify the number of logistics workers assigned to each dock in each time period.

Both decisions are heavily interdependent. The processing time required to complete a truck depends on the number of workers assigned to the respective dock during the processing time window. More workers accelerate the processing whereas fewer workers prolong the required time span.

Our two representative truck scheduling settings are based on the following assumptions:

(A1): We consider a truck scheduling problem for inbound trucks only, where the outbound trucks are assumed to be scheduled beforehand.

In the real world, the outbound side is often separately planned (Ladier and Alpan, 2016). In many practical settings, outbound trucks are assigned fixed departure times,

so that they can arrive at their destinations in a timely manner. This can, for instance, be another depot where the delivery tours to the final customers, started the next morning, have to be reached on time (Boysen and Fliedner, 2010). In such a setting, inbound trucks are to be scheduled such that their shipments reach the predefined departure times of the dedicated outbound trucks (Boysen et al., 2013). The alternative setting where an outbound truck may only leave once all predefined shipments dedicated to this truck have been loaded (e.g., Boysen et al., 2010, Bodnar et al., 2017) requires a concerted planning of inbound and outbound trucks. Such a setting is, however, rarely found in practical applications (Ladier and Alpan, 2016), so that we assume the outbound side to be given. We, thus, only derive a schedule for a given set $J = \{1, \ldots, |J|\}$ of inbound trucks, which are to be processed at a given set $D = \{1, \ldots, |D|\}$ of inbound dock doors.

(A2): Processing an inbound truck at a dock door consists of multiple steps. Among these steps, only the unloading step can be accelerated by adding extra resources.

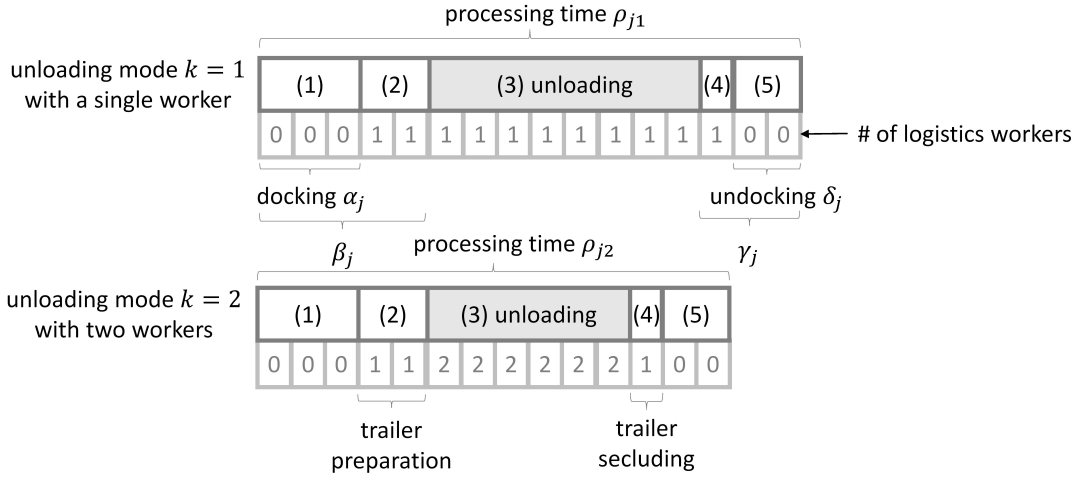


Figure B.1.: Components of the processing time of truck j

The basic steps to be executed when processing an inbound truck at a dock door are schematically depicted in Figure B.1. First, the trailer needs to be maneuvered backwards towards the respective dock (1). This docking step is either directly executed by the delivery truck or by a terminal-owned trailer truck (also denoted as spotter or switcher, see Yano et al., 1998). In both cases (which need not be further differentiated), a logistics worker is not required and the duration of this step takes α_j time units. Once the trailer is docked, a logistics worker has to prepare unloading (2). This step includes activities like breaking the seal, opening the trailer doors, removing load securing, and

positioning a telescope conveyor. This step can hardly be accelerated by additional workers, so that we assume that it is only executed by a single worker. In total, executing steps (1) and (2) for truck j takes β_j time units. Then, the unloading step is to be executed where the shipments are removed from the trailer (3). This step can be accelerated by additional workers and depending on the number of workers applied we differentiate alternative *modes of unloading*. Set $K_j = \{1, \ldots, |K_j|\}$ contains all modes associated with truck j and processing time ρ_{jk} denotes the resulting total processing time, i.e., of all five steps, if unloading mode k, i.e., with k workers unloading, is selected for truck j. Recall that the limited space in a trailer, typically, restricts the possible modes to one, two, or three workers during the unloading phase and obstructions among them lead to a subadditive performance increase. Once all shipments are removed, unloading is secluded and the trailer has to be closed by a single worker (4). Finally, the trailer is to be removed from the dock (5). The undocking step (5) takes δ_j time units and steps (4) and (5) together last γ_j time units. Depending on the unloading mode k^* selected for truck j, the actual processing time $p_j = \rho_{jk^*}$ is realized and once this time has elapsed, the dock door is available again for a subsequent inbound truck.

(A3): The number of workers executing the unloading step of a truck remains unchanged.

This assumption rules out that workers execute only parts of the unloading process step. If additional workers are applied to support the unloading step (3), they are bound from its start until completion. This simplifying assumption considerably reduces the solutions space and eases the mathematical problem definition. However, it seems also reasonable because of practical considerations. It is much simpler to direct workers from trailer to trailer whenever a task is completed, instead of forcing them to steadily observe time to not miss a premature changeover.

(A4): We assume that workers can switch from one dock to another instantaneously.

This is without doubt a simplification of reality where the setup time depends on the distance to be bridged between two successive docks. However, the walking time is typically considerably shorter than the unloading time and at least a fixed setup time can be considered when calculating the processing times of a mode, i.e., by an ex-ante reduction of the performance increase. The notable benefit of this simplification on the modeling side is that workers can be treated as an aggregate resource and no individual workers must be assigned to specific unloading tasks in order to calculate sequence-dependent setup times. For answering our general research question whether integrating workforce planning is a worthwhile task, this simplification seems pardonable, but relaxing this

limiting assumption is undoubtedly a challenging task for future research.

(A5): We assume that arrival times and latest departure times for each incoming truck are known and have to be considered.

To extract the impact of an integrated truck and workforce scheduling, we model a basic truck scheduling problem and abstain from integrating further peculiarities like limited intermediate storage space (Miao et al., 2009), preemptive truck processing (Larbi et al., 2011), or interchangeable products (Boysen et al., 2010). We only consider earliest arrival times r_j and latest departure times $\overline{d}_j$, which are among the most elementary practical constraints a terminal faces (Bodnar et al., 2017).

(A6): We presuppose that all time related parameters have integer values.

Note that any real-valued parameters can be scaled to integer values to arbitrary precision.

(A7): The total number of available workers for unloading operations is assumed to be given and fixed.

Truck scheduling and workforce scheduling have conflicting objectives. On the one hand, fast processing of trucks can be supported by additional workers. Workforce scheduling, however, aims at an efficient resource allocation and rather prefers schedules where fewer workers are applied. To resolve this basic trade-off, we assume a given constant workforce W of unloading workers. This way, optimized truck schedules can be derived for differently sized workforces, so that an efficient frontier is gained and an informed terminal manager can select the best truck and workforce schedule among the given alternatives. We address this topic in more detail in Section B.5. Note that the models and solution methods proposed in the following can easily be extended to account for time-varying availability of workers by replacing the static set of workers W by the dynamic sets of workers W_t.

We consider the truck scheduling problem in two basic settings, which differ from one another with regard to their objective functions:

- The first objective covers all cases in which the performance exclusively depends on the completion times C_j of the trucks. Specifically, we minimize the sum of the trucks' flow times. Typically, completion time dependent objectives are well-suited for distribution centers or factories, where inbound shipments can be stored tem-

porarily before they are reloaded onto outbound trailers in order to be transported to their next destination. In such a setting, it is the main objective to get the trucks back on the road as fast as possible.

- Our second objective minimizes the weighted sum of missed shipments that are too late to be loaded onto their outbound trailers. It considers the case where the goods must be directly reloaded onto outbound trucks without the intermediate storage step. This objective is suitable for a cross-docking environment or a terminal used by the postal service industry (Boysen et al., 2013), where shipments are to be directly reloaded onto outbound trucks. Here, the performance depends not only on the completion times of the inbound trucks but also their dock assignments which plays a crucial role for intra-facility transportation times of shipments: e.g., if an assigned dock door of an inbound truck is far from the dock door of the corresponding outbound truck, then the transportation between inbound and outbound doors takes a long time, which could result in late transshipment.

These assumptions and objectives lead us to two different integrated truck and workforce scheduling problems (dubbed ITWS), which we elaborate in the following two sections.

B.2.2. ITWS in the distribution center context

Consider a given set $J = \{1, \ldots, |J|\}$ of inbound trucks, which in a given discrete time frame $t = 1, \ldots, T$ are to be processed at a given set $D = \{1, \ldots, |D|\}$ of dock doors, which are exclusively reserved to inbound operations. Processing trucks includes their unloading for which a given workforce $W = \{1, \ldots, |W|\}$ of logistics workers is available. Each truck $j \in J$ can be processed in different (predefined) modes of unloading. The modes $K_j = \{1, \ldots, |K_j|\}$ vary in the number of workers executing the unloading step and, thus, each mode $k \in K_j$ of truck j is assigned a specific processing time ρ_{jk}. Truck processing is bound to time windows defined by arrival time r_j and deadline $\overline{d}_j$. A solution consists of the following decisions:

- for each truck $j \in J$, a mode of unloading $k \in K_j$ is to be selected,
- for each truck $j \in J$, a completion time C_j has to be set: once a mode of unloading $k \in K_j$ for truck j is selected, its processing time p_j is determined; the remaining decision is to schedule this processing interval within the truck time window,

- a dock door is to be assigned to each truck $j \in J$ where it is processed, and
- workers need to be assigned to docks over time, such that always enough workers are available to execute the processing intervals (according to the number of workers defined by the selected mode of unloading) and the total workforce is not exceeded.

Table B.1.: Notation for ITWS-DC

Sets:	
J	Set of inbound trucks (index: $i, j = \{1, 2, \ldots, \lvert J\rvert\}$)
D	Set of available inbound docks (index: $g = \{1, 2, \ldots, \lvert D\rvert\}$)
K_j	Set of unloading modes for truck j (index: $k = \{1, 2, \ldots, \lvert K_j\rvert\}$)
Input data:	
T	Number of periods with $t = 1, \ldots, T$
$\lvert W\rvert$	Total number of available workers per time unit
ρ_{jk}	Processing time of truck j if it is unloaded in mode k
r_j	Arrival time of truck j
$\overline{d}_j$	Deadline of truck j
α_j	Duration of truck j's initial docking (step (1))
β_j	Duration of truck j's initial docking plus preparation (step (1) + step (2))
γ_j	Duration of truck j's preparation for trailer secluding plus undocking (step (4) + step (5))
δ_j	Duration of truck j's final undocking (step (5))
M	Big integer
Decision variables:	
p_j	Continuous variable: processing time for unloading inbound truck j
P	Set of p_j variables: $P = \{p_j \mid j \in J\}$
C_j	Continuous variable: completion time of inbound truck j
C	Set of C_j variables: $C = \{C_j \mid j \in J\}$
x_{ij}	Binary variable: 1, if inbound truck j is processed directly after inbound truck i at the same dock; 0, otherwise
x_{0j}	Binary variable: 1, if inbound truck j is processed first at a dock; 0, otherwise
x_{jn}	Binary variable: 1, if inbound truck j is processed last at a dock; 0, otherwise ($n = \lvert J\rvert + 1$)
X	Set of x_{ij} variables: $X = \{x_{ij} \mid i \in J \cup \{0\}, j \in J \cup \{n\}\}$
y_{jkt}	Binary variable: 1, if processing truck j is in mode k and ends in period t; 0, otherwise
Y	Set of y_{jkt} variables: $Y = \{y_{jkt} \mid j \in J, k \in K_j, t \in \{1, \ldots, T\}\}$

Among all feasible solutions we seek one that minimizes the flow times of trucks, i.e., $F = \sum_{j \in J}(C_j - r_j)$, such as to minimize the total time during which trucks are bound to the terminal. Such a completion time dependent objective seems well-suited if the shipments are not instantaneously loaded onto outbound trucks, but remain (at least for some time) in intermediate storage at a distribution center or manufacturing plant. From a machine scheduling perspective, this corresponds to the following (adapted) tuple

notation based on Graham et al. (1979): $[P|r_j;\overline{d}_j;p_j(k)|\sum F_j]$. Here, $\sum F_j$ and $p_j(k)$ indicate the flow time objective and a processing time depending on the unloading mode, respectively. However, due to division of trucks' processing in several phases, our ITWS problem cannot be purely characterized as a classical machine scheduling problem. An optimal schedule for an example problem is illustrated in Appendix B.7.

A mixed-integer program for the resulting problem setting dubbed ITWS-DC consists of objective function (B.1) and constraints (B.2) - (B.13). The applied notation is summarized in Table B.1.

$$\text{[ITWS-DC] Minimize } F(P,C,X,Y) = \sum_{j\in J}(C_j - r_j) \tag{B.1}$$

subject to

$$\sum_{k\in K_j} \rho_{jk} \cdot \sum_{t=1}^{T} y_{jkt} = p_j \qquad \forall\, j \in J \tag{B.2}$$

$$\sum_{t=1}^{T} t \cdot \sum_{k\in K_j} y_{jkt} = C_j \qquad \forall\, j \in J \tag{B.3}$$

$$\sum_{k\in K_j}\sum_{t=1}^{T} y_{jkt} = 1 \qquad \forall\, j \in J \tag{B.4}$$

$$\sum_{j\in J}\sum_{k\in K_j}\left(\sum_{t'=t+\rho_{jk}-\beta_j}^{\min\{T;\ t+\rho_{jk}-\alpha_j-1\}} y_{jkt'} + \sum_{t'=t+\gamma_j}^{\min\{T;\ t+\rho_{jk}-\beta_j-1\}} y_{jkt'}\cdot k + \sum_{t'=t+\delta_j}^{\min\{T;\ t+\gamma_j-1\}} y_{jkt'}\right) \le |W| \qquad \forall\, t = 1,\ldots,T \tag{B.5}$$

$$\sum_{\substack{i\in J\cup\{0\}:\\ i\neq j}} x_{ij} = 1 \qquad \forall\, j \in J \tag{B.6}$$

$$\sum_{j\in J} x_{0j} \le |D| \tag{B.7}$$

$$\sum_{\substack{i\in J\cup\{0\}:\\ i\neq j}} x_{ij} = \sum_{\substack{i\in J\cup\{n\}:\\ i\neq j}} x_{ji} \qquad \forall\, j \in J \tag{B.8}$$

$$C_j \ge C_i + p_j - M\cdot(1 - x_{ij}) \qquad \forall\, j \in J;\ i \in J\cup\{0\} \tag{B.9}$$

$$r_j + p_j \leq C_j \leq \overline{d}_j \qquad \forall\, j \in J \tag{B.10}$$

$$C_0 = 0 \tag{B.11}$$

$$x_{ij} \in \{0,\, 1\} \qquad \forall\, i, j \in J \cup \{0, n\} \tag{B.12}$$

$$y_{jkt} \in \{0,\, 1\} \qquad \forall\, j \in J;\, k \in K_j;\, t = 1, \ldots, T \tag{B.13}$$

Objective function (B.1) minimizes the sum of flow times over all trucks. Constraints (B.2) ensure that the realized processing time p_j of truck j equals that of the selected unloading mode. Furthermore, (B.3) take care that the completion time of each truck matches the mode selection variable y_{jkt}, which with a value of 1 indicates that mode k for truck j ends in period t and (B.4) ensure that exactly one mode and a single completion time is selected for each truck. The given workforce is considered in (B.5). In no period, the number of required workers may exceed the size $|W|$ of the given workforce. The three terms on the left-hand side of (B.5) represent the workforce currently required for trailer preparation (step (2) of Figure B.1), unloading (step (3)), and trailer secluding (step (4)), respectively. To avoid symmetric solutions, we do not explicitly assign trucks to docks, but build chains of trucks (i.e., groups of trucks) that are jointly assigned to the same door. Constraints (B.6) ensure that each inbound truck is processed at some dock. Inequalities (B.7) guarantee that the number of truck chains composed does not exceed the number of available doors. Constraints (B.8) ensure that the sequences of inbound trucks are well-defined. Specifically, each inbound truck j is processed either as the first truck at a door ($x_{0j} = 1$) or directly after another inbound truck $i \in J$ ($x_{ij} = 1$); analogously, a truck is processed either as the last truck at a dock ($x_{jn} = 1$) or is directly followed by another inbound truck $i \in J$ ($x_{ji} = 1$). Constraints (B.9), (B.10), and (B.11) define the completion time C_j for each truck j. It has to be later than that of its predecessor truck (B.9), range in-between arrival time and deadline (B.10), and processing of virtual truck 0 ends at the beginning of the planning horizon (B.11). Note that $M = \max_{j \in J}\{\overline{d}_j\}$ is sufficiently large. Finally, constraints (B.12) and (B.13) represent the binary integrality requirement of 0-1 variables.

Theorem B.1. *Finding a feasible solution to ITWS-DC is strongly NP-complete for* $|D| \geq 1$.

The complexity status of ITWS-DC, i.e., even finding a feasible solution turns out as a complex task, directly follows from the ITWS-DC being a generalization of the single machine scheduling problem with given arrival times and deadlines, which is well-known to be strongly NP-complete (Garey and Johnson, 1979).

B.2.3. ITWS in the cross-docking context

In a cross-docking environment, incoming shipments are to be quickly reloaded onto outbound trucks. Here, apart from unloading of shipments from inbound trucks, also their intra-facility transportation time between the dock doors needs to be considered. Hence, our objective depends on both trucks' completion times and their assignments to the doors. Except for the workforce aspect, we presuppose the problem version described in Boysen et al. (2013). Specifically, we are additionally given a set of outbound trucks O where each outbound truck $o \in O$ may receive shipments from several inbound trucks. Thus, we are additionally given the weight of shipment ϕ_{jo} (e.g., the number of pallets) delivered by inbound truck j for outbound truck o. Further, each outbound truck $o \in O$ is bound to a specific outbound door and departure time d_o that is assumed to have been determined in a previous planning step, such that each outbound truck reaches its successive destination in a timely manner. Only inbound shipments that are unloaded on time, such that their completion time plus the transfer time φ_{go} from their respective dock door g towards that of outbound truck o does not exceed d_o, reach their outbound truck. All other shipments are delayed up to the next outbound truck serving the respective destination. Thus, our problem representing the cross-docking context varies in the objective function compared to ITWS-DC.

Table B.2.: Additional and altered notations for ITWS-CD

Sets:	
O	Set of outbound trucks (index: $o = \{1, 2, \ldots, \lvert O \rvert\}$)
Input data:	
ϕ_{jo}	Weight of a shipment (e.g., the number of pallets) delivered by inbound truck j for outbound truck o
d_o	Fixed departure time for outbound truck o
φ_{go}	Transfer time for moving goods from dock g to outbound truck o
Decision variables:	
x_{ij}^g	Binary variable: 1, if inbound truck j is processed directly after inbound truck i at dock g; 0, otherwise
x_{0j}^g	Binary variable: 1, if inbound truck j is processed first at dock g; 0, otherwise
x_{jn}^g	Binary variable: 1, if inbound truck j is processed last at dock g; 0, otherwise ($n = \lvert J \rvert + 1$)
X	Set of x_{ij}^g variables: $X = \{x_{ij}^g \mid i \in J \cup \{0\}, j \in J \cup \{n\}, g \in D\}$
z_{jo}	Binary variable: 1, if the shipments of truck j are too late for reaching outbound truck o; 0, otherwise
Z	Set of z_{jo} variables: $Z = \{z_{jo} \mid j \in J, o \in O\}$

To make a connection to machine scheduling, weights ϕ_{jo} can be interpreted as penalty costs, which arise whenever due date d_o cannot be met, and transfer times φ_{go} resemble

delivery times (or tails) that depend on the specific machine pairs. Thus, from a machine scheduling perspective, our objective is analogous to minimizing the total weighted number of tardy jobs (i.e., $\sum w_j U_j$), so that an (adapted) tuple notation according to Graham et al. (1979) is $[R|r_j; \overline{d}_j; p_j(k); q^j_{go}|\sum w_j U_j]$.

Applying the additional (or slightly altered) notation defined in Table B.2, the resulting MIP model (denoted ITWS-CD) consists of objective function (B.14) subject to constraints (B.2) - (B.5), (B.10), (B.11), (B.13) and (B.15) - (B.21).

$$\text{[ITWS-CD] Minimize } F(P,C,X,Y,Z) = \sum_{j\in J}\sum_{o\in O} \phi_{jo}\cdot z_{jo} \tag{B.14}$$

subject to (B.2) - (B.5), (B.10), (B.11), (B.13) and

$$\sum_{g\in D}\sum_{\substack{i\in J\cup\{0\}:\\ i\neq j}} x^g_{ij} = 1 \qquad \forall\, j\in J \tag{B.15}$$

$$\sum_{j\in J} x^g_{0,j} \leq 1 \qquad \forall\, g\in D \tag{B.16}$$

$$\sum_{\substack{i\in J\cup\{0\}:\\ i\neq j}} x^g_{ij} = \sum_{\substack{i\in J\cup\{n\}:\\ i\neq j}} x^g_{ji} \qquad \forall\, j\in J;\, g\in D \tag{B.17}$$

$$C_j \geq C_i + p_j - M\cdot(1-x^g_{ij}) \qquad \forall\, j\in J;\, i\in J\cup\{0\};\, g\in D \tag{B.18}$$

$$z_{jo}\cdot M \geq C_j - d_o + \sum_{g\in D}\varphi_{go}\cdot\left(\sum_{\substack{i\in J\cup\{0\}:\\ i\neq j}} x^g_{ij}\right) \qquad \forall\, j\in J;\, o\in O \tag{B.19}$$

$$x^g_{ij}\in\{0,\,1\} \qquad \forall\, i,j\in J\cup\{0,n\};\, g\in D \tag{B.20}$$

$$z_{io}\in\{0,\,1\} \qquad \forall\, i\in J;\, o\in O \tag{B.21}$$

Objective function (B.14) minimizes the weighted sum of lost shipments. All formulae defining the selection of unloading modes can directly be taken over from model ITWS-DC. However, as the objective function now depends on the dock each truck is assigned to, we have to assign each truck chain to its specific dock g. Hence, we define an additional index $g\in D$ for decision variables x^g_{ji}. The basic new ingredient is (B.19), which sets decision variables z_{jo} to one whenever an inbound truck j is scheduled such that the shipments dedicated to outbound truck o cannot be loaded on time. Note that $M = \max_{j\in J}\{\overline{d}_j\}$ is not sufficiently large for ITWS-CD. Instead, $M = \max_{j\in J}\{\overline{d}_j\} +$

$\max_{g \in D, o \in O}\{\phi_{go}\}$ must be applied.

An optimal schedule in the cross-docking environment for an example problem can be found in Appendix B.7.

Theorem B.2. *Finding a feasible solution to ITWS-CD is strongly NP-complete for* $|D| \geq 1$.

The complexity status of ITWS-CD is the same as that of ITWS-DC, because of the same consideration presented above.

B.3. Solution procedures

To derive a relatively simple solution procedure that is easily adaptable to all variants of truck scheduling problems, we reduce both versions of ITWS to interval scheduling problems. This approach has proven to be very successful in related truck scheduling problems (Boysen et al., 2017). It is well-known that any time-indexed scheduling problem can be formulated as a discrete interval scheduling problem (e.g., Kolen et al., 2007). This model formulation assumes all time-related parameters to have an integer value. For practical purposes, this assumption can be abrogated by scaling and rescaling the time unit before and after conducting the solution procedure, respectively. This approach allows us to generate a subset of feasible execution intervals, so that not all feasible intervals per truck are considered a potential interval. Instead, for each inbound truck j a finite set M_j of execution intervals is selected. Each execution interval $m \in M_j$ defines an unloading mode $k_{jm} \in K_j$ (i.e., number of workers), which determines the interval length, and a completion time $C_{jm} \in [r_j + \rho_{jk_{jm}}; \overline{d}_j]$ (note that the processing start time is implicit). This way, for each inbound truck $j \in J$, execution interval $m \in M_j$, and period $t = 1, \ldots, T$, we can compute the following parameters:

- $a_{jmt} \in \mathbb{N}$: number of required workers during period t processing inbound truck j with regard to execution interval $m \in M_j$, and
- $b_{jmt} \in \{0, 1\}$: has a fixed value 1, if execution interval $m \in M_j$ considers corresponding truck j to be processed during period t (i.e., docked at the door), 0 otherwise.

As a result, the problem reduces to selecting one execution interval for each truck j from

given set M_j of intervals.

It stands to reason that the set M_j of potential intervals has a substantial effect on the performance of the solution procedure. The number of all feasible intervals for truck j consists of exactly $\sum_{k \in K_j} (\overline{d}_j - r_j - \rho_{jk} + 1)$ different elements, which differ from each other either in unloading mode k_{jm} or in completion time C_{jm} or both. While it would be possible to include all possible intervals in sets M_j, this would likely lead to slow solution times. Instead, we only select $|M_j| = \sum_{k \in K_j} \left\lceil \frac{\overline{d}_j - r_j - \rho_{jk} + 1}{\mu} \right\rceil$ execution intervals with μ as a predefined integer number ($\mu \geq 1$). By varying parameter μ, we can reduce the solution space: a greater value of μ reduces $|M_j|$, and the interval scheduling versions of our ITWS problems become easier to solve. On the other hand, fewer intervals (i.e., greater μ) increase the risk that the optimal execution intervals are not in M_j and that the optimality gap increases. We further investigate this trade-off in our computational study (Section B.4.3).

We develop the following three interval selection approaches that, for each truck $j \in J$, generate $|M_j|$ execution intervals:

- *Random* approach: as a first and very basic approach, for each execution interval $m \in M_j$, we randomly select a feasible unloading mode k_{jm} and a completion time C_{jm} from sets K_j and $\{r_j + \rho_{jk_{jm}}, r_j + \rho_{jk_{jm}} + 1, \ldots, \overline{d}_j\}$, respectively (discrete uniform distribution). To select exactly $|M_j|$ execution intervals, this step is simply repeated $|M_j|$ times. This approach is quite simple, but it does not prevent two or more identical execution intervals per truck from being generated. That is because for each interval m, both unloading mode k_{jm} (i.e., number of workers) and completion time C_{jm} are selected independently from each other. Thus, there is no guarantee that some execution intervals are missing even when $\mu = 1$.

- *Semi-random* approach: to rule out the above described redundancy (i.e., selecting multiple identical execution intervals per truck) we develop the following strategy: at first, for each unloading mode $k_{jm} \in K_j$, the set of theoretically possible completion times $\{r_j + \rho_{jk_{jm}}, r_j + \rho_{jk_{jm}} + 1, \ldots, \overline{d}_j\}$ is divided in disjunct non-overlapping subsets where each subset contains exactly μ elements (with the exception of the last subset if the term $\overline{d}_j - r_j - \rho_{jk_{jm}}$ is not divisible by μ). Then, for each feasible unloading mode $k_{jm} \in K_j$, we randomly select exactly one completion time C_{jm} from each subset. As a result, we generate more diverse execution intervals as each selected interval m is unique either in its number of workers (i.e., interval length)

or in its completion time.

- *Smoothed* approach: our third method of selecting intervals is similar to the semi-random approach, with the exception that completion times for each unloading mode $k_{jm} \in K_j$ are selected fully deterministically. Specifically, for each unloading mode $k_{jm} \in K_j$ of truck $j \in J$, we select the first and every μ-th interval m from a sorted set $\{r_j + \rho_{jk_{jm}}, r_j + \rho_{jk_{jm}} + 1, \ldots, \overline{d}_j\}$ of execution intervals, where they are sorted in ascending order according to their completion time C_{jm}. Consequently, we have such set M_j of execution intervals that the completion times for each unloading mode k are smoothly distributed within the interval of all theoretically possible completion times $[r_j + \rho_{jk}; \overline{d}_j]$.

Note that in case of $\mu = 1$, in the semi-random and smoothed approaches, *all* intervals are selected as each sub-interval consists of exactly one execution interval. Hence, the entire solution space is covered and the corresponding interval scheduling problem is equivalent to the original ITWS problem.

Given these intervals, the resulting interval scheduling problems for the two versions of our ITWS are discussed in Sections B.3.1 and B.3.2. The proposed three interval selection schemes applying different values of μ for an example problem are illustrated in Appendix B.8.

B.3.1. Interval scheduling for ITWS-DC

Once for each truck $j \in J$ a finite set M_j of execution intervals is given, we can compute their contribution to the objective value, i.e., the resulting flow time, as follows:

$$w_{jm} = C_{jm} - r_j \quad \forall j \in J;\, m \in M_j. \tag{B.22}$$

Given these weights w_{jm} and the notation summarized in Table B.3, the MIP model for the interval scheduling version of ITWS-DC, which we denote as ITWS-DC-I, consists of objective function (B.23) and constraints (B.24) - (B.27).

$$\text{[ITWS-DC-I] Minimize } F(X) = \sum_{j \in J} \sum_{m \in M_j} w_{jm} \cdot x_{jm} \tag{B.23}$$

Table B.3.: Notations for ITWS-DC-I

Sets:	
M_j	Set of predetermined intervals for processing truck j (index: $m = \{1, 2, \ldots, \lvert M_j \rvert\}$)
Input data:	
w_{jm}	Weight for selecting interval m for truck j
a_{jmt}	Number of workers applied during period t when selecting interval m for truck j
b_{jmt}	Binary parameter indicating whether ($b_{jmt} = 1$) or not ($b_{jmt} = 0$) interval m for truck j is active during period t
Decision variables:	
x_{jm}	Binary variable: 1, if interval m is selected for inbound truck j; 0, otherwise
X	Set of x_{jm} variables: $X = \{x_{jm} \mid j \in J, m \in M_j\}$

subject to

$$\sum_{m \in M_j} x_{jm} = 1 \qquad \forall\, j \in J \tag{B.24}$$

$$\sum_{j \in J} \sum_{m \in M_j} a_{jmt} \cdot x_{jm} \leq |W| \qquad \forall\, t = 1, \ldots, T \tag{B.25}$$

$$\sum_{j \in J} \sum_{m \in M_j} b_{jmt} \cdot x_{jm} \leq |D| \qquad \forall\, t = 1, \ldots, T \tag{B.26}$$

$$x_{jm} \in \{0, 1\} \qquad \forall\, j \in J;\, m \in M_j \tag{B.27}$$

Objective function (B.23) minimizes the sum of flow times over all selected intervals. Constraints (B.24) ensure that for each truck exactly one interval is selected. (B.25) and (B.26) take care that the capacities of the two limiting resources, i.e., unloading workers and docks, are not exceeded in either period. Finally, (B.27) sets the domain of the binary variables.

Unfortunately, the following complexity status holds for ITWS-DC-I. Nevertheless, as the interval scheduling formulation has the structure of general set partitioning problem, which is a very well studied problem, the respective MIP models can be expected to be quickly solvable by commercial solvers for instances of relevant size.

Theorem B.3. *Finding a feasible solution to ITWS-DC-I is strongly NP-complete.*

The proof is by reduction from the selective k-colorability problem (dubbed SKC), which is defined as follows (Demange et al., 2014):

SKC: Given an undirected graph $G = (V, E)$, where vertex set V is partitioned into given disjunct clusters $V_1, \ldots, V_p$. Is there a selection of vertices (i.e., a set V^*), such

that exactly one vertex per cluster is selected (i.e., $|V^* \cap V_i| = 1$ for all $i \in \{1, \ldots, p\}$), and is this selection colorable with exactly k colors (i.e., k-colorable), such that no pair of selected vertices connected by an edge receives identical color?

SKC was shown to be strongly NP-complete even if $k = 1$ and the graph is an interval graph, i.e., vertices represent intervals on a line, see (Demange et al., 2015). Both properties are assumed for SKC in the following proof.

Proof. The membership in NP is obvious, so that we merely define how an instance of ITWS-DC-I is derived. Each cluster represents one truck. For each truck, we assume only a single mode of unloading with a single worker, and any vertex of a cluster represents an interval for its truck. If an edge exists between two vertices, then their completion time has to be selected such that their processing intervals overlap. If we, finally, interpret the k colors as k dock doors, then the one-to-one mapping between SKC and finding a feasible ITWS-DC-I solution is readily available. □

B.3.2. Interval scheduling for ITWS-CD

Analogously, interval scheduling can also be applied to solve ITWS-CD. To derive MIP model ITWS-CD-I, the first alteration required is an alternative definition of the decision variables. Variables $x_{jmg} \in \{0, 1\}$ receive value 1, if interval m for truck j is executed at dock g (0 otherwise) (with X as set of x_{jmg} variables: $X = \{x_{jmg} \mid j \in J, m \in M_j, g \in D\}$). Furthermore, weights

$$w_{jmg} = \sum_{o \in O} \phi_{jo} \cdot \begin{cases} 1 & \text{if } C_{jm} + \varphi_{go} > d_o \\ 0 & \text{otherwise,} \end{cases} \quad \forall\, j \in J;\, m \in M_j;\, g \in D \tag{B.28}$$

define the penalty if interval m for truck j is executed at dock g. These penalties amount to the weighted sum of all missed outbound trucks. The resulting interval scheduling model ITWS-CD-I consists of objective function (B.29) and constraints (B.30) - (B.33).

$$\text{[ITWS-CD-I] Minimize } F(X) = \sum_{j \in J} \sum_{m \in M_j} \sum_{g \in D} w_{jmg} \cdot x_{jmg} \tag{B.29}$$

subject to

$$\sum_{m \in M_j} \sum_{g \in D} x_{jmg} = 1 \qquad \forall\, j \in J \tag{B.30}$$

$$\sum_{j \in J} \sum_{m \in M_j} \sum_{g \in D} a_{jmt} \cdot x_{jmg} \leq |W| \qquad \forall\, t = 1, \ldots, T \tag{B.31}$$

$$\sum_{j \in J} \sum_{m \in M_j} b_{jmt} \cdot x_{jmg} \leq 1 \qquad \forall\, t = 1, \ldots, T;\ g \in D \tag{B.32}$$

$$x_{jmg} \in \{0, 1\} \qquad \forall\, j \in J;\ m \in M_j;\ g \in D \tag{B.33}$$

Theorem B.4. *Finding a feasible solution to ITWS-CD-I is strongly NP-complete.*

This result directly follows from the proof provided for ITWS-DC-I.

B.4. Computational performance

This section explores the computational performance of our models and solution approaches. Specifically, we investigate the following research questions. First, we compare the computational performance of the two alternative MIP formulations when seeking optimal solutions: the original MIP model vs. its interval scheduling counterpart. Additionally, we investigate the sensitivity of their solution quality and computational time depending on different instance characteristics (Section B.4.2).

Our second research question is to find the best solution procedure for instances of real-world size where only heuristic solutions are obtainable (Section B.4.3). Firstly, we compare the three proposed interval selection approaches, which are described in Section B.3. Secondly, we investigate the trade-off between computational time and solution quality by tuning the parameter μ, i.e., defining the appropriate number of intervals.

Unfortunately, there is no established testbed for ITWS-DC and ITWS-CD, so that we begin this section with a description of how our test instances have been generated (Section B.4.1).

B.4.1. Instance generation

For each of our research questions, we generate data sets of two different sizes. In particular, the number of inbound trucks and dock doors are certainly critical factors that make the problem either harder or easier to solve. Hence, the size of our data sets is characterized by these two input parameters. Instances suited for our first research question have to be small enough to be solved to optimality in reasonable time by a default solver. We, thus, selected $|J| = 20$ inbound trucks and $|D| = 5$ docks to obtain very small instances (instance size dubbed as XS), and $|J| = 50$ inbound trucks and $|D| = 15$ docks for small-sized instances (instance size: S).

When addressing our second research question and seeking for best performing solution procedures, we require larger instances, considering 25 and 50 inbound docks and 100 and 200 inbound trucks (instance size: M and L), respectively. Note that these numbers are representative for real-world facilities of the postal service industry discussed in (Boysen et al., 2017). The input parameters displayed in Table B.4 are applied to initialize our data generator, which generates 10 instances for each parameter combination as follows.

A time unit for all generated instances consists of five minutes. We assume that a truck can be unloaded by a maximum of three logistics workers. Consequently, for each inbound truck j we randomly determine a number of unloading modes $|K_j|$ from the set $\{1, 2, 3\}$. The total number of available workers $|W|$ is set to $2 \cdot |D|$. For each job, the processing times for steps (1) (docking) and (5) (undocking) are set to 5 minutes, and for the steps (2) (trailer preparation) and (4) (trailer secluding) to 10 minutes. To generate the processing time for the third step (unloading) using the different unloading modes, we first determine the duration of this phase assuming that merely one logistics worker is required to unload the trailer. The duration $\tau_{j,1}$ of the unloading step for the inbound truck j for its first mode is determined randomly from interval $[6; 18]$, meaning that the unloading step lasts between half- and one-and-a-half hours. Subsequently, we derive τ_{jk} for higher modes ($k \in \{2, \ldots, |K_j|\}$) using the equation $\tau_{jk} = \frac{\tau_{j1}}{k^{\frac{1}{\kappa}}}$, where κ is a predefined parameter. It represents the acceleration effect of additional logistics workers. $\kappa = 1$ means that unloading a truck with k logistics workers takes exactly k times less than the task being executed just by a single worker. A larger value of κ ensures a lower acceleration effect and, therefore, considers a sub-additive performance increase. Finally, the total processing time ρ_{jk} of a truck j in unloading mode k is derived by summing up the processing times over all five steps.

Arrival and departure times of inbound trucks are generated by the following procedure proposed by Boysen et al. (2017). First, a random no-wait truck schedule is created and then the time windows around the respective processing intervals are generated. Specifically, we initially create a random sequence of trucks and successively assign them to the furthest possible dock and fix preliminary start times η_j to the earliest time the selected dock is available again. Once all trucks are assigned, we determine the length of the time window for each truck by randomly drawing Ω_j and ω_j from intervals $[0;\Omega_{max}]$ and $[0;1]$, respectively. Then, arrival time r_j and deadline $\overline{d}_j$ are set to $r_j = \max\{0;\eta_j - \omega_j \cdot \Omega_j \cdot \rho_{j1}\}$ and $\overline{d}_j = \eta_j + \rho_{j1} \cdot (1 + (1-\omega_j) \cdot \Omega_j)$, respectively. In this way, we receive instances that either have tight time windows (low Ω_{max}) or more flexibility (high Ω_{max}). Finally, the length of the time horizon is set to the latest deadline of inbound truck, i.e., $T = \max_{j \in J}\{\overline{d}_j\}$.

Table B.4.: Parameters for instance generation

	Research Question 1		Research Question 2	
Instance size	XS	S	M	L
$\|D\|$	5	15	25	50
$\|J\|$	20	50	100	200
$\|O\|$	5	15	25	50
Ω_{max}	(1,2,3,4)		(1,2,3)	
κ	2		2	
μ	1		(1,2,3,4,5)	

ITWS-CD instances additionally contain a set of outbound trucks O with related parameters: φ_{go} (transfer time), Φ_{jo} (weight), and d_o (departure time of outbound trucks). For all instances, we equate the number of outbound trucks $|O|$ with the number of inbound docks $|D|$. φ_{go} is assumed to be proportional to the distance between two respective docks. The distances are derived for the cross-docking terminal with a simple I-shape and rectilinear distances, where the inbound and outbound trucks are docked at opposite sides of the terminal (for a detailed description see e.g., Bartholdi and Gue, 2004, Stephan and Boysen, 2011b). Furthermore, the number of total unit loads per inbound truck ν_j is randomly drawn with uniform distribution from the interval $[30;38]$. This corresponds to a typical truckload measured in European block pallets (100×120 cm). For each inbound truck j we draw $\lfloor 0.5 \cdot |O| \rfloor$ uniformly distributed numbers from interval $[0;1]$ and proportionally assign the ν_j pallets to random outbound trucks (as far as rounding differences allow). Finally, we determine d_o, such that the objective value

is potentially affected by the respective truck assignment. To do so, for each outbound truck o we derive a feasible no-wait truck schedule of all inbound trucks dedicated to o, similar to the procedure outlined above while we only take the inbound trucks carrying goods for truck o into account. According to this schedule, we set d_o to equal the average completion time of its inbound trucks, i.e., the time when the movement of the goods from inbound truck j to outbound truck o is completed. In total, we generated and solved 320 MIPs for the first research question and 1800 MIPs for the second. The problem instances can be downloaded using the following DOI: 10.5281/zenodo.1487845.

We have implemented our solution procedures in VB.NET (Visual Studio 2015) and applied standard solver IBM ILOG CPLEX Optimizer V12.6.3 for solving our MIP models. All tests have been executed on a PC with a 4.01 GHz Intel Core processor and 64 GB RAM. The solution time for all solution approaches is limited to 30 minutes per instance.

B.4.2. Comparison of MIPs when seeking optimal solutions

In this section, we compare the computational performance of the original MIP formulation with the interval scheduling formulation for both ITWS settings. In addition, we execute a sensitivity analysis for the following input parameters: the instance size and the relative width of the time window. We observe instances from the different sized data sets (XS and S) with the varying width of the trucks' time windows $\Omega_{max} \in \{1, 2, 3, 4\}$. According to preliminary tests, acceleration parameter κ has only a marginal impact on the problem complexity. Hence, κ is set to the fixed value of 2 for all instances. First, we solve our instances with standard solver CPLEX (MIP models ITWS-DC or ITWS-CD). Furthermore, we convert them to interval scheduling problems including all possible intervals (i.e., applying $\mu = 1$) and solve them again. Note that, since we do not prune intervals heuristically, the optimal interval scheduling solution is equivalent to the optimal solution of the actual problem. Further note that, to make the comparison of two alternative formulations fair, we report both the CPU time for generating the MIP models and the CPU time the solver needs to solve them.

Table B.5 presents the numerical results for the instances in the distribution center context. Each table entry contains the aggregated results over all instances of the same parameter configuration (i.e, 10 instances). In particular, we report the average MIP generation times (column "CPU^{gen}"), - MIP solving times (column "CPU^{sol}"), the num-

ber of instances that are solved to proven optimality (column "#*") and the optimality gaps of the best upper bounds obtained by the solver using ITWS-DC within 30 minutes (column "gap^*").

Table B.5.: Numerical results: comparison between ITWS-DC and ITWS-DC-I formulations

Instance				ITWS-DC				ITWS-DC-I			
Size	$\|J\|$	$\|D\|$	Ω_{max}	CPU^{gen}	CPU^{sol}	$\#^*$	gap^*	CPU^{gen}	CPU^{sol}	$\#^*$	gap^*
XS	20	5	1	0.04	25.87	10	0.00	0.09	0.20	10	0.00
XS	20	5	2	0.09	1324.85	3	0.36	0.19	0.91	10	0.00
XS	20	5	3	0.14	1114.05	5	0.36	0.29	1.48	10	0.00
XS	20	5	4	0.17	1263.74	3	0.41	0.47	2.24	10	0.00
S	50	15	1	0.19	1453.46	3	0.57	0.27	1.42	10	0.00
S	50	15	2	0.17	1630.79	1	2.89	0.45	2.11	10	0.00
S	50	15	3	0.19	1462.48	2	2.51	0.67	2.26	10	0.00
S	50	15	4	0.25	1696.07	1	4.80	0.99	4.12	10	0.00

ITWS-DC-I is strictly superior to ITWS-DC, since it provides optimal solutions for all 80 instances considerably faster while for some ITWS-DC instances, the solver cannot terminate within the time limit of 30 minutes. In particular, ITWS-DC finds the optimal solution for only 28 of the 80 instances.

Table B.6.: Numerical results: comparison between ITWS-CD and ITWS-CD-I formulations

Instance				ITWS-CD					ITWS-CD-I			
Size	$\|J\|$	$\|D\|$	Ω_{max}	CPU^{gen}	CPU^{sol}	$\#^{feas}$	$\#^*$	gap^*	CPU^{gen}	CPU^{sol}	$\#^*$	gap^*
XS	20	5	1	0.05	1.60	10	10	0.00	0.33	0.24	10	0.00
XS	20	5	2	0.06	11.37	10	10	0.00	0.65	0.42	10	0.00
XS	20	5	3	0.16	515.50	10	9	0.00	0.71	4.60	10	0.00
XS	20	5	4	0.08	447.57	10	8	0.00	1.18	1.50	10	0.00
S	50	15	1	0.66	>1800	10	0	22.34	2.22	18.99	10	0.00
S	50	15	2	0.91	>1800	2	0	29.68	4.64	44.50	10	0.00
S	50	15	3	0.71	1526.26	4	1	17.43	4.35	21.15	10	0.00
S	50	15	4	0.74	160.07	1	1	0.00	10.07	52.48	10	0.00

Parameter Ω_{max} shows ambiguous effects on the problem complexity. On the one hand, larger time windows raise planning horizon T and, thus, the amount of feasible intervals per truck. As a result, the size of the interval based MIP models (in terms of number of constraints and variables) and the size of the solution space increase. This leads to longer computational times for both models. On the other hand, a very high value of Ω_{max} simplifies the time window related constraints of ITWS-DC, which facilitates the solution process. Our results do not show any clear trends with regard to the effect of Ω_{max}. In any case, however, the interval scheduling formulation considerably outperforms the original MIP model formulation.

The corresponding results for the cross-docking context are summarized in Table B.6. Standard solver CPLEX solving the original MIP model ITWS-CD fails to find even a feasible solution for some instances from data set S. Therefore, we additionally report the number of instances solved to feasibility per parameter setting (column "$\#^{feas}$"). As expected, the ITWS problem in the cross-docking context is harder to solve than in the distribution center context, where the assignment of inbound trucks to specific docks is not part of the decision problem. In total, ITWS-CD provides a feasible solution merely for 57 out of 80 instances, out of which 39 are optimal. Applying interval scheduling formulation ITWS-CD-I, CPLEX rapidly solves all generated 80 instances to proven optimality. Nonetheless, a comparison between the results of ITWS-CD-I and ITWS-DC-I highlights the higher problem complexity in the cross-docking context. With regard to our first research question, it can be concluded that interval scheduling clearly outperforms original MIP formulations when seeking optimal solutions with a default solver.

B.4.3. Performance of heuristic solution procedures

In this section, we evaluate the heuristic performance of our interval scheduling approach applying the three interval selection methods and the varying values of parameter μ. To do so, we apply the problem instances from data sets M and L with varying relative width of time windows ($\Omega_{max} \in \{1, 2, 3\}$). Each data set consists of 30 instances per problem setting, so that for each version of ITWS we observe 60 instances. Each ITWS instance is first transformed to the corresponding interval scheduling problems using the three approaches described in Section B.3 and the values of μ from set $\{1, 2, 3, 4, 5\}$, and then solved by the default solver. As a result, each ITWS instance is solved 15 times.

Table B.7 summarizes the computational results of our comparison for the distribution center context. The average computational times required by CPLEX to generate and solve the interval scheduling problems are reported within column "CPU". Moreover, column "$\#'^{*}$" contains the number of instances that are solved to optimality within the time limit. Recall that solving an interval scheduling problem instance to optimality does not necessarily lead to the optimal solution of the original ITWS problem (with the exception, when the exact method is applied, i.e., $\mu = 1$ and the smoothed/semi-random approach is used). Finally, the average optimality gaps are reported in column "gap^{*}", where they are computed with respect to the actual optimal solution of the problem (i.e., those obtained when $\mu = 1$). As can be seen from the results, all 900 generated interval

Table B.7.: Performance of interval selection approaches and space reduction level μ for ITWS-DC-I

Instance					Random approach			Semi-random approach			Smoothed approach		
Size	$\|J\|$	$\|D\|$	Ω_{max}	μ	CPU	$\#'^*$	gap^*	CPU	$\#'^*$	gap^*	CPU	$\#'^*$	gap^*
M	100	25	1	1	0.75	10	2.80	0.79	10	0.00	0.79	10	0.00
M	100	25	2	1	1.32	10	2.41	1.41	10	0.00	1.37	10	0.00
M	100	25	3	1	1.82	10	2.46	1.95	10	0.00	1.96	10	0.00
L	200	50	1	1	1.42	10	2.89	1.51	10	0.00	1.48	10	0.00
L	200	50	2	1	2.71	10	2.18	3.03	10	0.00	3.03	10	0.00
L	200	50	3	1	2.86	10	2.43	3.09	10	0.00	3.11	10	0.00
M	100	25	1	2	0.46	10	6.46	0.47	10	2.89	0.51	10	0.57
M	100	25	2	2	0.78	10	5.58	0.80	10	2.50	0.80	10	0.51
M	100	25	3	2	0.98	10	5.80	1.01	10	2.52	1.00	10	0.58
L	200	50	1	2	0.83	10	6.58	0.80	10	2.71	0.81	10	0.23
L	200	50	2	2	1.38	10	5.15	1.42	10	2.40	1.42	10	0.28
L	200	50	3	2	1.47	10	5.15	1.51	10	2.52	1.55	10	0.35
M	100	25	1	3	0.33	10	9.45	0.33	10	5.58	0.35	10	0.93
M	100	25	2	3	0.62	10	8.01	0.62	10	4.74	0.57	10	0.95
M	100	25	3	3	0.77	10	8.47	0.73	10	5.07	0.73	10	0.97
L	200	50	1	3	0.61	10	8.55	0.61	10	5.26	0.61	10	0.62
L	200	50	2	3	0.99	10	7.87	1.02	10	4.36	0.95	10	0.62
L	200	50	3	3	1.03	10	7.91	1.03	10	5.04	1.03	10	0.75
M	100	25	1	4	0.32	10	12.49	0.27	10	8.34	0.31	10	1.48
M	100	25	2	4	0.49	10	9.70	0.51	10	7.13	0.46	10	1.71
M	100	25	3	4	0.71	10	10.23	0.63	10	7.05	0.58	10	1.79
L	200	50	1	4	0.52	10	11.50	0.53	10	7.53	0.50	10	0.78
L	200	50	2	4	0.85	10	9.73	0.83	10	6.78	0.79	10	1.08
L	200	50	3	4	0.86	10	10.50	0.85	10	7.67	0.84	10	1.11
M	100	25	1	5	0.28	10	13.60	0.30	10	11.04	0.22	10	2.12
M	100	25	2	5	0.48	10	12.41	0.42	10	9.77	0.45	10	1.97
M	100	25	3	5	0.59	10	12.94	0.49	10	9.53	0.49	10	2.54
L	200	50	1	5	0.47	10	13.38	0.49	10	10.28	0.45	10	1.31
L	200	50	2	5	0.72	10	11.79	0.69	10	8.13	0.71	10	1.54
L	200	50	3	5	0.71	10	11.89	0.72	10	9.74	0.73	10	1.65

scheduling instances are solved to optimality, and the computational times are (almost) negligible. Even if all possible intervals are taken into consideration (i.e., $\mu = 1$) and we apply interval scheduling as an exact approach, the average CPU time for the instances from data set M is under 1.4 seconds and for instances from data set L (i.e., $|J| = 200$ and $|D| = |O| = 50$) under 3 seconds.

While the computational times are rather short for all three approaches, the smoothed approach has the lowest optimality gap for all values of μ. The random approach is dominated by both competitors, as they are able to prevent the generation of identical intervals, while the semi-random approach is somewhere in the middle. Recall that the semi-random approach with $\mu = 1$ is an exact algorithm and provides the same (i.e., optimal) results as the smoothed approach. When the value of μ increases, however, the semi-random approach converges to the random approach, due to the random choice within the larger sub-intervals.

Figure B.2 highlights the decrease of solution quality and computational time (i.e., MIP generation and solution time) for ITWS-DC-I instances when the value of μ rises. Aggregated results yield a similar picture for both data sets. They suggest approximately a linearly rising curve of optimality gaps and an exponentially falling curve of computational times for all three approaches.

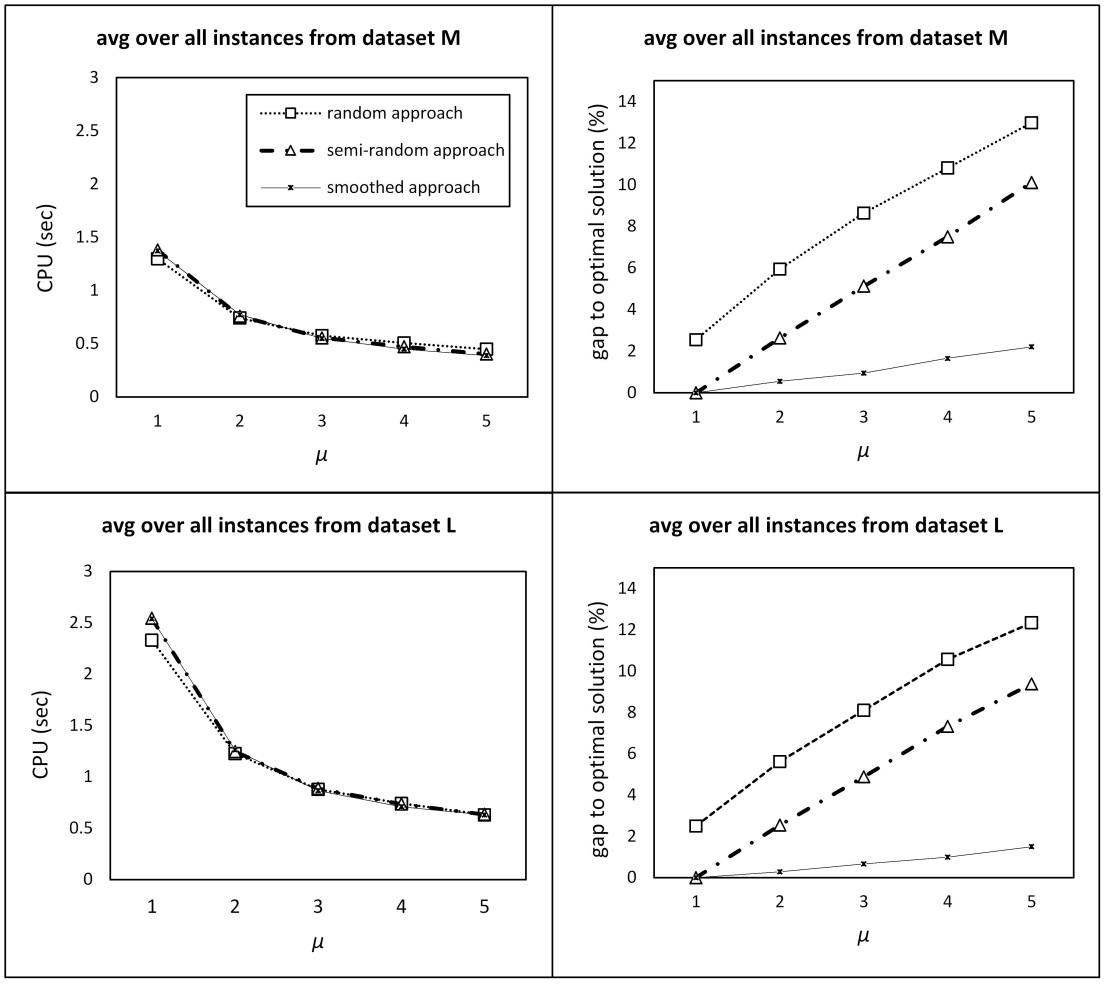


Figure B.2.: Influence of solution space reduction level μ on the solution quality (ITWS-DC-I)

Instance					Random approach				Semi-random approach				Smoothed approach			
Size	$\|J\|$	$\|D\|$	Ω_{max}	μ	CPU	$\#'^*$	gap^{UB}	gap^{LB}	CPU	$\#'^*$	gap^{UB}	gap^{LB}	CPU	$\#'^*$	gap^{UB}	gap^{LB}
M	100	25	1	1	64.95	10	2.17	2.17	68.94	10	0.00	0.00	68.68	10	0.00	0.00
M	100	25	2	1	89.80	10	2.35	2.35	192.95	10	0.00	0.00	193.59	10	0.00	0.00
M	100	25	3	1	142.33	10	2.29	2.29	269.61	10	0.00	0.00	269.19	10	0.00	0.00
L	200	50	1	1	1651.36	3	1.21	1.36	1813.92	1	0.00	0.15	1814.24	1	0.00	0.15
L	200	50	2	1	1876.21	1	1.23	1.43	1851.24	1	0.00	0.20	1851.34	1	0.00	0.20
L	200	50	3	1	1833.17	2	1.31	1.55	1739.79	1	0.00	0.24	1740.42	1	0.00	0.24
M	100	25	1	2	29.73	10	4.74	4.74	30.20	10	2.41	2.41	23.83	10	1.19	1.19
M	100	25	2	2	46.19	10	4.43	4.43	70.34	10	2.61	2.61	74.10	10	1.13	1.13
M	100	25	3	2	86.21	10	4.91	4.91	93.71	10	2.62	2.62	96.71	10	1.26	1.26
L	200	50	1	2	1355.92	6	2.73	2.88	1364.09	6	1.48	1.63	1251.03	5	0.73	0.89
L	200	50	2	2	1661.75	3	2.64	2.84	1609.82	4	1.48	1.67	1659.79	2	0.75	0.95
L	200	50	3	2	1530.89	3	2.79	3.02	1701.83	1	1.59	1.83	1605.74	2	0.79	1.02
M	100	25	1	3	18.30	10	6.11	6.11	13.76	10	4.31	4.31	13.69	10	2.10	2.10
M	100	25	2	3	34.41	10	7.07	7.07	50.31	10	4.52	4.52	40.99	10	2.18	2.18
M	100	25	3	3	47.97	10	7.02	7.02	65.58	10	4.73	4.73	47.86	10	2.31	2.31
L	200	50	1	3	1335.39	5	3.73	3.88	980.98	8	2.66	2.81	964.13	8	1.27	1.42
L	200	50	2	3	1626.47	4	3.81	4.00	1293.70	8	2.57	2.76	1407.03	5	1.25	1.45
L	200	50	3	3	1553.53	3	3.88	4.12	1505.83	4	2.64	2.87	1448.62	3	1.40	1.64
M	100	25	1	4	17.24	10	8.21	8.21	15.04	10	6.29	6.29	13.14	10	2.99	2.99
M	100	25	2	4	44.07	10	8.61	8.61	32.23	10	6.18	6.18	25.92	10	2.87	2.87
M	100	25	3	4	33.95	10	8.60	8.60	40.66	10	6.15	6.15	30.93	10	3.43	3.43
L	200	50	1	4	799.56	8	4.69	4.83	1127.07	8	3.69	3.84	904.73	8	1.75	1.90
L	200	50	2	4	1431.26	5	4.76	4.94	1328.29	5	3.79	3.98	1277.40	6	1.88	2.08
L	200	50	3	4	1571.65	3	4.68	4.91	1578.46	3	3.78	4.01	1164.37	7	1.86	2.10
M	100	25	1	5	13.97	10	9.35	9.35	7.77	10	7.57	7.57	9.15	10	3.78	3.78
M	100	25	2	5	24.39	10	9.90	9.90	23.78	10	7.73	7.73	21.52	10	4.03	4.03
M	100	25	3	5	25.98	10	10.04	10.04	19.80	10	7.85	7.85	30.72	10	4.29	4.29
L	200	50	1	5	1154.39	5	5.50	5.65	778.31	8	4.78	4.92	639.24	9	2.20	2.35
L	200	50	2	5	1294.59	4	5.43	5.61	1492.54	5	4.32	4.51	920.90	8	2.23	2.42
L	200	50	3	5	1450.08	5	5.72	5.95	1523.89	5	4.73	4.96	1079.28	6	2.39	2.63

Table B.8.: Performance of interval selection approaches and space reduction level μ for ITWS-CD-I

Table B.8 presents the numerical results of the ITWS-CD-I problem instances representing the cross-docking context. Here, not in all cases an optimal solution is found, which is why we can not compute the optimality gaps. Instead, we report the gap to the best found solution (i.e. upper bound) among all 15 interval scheduling instances solving the same original problem instance (column "gap^{UB}") and the gap to the lower bound (column "gap^{LB}"), found by CPLEX in 30 minutes while applying the exact approach (i.e., $\mu = 1$). While all generated instances from data set M (in total 450) are solved to optimality, only for 204 out of 450 instances from data set L optimal solutions are found.

The smoothed approach shows superior performance also in the cross-docking context. For problem instances from the data set M (i.e., $|J| = 100$, $|D| = 25$, and $|O| = 25$), the average CPU time for the exact approach is under 3 minutes. The average computational time for the larger instances from data set L exceeds 30 minutes, because the solver often struggles to find an optimal solution within the given time limit. Nevertheless, it is remarkable that our exact approach still provides good solutions within the time limit of 30 minutes, the average gap to the lower bound being under 0.2%.

Figure B.3 visualizes the impact of μ on the computational time (left) and solution quality (right) for all three interval selection approaches. The two diagrams on top show the results for the data set M and the bottom diagrams the results for the data set L. The curves for the medium instances support the finding from the distribution center context: computational times decrease approximately exponentially and the optimality gaps rise approximately linearly in the value of μ. The curves of the computational times of instances from data sets L is not characterized by such a clear trend. The computational times of the smoothed approach for the instances from data set L show a decreasing trend with increasing μ values. However, the graphs of the semi-random and random approaches do not show such a clear trend, which can be explained by the coincidental effect during interval selection. In contrast, optimality gaps are characterized by approximately rising linear curves, where the smoothed approach shows the smallest slope. This highlights the dominance of this approach in the context of cross-docking terminals as well. The choice of a suited μ-value has to consider the trade-off between solution quality and computational time.

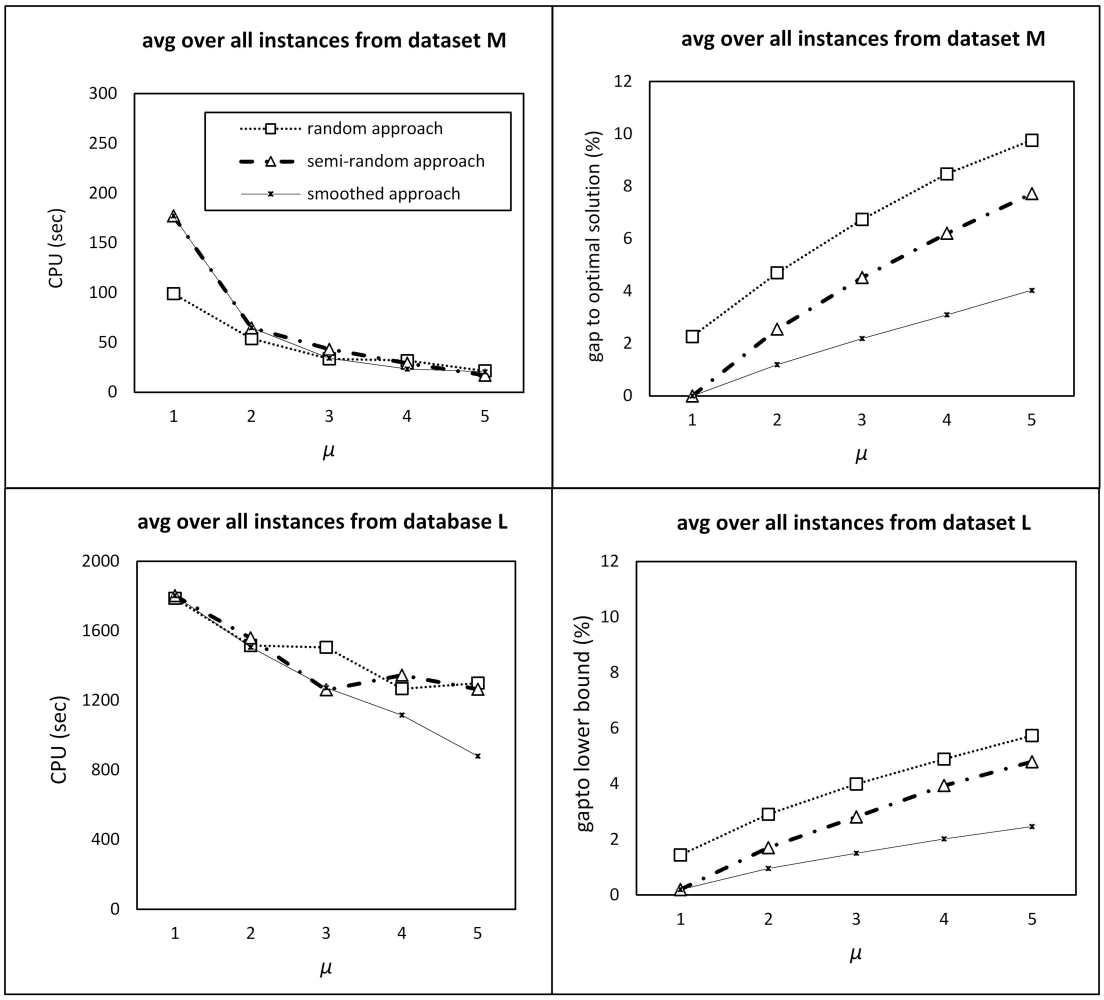


Figure B.3.: Influence of solution space reduction level μ on the solution quality (ITWS-CD-I)

B.5. Effects of integrated planning

To benchmark whether an integrated truck and workforce scheduling leads to considerable improvements over traditional planning approaches, we apply two competitors, which both execute our two planning tasks in a successive manner. First, a workforce planning step assigns workers to docks, so that fixed processing times for the trucks can be derived. Only then, a separate truck scheduling approach is applied. Recall that this methodology is the basic procedure underlying all existing truck scheduling approaches, which invariably presuppose fixed processing times. Specifically, the workforce assignment step can lead to the following two different results:

- *equal workforce* approach: If each dock receives the same number of workers, then the processing times p_j of trucks are fixed and independent of the dock assignment. In other words, the corresponding truck scheduling problem becomes a type of identical parallel machine scheduling problem. Note that this approach requires the total number of available workers to be divisible by the number of inbound docks, i.e., $\frac{|W|}{|D|}$ must be integer.
- *dock-specific workforce* approach: An alternative to the former approach is to assign varying numbers of workers to docks. In this setting, some docks receive a greater workforce, so that processing a truck at these (faster) docks is accelerated. Consequently, in the latter alternative, the processing times p_{jg} depend on the dock g at which truck j is processed. In the machine scheduling literature, this environment is known as parallel related machine scheduling.

The MIP models for the resulting truck scheduling problems as well as the corresponding interval scheduling approaches for both successive approaches and objectives are defined in Appendix B.9.

In this section we compare the three alternative workforce planning approaches (equal, dock-specific and integrated workforce approach) for different settings and scenarios. Specifically, we explore the following managerial insights: Firstly, we investigate the performance improvements of the ITWS over the traditional approaches with fixed workforce. Moreover, for representative problem instances, we illustrate how the number of employed workers varies over time using the three different workforce planning approaches. Finally, we observe the workforce saving potential when instead of the equal- the dock specific- or the integrated workforce planning approach is applied.

B.5.1. Comparison of ITWS with the non-integrated planning approaches

To compare the performance of our integrated planning approach with the two baseline approaches applying a successive planning, we generate new problem instances and solve each instance with all three approaches (i.e., ITWS, equal workforce and dock-specific workforce). For each resulting instance, we reduce the ITWS instances to the corresponding interval scheduling problem by selecting all possible execution intervals.

Problem instances are generated as described in Section B.4.1. The results from Section B.4.3 have shown that the instances from data set M (i.e., with parameters $|J| = 100$, $|D| = 25$, and $|O| = 25$) can be solved optimally by interval scheduling in reasonable time. Thus, we generate medium sized instances with $\Omega_{max} = 2$. Moreover, our instances vary in the size $|W|$ of the available workforce. Recall that the equal workforce approach is applicable only if $|W|$ is divisible by $|D|$ without remainder. To apply all three approaches we, thus, generate instances for two scenarios: $\frac{|W|}{|D|} = 1$, (i.e., $|W| = 25$), dubbed the *scarce workforce* scenario and $\frac{|W|}{|D|} = 2$ (i.e., $|W| = 50$), dubbed the *plentiful workforce* scenario. In the former case, we have the same number of workers than docks, so that our integrated approach can only utilize the maneuvering time at the docks where no worker is momentarily required to gather more than a single worker at another dock in the meanwhile. Both successive approaches can only fixedly assign one worker per dock. In the plentiful workforce scenario, the equal workforce approach fixedly assigns two workers per dock, whereas the other two approaches have additional flexibility. Note that $\frac{|W|}{|D|} \geq 3$ (i.e., $|W| \geq 75$) makes workforce scheduling trivial, because more than three workers per dock cannot increase the processing speed anyway. When benchmarking all three planning approaches, we also investigate the impact of acceleration effect κ on the performance by varying $\kappa \in \{1, 1.25, 1.5, 1.75, 2\}$. Recall that a larger κ value represents additional obstruction among workers and, thus, a smaller acceleration of truck processing with multiple workers. For each resulting parameter combination, we have generated 10 problem instances using our instance generator defined in Section B.4.1. Each of the resulting 100 instances is solved by our three planning approaches in both problem settings (distribution center and cross-docking context) as follows:

- Our integrated planning approach solves truck scheduling and workforce assignment simultaneously by solving the respective exact version of interval scheduling.
- The equal workforce approach, first, plans the workforce by assigning the same number of $\frac{|W|}{|D|}$ workers per dock. Then, truck scheduling for the given workforce

and the resulting processing times is solved by the exact versions of the interval scheduling approaches.

- The dock-specific planning approach fixes a varying number of workers per dock first. In our experiment, we emulate this by assigning workers to docks so that the central doors, which in an I-shaped cross-dock can be assumed to be the most critical, receive more and the extreme doors receive fewer workers. Specifically, we assign three workers to each of the $\frac{|D|}{3}$ central doors, one worker to each of the $\frac{|D|}{6}$ doors at each end of the I-shaped facility, and two workers to each of the remaining doors. Note that experiments with alternative worker assignment schemes have not led to any noteworthy improvement. This assignment results in dock-dependent processing times of trucks and the resulting truck scheduling problems are solved by applying the exact versions of the interval scheduling approaches.

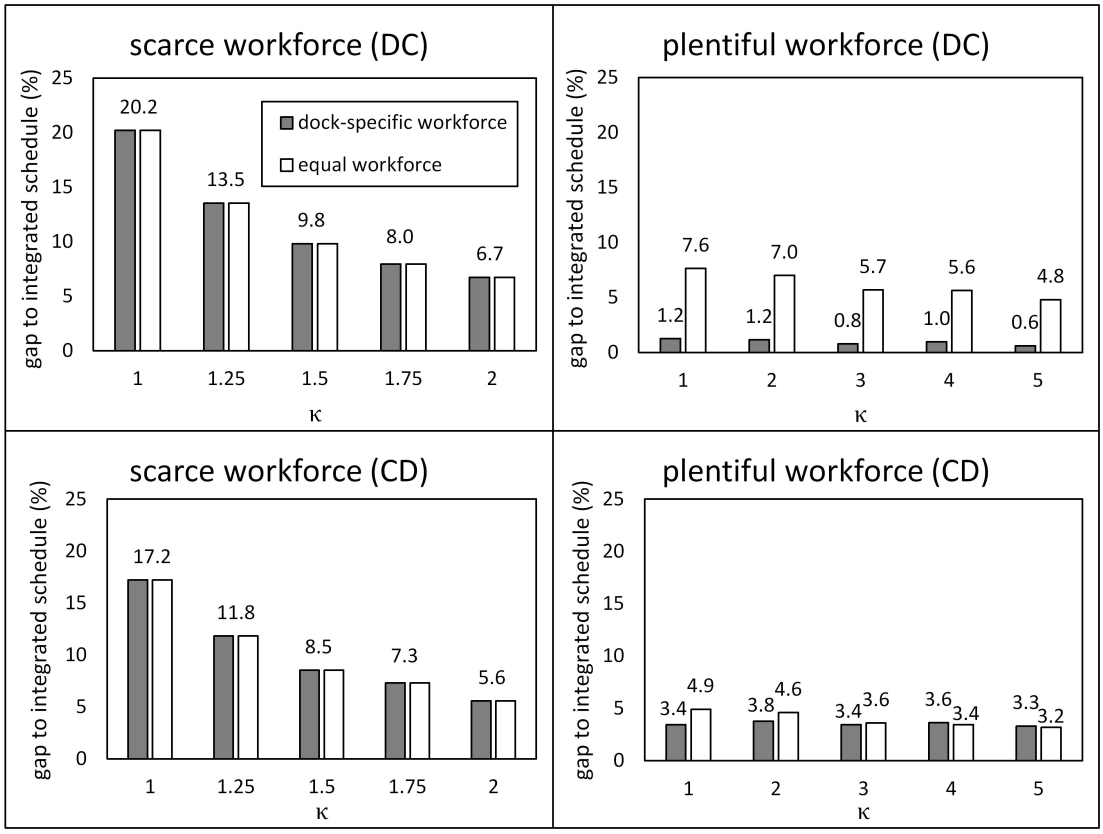


Figure B.4.: Comparison of fixed workforce assignment approach with integrated planning

The results of our experiment are depicted in Figure B.4. Here, we record the relative

gaps of the two successive planning approaches in relation to the solution value of our integrated planning approach depending on the value of κ. Specifically, the relative gaps are calculated by $(F^* - F^{\text{ITWS}})/F^{\text{ITWS}}$, where F^* and F^{ITWS} denote the objective values obtained by the respective successive planning approach and by our integrated truck and workforce scheduling, respectively. The upper two diagrams of Figure B.4 depict the results of the comparison in the distribution center context and the lower two in the cross-docking context. Furthermore, we differentiate between a scarce workforce (left two diagrams) and a plentiful workforce (right two diagrams). The following conclusions can be drawn from these results:

- *Impact of κ:* The results clearly indicate that the performance gains of an integrated truck and workforce scheduling heavily depend on the obstructions of the workers among each other. For $\kappa = 1$, we have a linear increase of the working speed with every additional worker, so that an integrated planning enables a considerably more efficient truck processing. According to our observations in real-world facilities, rather unobstructed collaboration is often possible if two workers unload parcels on the left- and right-hand side of a telescope conveyor pulled directly into the trailer. With three workers unloading parcels or if pallet jacks or forklifts are applied to unload pallets, obstruction is unavoidable, so that $\kappa > 1$ holds. In these cases, the performance gains of an integrated planning are less distinct.

- *Impact of workforce size:* Larger performance gains of an integrated planning approach are realized if the workforce is rather scarce. Recall that in the scarce workforce scenario the number of workers is equal to the number of inbound gates. Consequently, both successive planning approaches lead to identical results. Our holistic planning approach allows to gather more than a single worker at docks. This is less effective if the workforce is plentiful to begin with due to the subadditive performance increase of more than two workers at a dock.

- *Impact of objective:* In the scarce workforce scenario, the performance loss of both successive planning approaches compared to our holistic planning approach is on about the same level for the distribution center (DC) and the cross-docking (CD) setting. We recall that the values of these two objectives are measured in different units (flow time for ITWS-DC and number/weight of delayed shipments for ITWS-CD), which for the corresponding terminals can be associated with different costs. Hence, cost savings cannot be directly inferred from the optimality gaps. In the

plentiful workforce scenario, the dock-specific workforce approach leads to some faster docks, where multiple workers are applied, and other slower docks, where only a single worker unloads trailers. In the distribution center setting, time-critical trucks can deliberately be assigned to faster docks whereas less urgent trucks can be unloaded at slower docks. In this way, the dock-specific workforce approach is nearly as efficient as our integrated truck and workforce scheduling in avoiding waiting times of trucks. In the cross-docking context, this additional flexibility can no longer be exploited by the dock-specific workforce approach. To not miss departing outbound trucks, it is no longer sufficient to assign urgent trucks to faster docks. These faster docks also have to be close to the outbound dock of the outgoing truck. Apparently, this effect can be so distinct that the dock-specific workforce approach is actually no longer clearly superior to equal-workforce approach.

These results emphasize that the right planning approach has to be selected with great care. A holistic approach planning the workforce assignment and truck scheduling jointly may lead to considerable performance gains of about 20% (few worker obstruction among a small workforce), but the additional planning complexity may also lead to nearly no benefit (a lot of obstruction among plenty of workers). Moreover, the dock-specific successive planning approach can nearly be as efficient as our holistic planning approach (plenty workforce in the distribution center context), but can also be outperformed by the classic equal-workforce approach (plenty workforce in the cross-docking context).

To get a better idea of how effectively the three alternative workforce planning approaches utilize the given workforce $|W|$, we observe the distribution of employed workers over time. Figure B.5 depicts the total number of occupied workers for each period of the planning horizon using the three alternative workforce scheduling approaches. For each setting and scenario, we select a representative problem instance from the previous experiment and depict the distribution of utilized workforce over time. In all cases, the workforce utilization rate slightly varies over time, but it is still most effective when the integrated approach is applied. In case of the scarce workforce scenario, where the given workforce is a critical resource, almost always (apart from the beginning and the end of the active phase) nearly 100% of the workforce is utilized by the integrated approach (see both left-hand graphics of Figure B.5). This result no longer holds for the plentiful workforce scenario. Here, the number of busy workers varies too much over time (see both right-hand graphics of Figure B.5). A plentiful workforce allows shorter service times, and for the given truck arrivals, often many workers are superfluous. In practice,

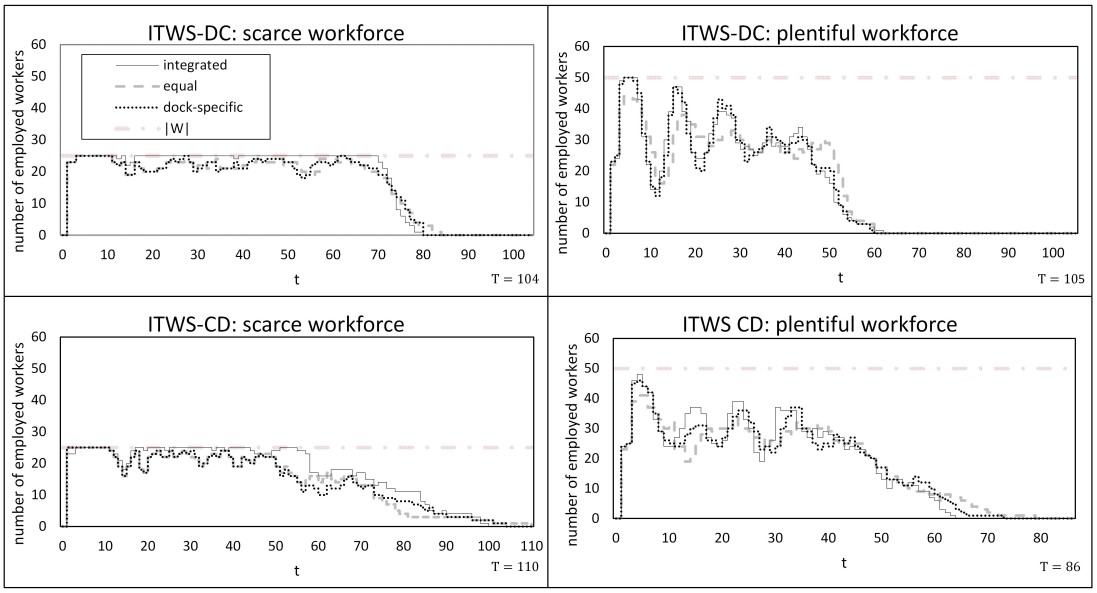


Figure B.5.: Distribution of utilized workforce size over the planning horizon

the proper size of the given workforce clearly depends on the truck arrival frequency and the desired service level. To investigate the trade-off between the workforce size and the truck scheduling performance, we set up an additional experiment.

Recall that the trade-off between the two conflicting objectives of truck scheduling and workforce planning is resolved by assuming a given workforce size and optimizing the truck schedules for a given fixed $|W|$ (see Section B.2). However, for many practical cases, it is also interesting to explore the minimal workforce size $|W'|$ necessary for achieving some predefined truck scheduling performance. In this context, we make a further comparison between our new approaches with workforce dependent processing times (i.e., our holistic approach ITWS and dock-specific) and the classic approach assuming fixed processing times (i.e., equal workforce). For this we generate 10 new instances for each of both problem settings with fixed parameters: $|J| = 100$, $|D| = 25$, $|O| = 25$, and $\kappa = 1.25$. Each instance is solved as follows:

- First, the problem is solved by the benchmark approach (i.e., equal workforce) with the fixed workforce size $|W| = 50$. The optimal value of objective function $F^*_{benchmark}$ is then recorded.
- Subsequently, we look for the minimal number of workers $|W'|^*$ that allows the holistic approach ITWS to reach at least as good a truck scheduling performance

as the benchmark approach does. For this we iteratively solve the problem with varying $|W'|$ via binary search for $|W'| \in \{1, \ldots, |W|\}$. The search terminates when a value $|W'|^*$ is found where $F^*_{ITWS}(|W'|^*) \leq F^*_{benchmark}$ and $F^*_{ITWS}(|W'|^* - 1) > F^*_{benchmark}$.

- Finally, the same search procedure is executed for the dock-specific approach. Recall from our previous tests that the dock-specific approach with $|W| = 50$ must not necessarily lead to a performance increase compared with equal workforce approach with the same workforce size (especially in the cross-docking context, see the lower graphic of Figure B.4). The binary search procedure is only started, if $F^*_{dock-specific}(|W'| = 50) \leq F^*_{benchmark}$.

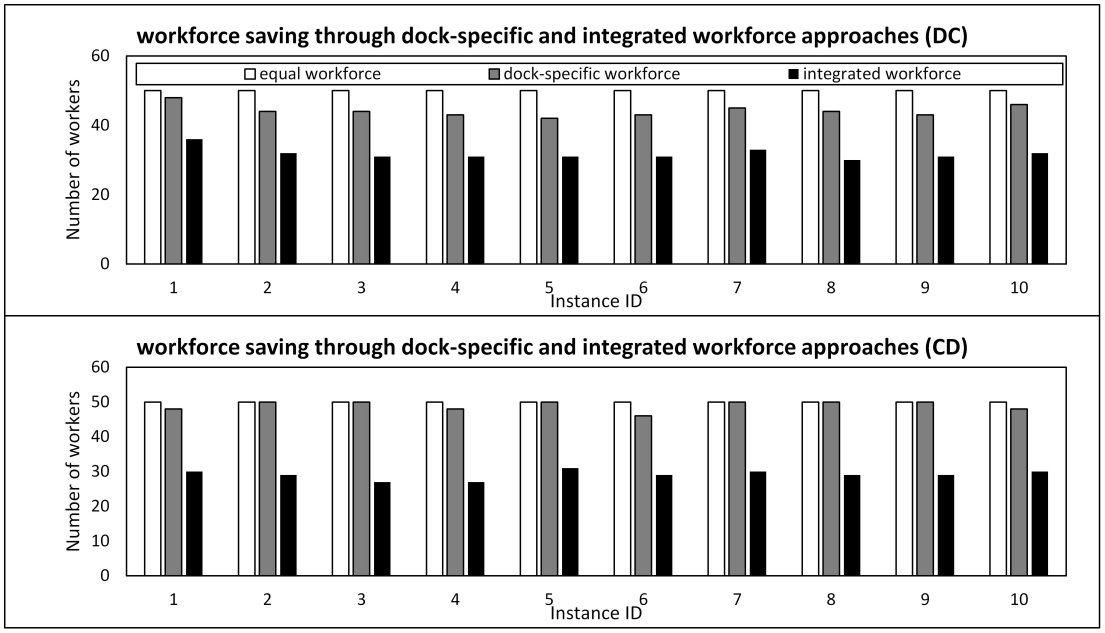


Figure B.6.: Resulting workforce sizes of all three planning approaches for a given performance of truck processing

Figure B.6 visualizes the results for all 10 instances in both the distribution center (DC) as well as the cross-docking (CD) settings. In the distribution center context, both holistic planning approach ITWS and the dock-specific workforce approach enable making significantly better use of the workforce. By applying ITWS, on average, less than 32 workers can match the performance of 50 workers who are equally distributed among the docks. This corresponds to a reduction of the workforce by 36%. The dock-specific approach saves about 12% of workers on average. In the cross-docking context, ITWS allows decreasing the number of workers to ca. 29 workers on average, which

corresponds to a workforce reduction by 42%. For the dock-specific approach, only for the 4 out of 10 instances a tiny decrease of workers is possible; otherwise, the performance is not better than that of the equal workforce approach.

Relating these results to the situation of a typical German car manufacturer stresses the great potential gains of our integrated truck scheduling and workforce planning. According to our discussions with automotive managers, $|D| = 25$ docks for unloading trucks operated by $|W| = 50$ forklift drivers and an obstruction factor of $\kappa = 1.25$ are seen as representative values for a typical mid-sized automotive factory. Given the yearly costs of approximately 250.000 euro per forklift driver these managers calculate with, savings of multiple millions of personnel costs are promised by integrating a workforce planning into the yard management software (e.g., INFORM, 2019) most factories apply anyway.

B.6. Conclusion

This chapter investigates the integrated truck and workforce scheduling problem (ITWS), which does not consider the trucks' processing times at the dock doors as fixed but as dependent on the number of assigned logistics workers. We formalize the problem for two representative settings: for distribution centers where unloaded goods are stored in the terminal at least for some time, and for cross-docking environments where goods need to be moved to outbound trucks quickly. For each setting, we develop a mixed-integer program and prove the complexity status of the problem. For efficiently solving larger instances we transform both versions of ITWS to interval scheduling problems, which allow us to flexibly handle the trade-off between computational time and solution quality by reducing the solution space to the desired level. We develop three approaches for selecting intervals in a heuristic manner.

Our experimental results indicate that for both ITWS versions, the interval scheduling formulation clearly outperforms its original MIP counterpart. Among the three proposed interval selection approaches, the smoothed approach shows superior performance and delivers near-optimal results in short time even for the most difficult instances in our testbed.

From a managerial perspective, integrating truck and workforce scheduling as proposed in this paper is shown to either substantially improve flow times and ease transship-

ment, if the same workforce is applied compared to a successive planning approach. Alternatively, the same unloading performance can be reached by our holistic planning approach with a considerably smaller workforce. An integrated planning is especially valuable whenever the terminal workforce is comparatively scarce and additional workers can significantly speed up truck processing without obstructing each other.

Future research could integrate sequence-dependent walking times of logistics workers into the models, so that the time for changing docks is considered in more detail. Furthermore, the integration of outbound operations may be a valid task for future research. In some cross-docking settings, outbound trucks depart not according to a predefined schedule, but once all dedicated shipments are loaded (see Boysen, 2010). It would be interesting to see whether the considerable gains of an integrated planning approach identified within our problem settings can also be realized there.

Appendix

B.7. An example problem

Consider an example problem with $|J| = 2$ inbound trucks, unloading modes $K_1 = \{1, 2\}$ and $K_2 = \{1, 2, 3\}$, respectively, to be processed at $|D| = 2$ inbound gates and the available workforce size of $|W| = 3$ workers. The arrival times r_j, deadlines $\overline{d}_j$, and the duration of the four processing steps $\alpha_j, \beta_j, \gamma_j$, and δ_j of each truck can be found in Figure B.7(a). Processing times ρ_{jk} of truck j in unloading mode $k \in K_j$ are given in Figure B.7(b).

j	r_j	$\overline{d}_j$	α_j	β_j	γ_j	δ_j
1	0	16	1	2	2	1
2	2	17	1	3	2	1

(a) time windows and duration of processing steps

ρ_{jk}	1	2	3
1	10	8	-
2	13	11	10

(b) processing times

ϕ_{jo}	1	2
1	1	4
2	5	2

(c) weights of shipments

φ_{go}	1	2
1	2	3
2	3	2

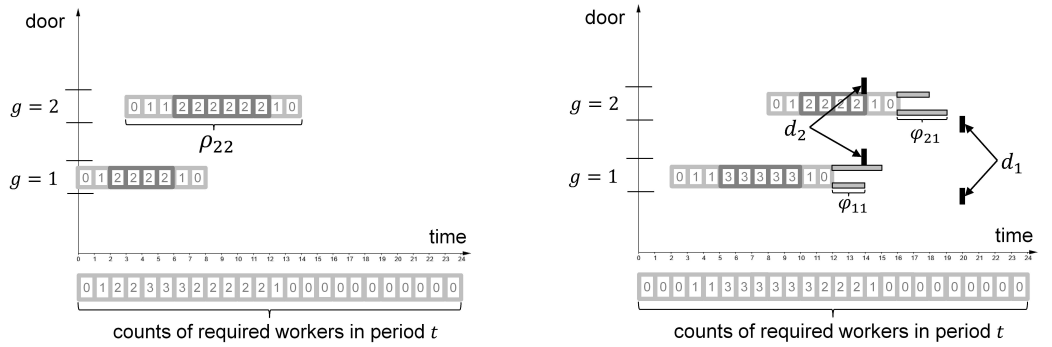
(d) transfer times between doors

Figure B.7.: Input data for the example problem.

For this problem, the optimal solution with regard to objective (B.1) is to assign truck 1 to gate 1 to be completed in period 8, unloaded by two workers, and assign truck

2 to gate 2 with completion time 14 and unloading mode 2. This leads to flow times $C_1 - a_1 = 8$ and $C_2 - a_2 = 12$ and hence to a total objective value of $\sum_{j \in J}(C_j - r_j) = 20$. This schedule is depicted as a Gantt chart in Figure B.8(a).

To extend this problem for the cross-docking environment, we further consider a set of outbound trucks $O = \{1, 2\}$ with departure times $d_1 = 14$ and $d_2 = 20$. The outbound truck's related data – the weight Φ_{jo} of a shipment delivered by inbound truck j dedicated to outbound truck o and the transfer time φ_{go} for moving goods from dock g to outbound truck o – are summarized in Tables B.7(c) and B.7(d), respectively. Using objective (B.14), the optimal solution is depicted in Figure B.8(b), implying the following total delayed shipments of the inbound trucks: $\sum_{o \in O} \Phi_{1o} \cdot z_{1o} = 1$ and $\sum_{o \in O} \Phi_{2o} \cdot z_{2o} = 2$. Thus, the objective value is $\sum_{j \in J} \sum_{o \in O} \phi_{jo} \cdot z_{jo} = 3$.



(a) Gantt Chart for ITWS in the distribution center context

(b) Gantt Chart for ITWS in the cross-docking context

Figure B.8.: Optimal schedules for two different objective for example data

B.8. Interval selection

The interval selection for truck 2 from the example from Appendix B.7 for the three approaches (random, semi-random and smoothed) and three different value of $\mu \in \{1, 2, 3\}$ are illustrated in Figures B.9, B.10 and B.11, respectively.

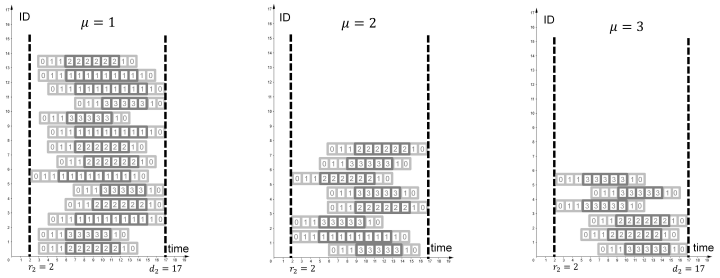


Figure B.9.: Selected intervals by random approach for different values of μ.

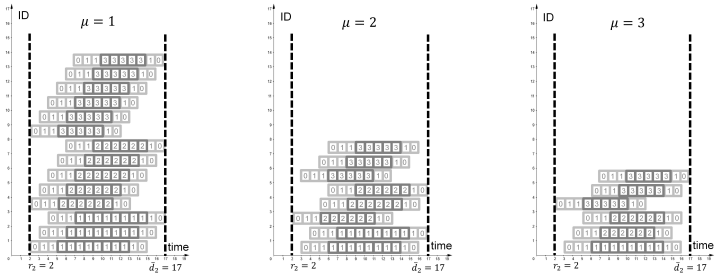


Figure B.10.: Selected intervals by semi-random approach for different values of μ.

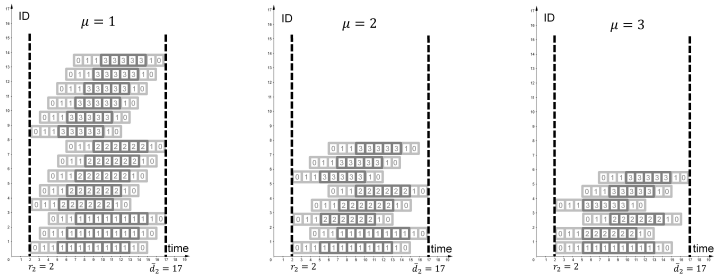


Figure B.11.: Selected intervals by smoothed approach for different values of μ.

B.9. MIPs for the benchmark problems with fixed workforce

The MIP model for the equal workforce approach in the distribution center context consists of objective function (B.1) and constraints (B.6) - (B.12). In the cross-docking context, the corresponding MIP consists of objective function (B.14) and constraints (B.15) - (B.21).

The interval scheduling formulation in the distribution center context consists of objective function (B.23) subject to constraints (B.24), (B.26), and (B.27). For the cross-docking setting, interval scheduling formulation consists of objective function (B.29) subject to constraints (B.30), (B.32), and (B.33).

A dock-specific workforce assignment leads to fixed processing times p_{jg} for each truck j at dock g. The resulting truck scheduling MIP model for the distribution center context consists of objective function (B.1) subject to constraints (B.10), (B.11), (B.15) - (B.18), (B.20) and (B.34). For the cross-docking setting the corresponding MIP consists of objective function (B.14) subject to constraints (B.10), (B.11), (B.15) - (B.18), (B.20), and (B.34).

$$C_j \geq C_i + p_{jg} - M \cdot (1 - x_{ij}^g) \quad \forall\, j \in J;\, i \in J \cup \{0\};\, g \in D \tag{B.34}$$

For the interval scheduling, sets M_{jg} of dock-specific intervals have to be defined, because, now, the intervals resulting from a specific number of workers can only be executed at a dock where the respective number of workers is actually available. The corresponding weights of the intervals in the distribution center context can be computed with Equation (B.35) and in the cross-docking context with (B.36).

$$\tilde{w}_{jmg} = C_{jmg} - r_j \quad \forall\, j \in J;\, g \in D;\, m \in M_{jg}. \tag{B.35}$$

$$\bar{w}_{jmg} = \sum_{o \in O} \phi_{jo} \cdot \begin{cases} 1 & \text{if } C_{jmg} + \varphi_{go} > d_o \\ 0 & \text{otherwise,} \end{cases} \quad \forall\, j \in J;\, g \in D;\, m \in M_{jg} \tag{B.36}$$

The resulting interval scheduling model consists of objective function (B.37) subject to constraints (B.38) - (B.40).

In the cross-docking setting, the interval scheduling approach consists of objective func-

$$\text{Minimize } F(X) = \sum_{j \in J} \sum_{g \in D} \sum_{m \in M_{jg}} \tilde{w}_{jmg} \cdot x_{jmg} \tag{B.37}$$

subject to

$$\sum_{g \in D} \sum_{m \in M_{jg}} x_{jmg} = 1 \qquad \forall\, j \in J \tag{B.38}$$

$$\sum_{j \in J} \sum_{m \in M_{jg}} b_{jmgt} \cdot x_{jmg} \leq 1 \qquad \forall\, t = 1, \ldots, T;\ g \in D \tag{B.39}$$

$$x_{jmg} \in \{0, 1\} \qquad \forall\, j \in J;\ g \in D;\ m \in M_{jg} \tag{B.40}$$

tion (B.41) subject to constraints (B.38) - (B.40).

$$\text{Minimize } F(X) = \sum_{j \in J} \sum_{g \in D} \sum_{m \in M_{jg}} \bar{w}_{jmg} \cdot x_{jmg} \tag{B.41}$$

Chapter C.

Outbound truck scheduling*

Abstract: This chapter addresses the operational planning problem of loading and scheduling outbound trucks at a dispatch warehouse shipping goods to several customers. The problem consists of, first, assigning shipments to different outbound trucks considering the trailers' capacities and, second, scheduling the trucks' processing at the dock doors such that the amount of required resources at the terminal (e.g., dock doors and logistics workers) does not exceed the available levels. The trucks should be scheduled as late as possible within their time windows. Such planning problems arise, e.g., at dispatch warehouses of automotive parts manufacturers supplying parts to original equipment manufacturers (OEMs) in a just-in-time or even just-in-sequence manner. We formalize this operational problem and provide two mixed-integer linear programming models. Moreover, we develop an exact branch-and-price algorithm, which is shown to perform very well, solving most realistically sized problem instances to proven optimality within a few minutes. In a numerical study, we find evidence that too small a workforce in the dispatch warehouse can substantially compromise the punctuality of the deliveries. Finally, we also look into the interplay between time window management and just-in-time deliveries.

*This chapter is based on working paper Tadumadze, G., Emde, S. (2020): Loading and scheduling outbound trucks at a dispatch warehouse. *Working Paper TU Darmstadt.*

C.1. Introduction

Many manufacturing companies are nowadays part of just-in-time supply chains. This requires, on the one hand, that production programs and distribution networks must be planned such that parts and sub-assemblies are ready to be shipped as on-time as possible (e.g., Erengüç et al., 1999, Li et al., 2006). On the other hand, on an operational level, it must also be ensured that finished goods are loaded quickly and punctually onto outbound trucks at the loading dock of the dispatch warehouse of the manufacturing plant. This latter aspect has received little attention in the literature so far.

This research is inspired by the dispatch of goods that we observed at a manufacturer of trailer couplings for trucks and utility vehicles. However, our approaches can be also be applied at other cargo handling facilities, like cross-docks in a many-to-few network configuration (Buijs et al., 2014). Most of these couplings are shipped batch-wise to original equipment manufacturers (OEM) to be used in their production processes. The company ships orders just-in-time or even just-in-sequence, depending on the OEM's needs. Consequently, parts should be dispatched as late as possible but not past their deadline, because otherwise, steep contractual penalties apply.

Orders are usually produced several days in advance and then stored in a dispatch warehouse. Through their online time window management system, the company has some degree of control over when trucks bound for specific destinations arrive to be loaded. The company is also solely responsible for actually loading the trucks and deciding what goes on them. When loading palletized orders onto trucks, the limited carrying capacity of the trailer must be considered. Moreover, since the number of dock doors is limited, and trucks cannot be asked to dock indefinitely, the loading process must be executed in a speedy manner. Dispatching an order entails fetching the corresponding parts from storage, possibly sorting, packaging, and labeling them (in case of just-in-sequence shipments) and loading them onto the truck. Depending on the size of the order, this may require multiple workers, the total supply of which is limited. The dispatch area is schematically depicted in Figure C.1.

In this context, we solve the following problem, which we dub the *outbound truck scheduling and loading problem* (OTSLP). Given a set of trucks with varying capacities and earliest and latest departure times, and a set of items with varying sizes, which item should be loaded onto which truck, and when should each truck's departure be scheduled? Loading items takes manpower, of which only a limited quantity is available;

therefore, care must be taken when multiple items are loaded onto trucks concurrently. Moreover, items have given deadlines by which they must be dispatched. Items may be dispatched early, but the earliness is penalized as it violates the just-in-time principle and may entail contractual penalties (demurrages) or loss of goodwill. The goal is to minimize the total weighted earliness.

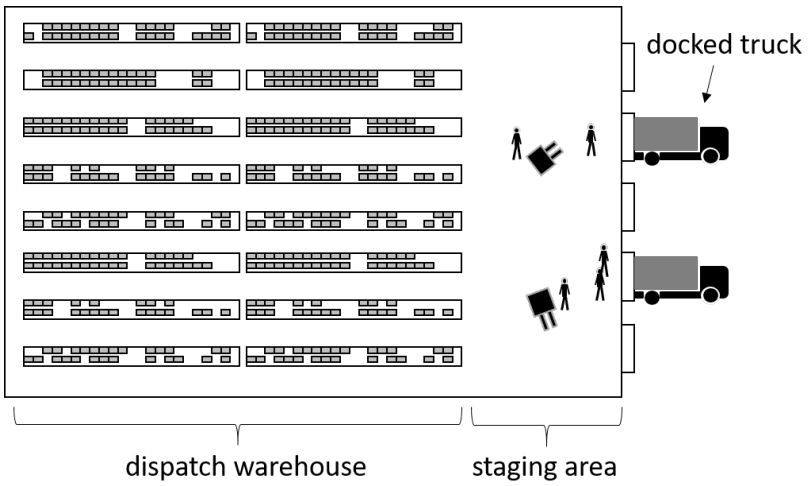


Figure C.1.: Schematic depiction of the dispatch area.

The contribution of this paper is as follows. We present the novel problem of planning the loading and scheduling of outbound trucks at a dispatch warehouse. To the best of our knowledge, this problem has not yet been addressed in the literature. We formalize the problem and provide two different mixed-integer linear programming models. Moreover, we develop an exact branch-and-price algorithm, which is shown to solve realistic instances in an acceptable time, far outperforming a default solver. Finally, we derive some managerial insights into the optimal operation of a dispatch area, in particular, the connection between the logistics workforce and punctuality of deliveries, and the use of effective time window management.

The remainder of this paper is organized as follows. In Section C.2, we review the literature. Section C.3 formally defines the problem and provides two mixed-integer linear programming models. An exact branch-and-price scheme is presented in Section C.4, which is tested in a computational study (Section C.5). Section C.6 concludes the paper.

C.2. Literature review

For a general overview of part delivery in (just-in-time) supply chains, we refer to survey papers on decision problems integrating outbound distribution into production processes and on part logistics in the automotive industry provided by Chen (2010) and Boysen et al. (2015), respectively. Furthermore, the spatial structure of (JIT)-based logistics for the distribution of automotive parts is elucidated by Kaneko and Nojiri (2008). The latter study is focused on the Japanese automotive industry, but we can expect similar networks in just-in-time supply chains all over the world.

Most existing research about just-in-time part logistics deals with routing problems of trucks from a single / multiple suppliers to a single / multiple receivers in a point-to-point network (e.g., Emde and Zehtabian, 2019, Gschwind et al., 2020), a milk-run system (e.g., Sadjadi et al., 2009) or a cross-docking system (e.g., Li et al., 2004, Buijs et al., 2016). Fliedner et al. (2016) investigate a vehicle scheduling problem between a single supplier and a receiver under the warehouse-on-wheels policy, where a truck also serves as a movable warehouse since it remains docked at the door of the OEM plant until all delivered parts are demanded at the assembly line. Hence, although the general setting of their problem is similar to ours, their problem structure is completely different. Another related work, dealing with the problem of delivering different types of parts from suppliers to an automotive assembly plant, is provided by Wang and Chen (2019). However, their problem is from the viewpoint of the OEM and deals with delivering parts from several suppliers (located close to each other in a supplier park) to a single OEM plant, whereas we observe the problem from the viewpoint of a part manufacturer supplying components to several OEMs. Thus, again, the structure of the derived models completely differs from ours.

A baseline problem of ours is the one discussed in Boysen et al. (2016). Their problem consists of two decisions: one, assigning a set of orders (to be delivered from a single supplier to a single receiver in a JIT manner) to outbound trucks such that the trucks' capacity is not violated and, two, deciding the departure time of trucks so that no item misses its deadline. The authors investigate two versions of the planning problem, which mainly differ with regard to their objective functions. In the first version of the problem (dubbed $min\#D$), the number of trucks is minimized, while in the second problem version ($min \sum E_j$), the aim is to minimize total earliness of the items for a given truck fleet. Another work dealing with the scheduling of outbound trucks for just-in-time

delivery from a single supplier (i.e., cross-docking terminal) to a single customer (i.e., OEM plant) is provided by Schwerdfeger et al. (2018). Minimizing the number of trucks while allowing the same trucks to sequentially execute multiple tours, their model can be seen as a generalization of the first model variant of Boysen et al. (2016) (i.e., $min\#D$), while our model is a generalization of the second model variant $min \sum E_j$. Specifically, our model allows orders with *different earliness costs* to be shipped from a single supplier to *several* (instead of one) customers. Further, our model considers the truck scheduling problem at the terminal in more detail: we take the *trucks' time windows* into account and take care that the required resources (i.e., *dock doors* and *logistics workers*) do not exceed the available limits.

Limited resources at a truck terminal are also considered by Tadumadze et al. (2019). They provide two variants of the integrated truck and workforce scheduling (ITWS) problem and also take the available workforce and dock door limits into account. However, the authors observe inbound operations (i.e., unloading of incoming trucks), while we deal with the outbound operations. Thus, their problem does not contain the item assignment component. Moreover, both of their objectives ($\sum F_j$ and $\sum w_j U_j$) differ from ours ($\sum w_i E_i$). This way, our outbound truck scheduling and loading problem can be seen as a combination of the $min \sum E_j$ problem variant observed by Boysen et al. (2016) and the ITWS problem provided by Tadumadze et al. (2019), which, to the best of our knowledge, has not been considered in the literature yet.

C.3. Problem description

We formally describe the OTSLP and discuss model assumptions in Sections C.3.1 and C.3.2, respectively. An illustrative example of an OTSLP instance is presented in Section C.3.3. To enable the use of default solvers and as a foundation for our branch-and-price approach, we propose three alternative mixed-integer linear programming models (MILP) in Section C.3.4.

C.3.1. Formal problem description

Let $B = \{1, \ldots, n\}$ be the set of trucks onto which items $I = \{1, \ldots, m\}$ are to be loaded at $D \in \mathbb{Z}^+$ loading docks. In this context, an "item" may be an entire shipment or order bound for a specific customer consisting of one or multiple pallets of individual SKUs.

Since trucks may be bound for different destinations, an item $i \in I$ can only be loaded onto the trucks in set $B_i \subseteq B$. An item i takes up $w_i \in \mathbb{Z}^+$ units of space on a truck, and it cannot leave the facility later than deadline $d_i \in \mathbb{Z}^+$. Early departure before the deadline is possible, but penalized by a factor of $r_i \in \mathbb{Z}^+$ per time unit of earliness. Each truck j is associated with a capacity $c_j \in \mathbb{Z}^+$, depending on the truck type (e.g., 20t vs. 40t). We assume that the required space units of the orders and carrying capacities of the trucks are measured one-dimensionally. At the manufacturer of truck couplings we visited, ordered couplings are first packed batch-wise onto standardized pallets, of which only a limited quantity can be loaded into a trailer; hence space is almost always the only limiting factor of truck capacity.

Truck arrivals can be scheduled by the warehouse operator within certain limits, i.e., the earliest departure time of truck j is $a_j \in \mathbb{Z}^+$ and the latest is $b_j \in \mathbb{Z}^+$. However, once a truck is scheduled in a fixed time window, it is docked for exactly $p_j \in \mathbb{Z}^+$ time units; this is the time that the trailer is effectively docked at one of the $D \in \mathbb{Z}^+$ loading docks of the terminal and actual loading work is being done. Since keeping trailers for longer than the agreed-upon processing time at the dock may entail demurrages, it is avoided. Moving item i from storage onto a truck uses up $q_i \in \mathbb{Z}^+$ units of a renewable resource during the processing time p_j. Resources may be such things as industrial trucks or forklifts; at the facility we visited, the bottleneck resource is the workforce. Hence, q_i denotes the number of workers necessary to load item i. Since shipments need to be retrieved from storage, labeled, and sometimes sorted (*just-in-sequence*), several employees may be engaged with readying just one order. The maximum number of workers available at any moment is given as $Q \in \mathbb{Z}^+$.

A schedule consists of a set S of tuples (i, j), denoting that item $i \in I$ is assigned to truck $j \in B_i$, and a vector $(\tilde{t}_1, \ldots, \tilde{t}_n)$, such that $\tilde{t}_j \in [a_j, b_j]$ is the departure time of truck $j \in B$. Note that this implies that truck j is docked during the interval $[\tilde{t}_j - p_j + 1, \tilde{t}_j]$. We say that a schedule is feasible if and only if it satisfies the following conditions.

- Each item is assigned to exactly one truck, i.e., for each $i \in I$, there is exactly one tuple $(i, j) \in S$.
- No item misses its deadline, i.e., for each tuple $(i, j) \in S$, it holds that $d_i \geq \tilde{t}_j$.
- No truck is overloaded, i.e., for each truck $j \in B$, it holds that $\sum_{\substack{i \in I: \\ (i,j) \in S}} w_i \leq c_j$.
- At no time are more resources used than are available. Let $\tilde{B}(\tau) \subseteq B$ be the set of

trucks which are being processed at time τ, i.e., $\tilde{B}(\tau) = \{j \in B \mid \tilde{t}_j \geq \tau > \tilde{t}_j - p_j\}$. Then, for each point in time $\tau \in \{\tilde{t}_1, \ldots, \tilde{t}_n\}$, $\sum_{\substack{(i,j)\in S: \\ j\in\tilde{B}(\tau)}} q_i \leq Q$ must hold.

- Similarly, at no time the number of docked trucks can exceed the total number of available docks D, i.e., for each $\tau \in \{\tilde{t}_1, \ldots, \tilde{t}_n\}$, $|\tilde{B}(\tau)| \leq D$ must hold.

At the supplier plant we visited, most items are to be shipped just-in-time or even just-in-sequence. Ideally, items should be shipped no sooner than when they come due because otherwise, they have to be stored on the truck or at the OEM's plant, which is only possible to a very limited extent and may entail contractual penalties for the supplier. Consequently, we consider a schedule optimal if it minimizes the total weighted earliness of all items, i.e.,

$$\sum_{(i,j)\in S} r_i \cdot (d_i - \tilde{t}_j). \tag{C.1}$$

Note that the relative importance of item r_i may express the urgency of sticking to the desired deadline as closely as possible. E.g., just-in-time and especially just-in-sequence items may have great weight, whereas other items may receive a lower weight.

As for computational complexity, OTSLP is clearly an NP-hard problem. Even if all weights r_i are zero, all deadlines and time windows are non-binding ($d_i = \infty$, $\forall i \in I$, $a_j = 0, b_j = \infty$, $\forall j \in B$), and resources are plentiful ($D = Q = \infty$), merely assigning items to trucks such that their capacity is not exceeded is tantamount to solving a variable-sized bin packing problem, which is well-known to be strongly NP-hard (Friesen and Langston, 1986). This observation leads to the following proposition.

Proposition C.1. *Deciding the feasibility of OTSLP is NP-hard in the strong sense.*

C.3.2. Model assumptions

Like many other models, our model is based on some simplifying assumptions, which we discuss below:

(A1): Earliest and latest departure times (a_j and b_j) for each truck $j \in B$ as well as the duration of its docking time p_j are deterministically known.

Nowadays, many truck terminals apply an online time window management system allowing logistics providers to book a desired processing time window beforehand (Karaenke et al., 2019). This way, the data about truck arrival and departure times can be au-

tomatically transferred to the yard management information system. Furthermore, in many practical applications, the docking time for each truck is predefined according to its size (i.e., the bigger the trailer, the longer the docking time). Note that the consideration of given time windows and fixed docking (or processing) times of the trucks is a widely used setting in the truck scheduling literature (Boysen and Fliedner, 2010).

(A2): We calculate the earliness of an order as the difference between its deadline and the completion time of the respective truck's docking.

This presupposes that the item deadlines d_i already include (deterministic) transportation times between the shipping warehouse and the receiving manufacturing plants. In the real world, the exact transportation time may depend on many factors, such as distance, traffic situation, specific vehicle route, etc., all of which are out of the scope of our work. This simplification seems pardonable, especially in the automotive industry, where dispatch warehouses inside the JIT corridor imply short distances and, thus, lead to predictable transportation times (Boysen et al., 2016, Schwerdfeger et al., 2018).

Note that if a truck is released early, the delivery may still arrive at the OEM on-time (i.e., not early) if the truck simply waits on a parking lot in the meantime. However, this merely shifts the problem of earliness to the logistics provider, leading to poor truck utilization. Clearly, for the JIT supply chain as a whole to work well, items should be moved from stage to stage (i.e., from supplier to OEM in this case) as late as possible.

(A3): The available resource limits for loading operations (i.e., dock doors and logistics workers) are assumed to be given and fixed.

Assuming given resource limits for the loading operations implies that inbound and outbound operations are planned separately, and certain numbers of outbound doors and workers are reserved exclusively for the outbound operations. Note that both practical and academic applications of truck scheduling often consider inbound and outbound operations to be separated (Ladier and Alpan, 2016, Boysen and Fliedner, 2010).

(A4): The workers assigned to load a truck are bound to the truck during its whole docking time and afterwards can switch to another truck instantaneously.

This simplifies staff scheduling significantly, since the work content does not need to be broken down into individual tasks and walking times between doors for specific workers, but carries the risk that some workers are still considered busy loading a truck even though they are actually already done with their particular task. While drawing up

specific schedules for each worker would undoubtedly be a worthwhile endeavor, it would create additional practical problems such as forecasting ahead of time exactly how many minutes each individual process step for preparing a specific order will take. Given the relatively short walking distances in the dispatch area that we visited, we decided against such a fine-grained modeling approach. Note that after solving OTSLP (i.e., determining the item assignment and departure time for each truck), the problem of assigning specific loading tasks to each individual worker, accounting for the sequence-dependent setup (walking) times, can be reduced to finding a perfect matching in a bipartite graph that can be solved in polynomial time (Tadumadze et al., 2020a).

(A5): We assume that all parameters have integer values.

Note that this can always be imposed by scaling any real-valued parameters to integer values to arbitrary precision. E.g., if the docking time of some truck is "0.5 hours", we can redefine a time unit to correspond to half an hour and adjust all other time-related parameters accordingly.

C.3.3. Example of a schedule

Consider an example problem instance with $m = 6$ items to be loaded onto $n = 3$ trucks at $D = 2$ doors by $Q = 5$ logistics workers. The item related parameters (required space w_i, deadline d_i, penalty cost per time unit of earliness r_i, required workforce q_i, and set of available trucks B_i for each item $i \in I$) are presented in Table C.1(a). The truck related parameters (processing time p_j, capacity of trailer c_j, as well as the earliest a_j and latest possible departure date b_j of each truck $j \in B$) are listed in Table C.1(b).

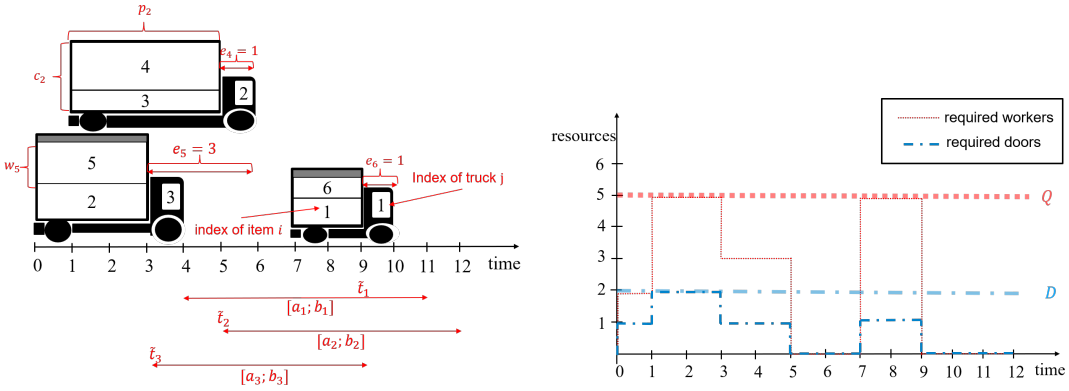
i	1	2	3	4	5	6
w_i	4	5	3	7	6	3
d_i	9	3	5	6	6	10
r_i	4	11	1	12	7	2
q_i	2	1	1	2	1	3
B_i	{1,2,3}	{1,2,3}	{2,3}	{1,2,3}	{1,3}	{1,2}

(a) Item characteristics

j	1	2	3
c_j	8	10	12
p_j	2	4	3
a_j	4	5	3
b_j	11	12	9

(b) Truck characteristics

Table C.1.: Input data for the example problem.



(a) truckloads and departure times. (b) resource consumption.

Figure C.2.: Optimal solution of example problem.

For this problem instance, the optimal solution consists of tuples

$$S^* = \{(1,1),(2,3),(3,2),(4,2),(5,3),(6,1)\}$$

and vector $\tilde{t^*}_j = (9,5,3)$. This means that the first truck ($j = 1$) transports items 1 and 6 and departs from the terminal in period 9, the second truck ($j = 2$) with departure time 5 transports items 3 and 4, and items 2 and 5 are loaded onto the third truck ($j = 3$) which departs in period 3. This schedule causes earliness of items 4, 5 and 6, namely $e_4 = 1$, $e_5 = 3$ and $e_6 = 1$ (and $e_1 = e_2 = e_3 = 0$). The total weighted earliness is $(4+11+1)\cdot 0+12\cdot 1+7\cdot 3+2\cdot 1 = 35$. The optimal truckloads and departure times for each truck $j \in \{1,2,3\}$ are depicted in Figure C.2(a), while Figure C.2(b) visualizes the required amount of resources (i.e., dock doors and logistics workers) for each period of the planing horizon.

C.3.4. Mixed-integer linear programming models

We propose three different mixed-integer linear programming models (MILP) for the OTSLP. In Sections C.3.4.1 and C.3.4.2, we present two alternative MILP formulations, using linear ordering and discrete-time frameworks, respectively, both of which are popular in the literature for modelling difficult scheduling problems (e.g., Baptiste and Sadykov, 2009). Our computational experience (see Section C.5) reveals, however, that neither of these models can be solved by a current default solver to optimality in acceptable time for realistic instances. Therefore, we reformulate our problem as a gen-

eralized set covering problem, which we present in Section C.3.4.3 and later use as the basis of a column generation scheme.

C.3.4.1. Linear ordering model

Our linear ordering mixed-integer linear programming model (denoted as MILP-LO) consists of objective function (C.2) and Constraints (C.3) - (C.12). The applied notation is summarized in Table C.2.

B	Set of trucks (indices j, j')
I	Set of items (index i)
B_i	Set of available trucks for item i ($B_i \subseteq B$)
w_i	Capacity (i.e., weight) of item i
d_i	Deadline of item i
r_i	Penalty cost per time unit of earliness for item i
c_j	Capacity of truck j
a_j	Earliest departure time of truck j
b_j	Latest departure time of truck j
p_j	Docking time (i.e, duration of loading) of truck j
q_i	Number of workers required to move item i onto a truck during its docking time
Q	Total number of available workers
D	Total number of available dock doors
$\mathcal{M}$	Big integer
x_{ij}	Binary variable: 1, if item i is loaded onto truck j; 0, otherwise
$\mathcal{X}$	Set of x variables: $\mathcal{X} = \{x_{ij} \mid i \in I, j \in B\}$
$y_{jj'}$	Binary variable: 1, if truck j departs after truck j' has docked; 0, otherwise
$\mathcal{Y}$	Set of y variables: $\mathcal{Y} = \{y_{jj'} \mid j, j' \in B\}$
t_j	Continuous variable: departure time of truck j; 0, otherwise
$\mathcal{T}$	Set of t variables: $\mathcal{T} = \{t_j \mid j \in B\}$
z_i	Continuous variable: earliness of item i
$\mathcal{Z}$	Set of z variables: $\mathcal{Z} = \{z_i \mid i \in I\}$

Table C.2.: Notation for the linear ordering MILP model.

$$\text{Minimize } \mathcal{F}(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}) = \sum_{i \in I} r_i \cdot z_i \tag{C.2}$$

subject to

$$\sum_{j \in B_i} x_{ij} = 1 \qquad \forall i \in I \tag{C.3}$$

$$\sum_{\substack{i \in I: \\ j \in B_i}} w_i \cdot x_{ij} \leq c_j \qquad \forall j \in B \tag{C.4}$$

$$t_j \leq d_i + (1 - x_{ij}) \cdot M \qquad \forall i \in I, j \in B_i \tag{C.5}$$

$$t_j \leq t_{j'} - p_{j'} + y_{jj'} \cdot M \qquad \forall j, j' \in B \tag{C.6}$$

$$d_i \cdot x_{ij} - t_j \leq z_i + (1 - x_{ij}) \cdot M \qquad \forall i \in I, j \in B_i \tag{C.7}$$

$$\sum_{j \in B'} \sum_{i \in I} q_i \cdot x_{ij} \leq Q + \sum_{j \in B'} \sum_{j' \in B'} (2 - y_{jj'} - y_{j'j}) \cdot M \qquad \forall j \in B, j' \in B', B' \subseteq B \tag{C.8}$$

$$\sum_{j \in B'} x_{ij} \leq D + \sum_{j \in B'} \sum_{j' \in B'} (2 - y_{jj'} - y_{j'j}) \cdot M \qquad \forall i \in I, j \in B, j' \in B', B' \subseteq B \tag{C.9}$$

$$a_j \leq t_j \leq b_j \qquad \forall j \in B \tag{C.10}$$

$$x_{ij} \in \{0; 1\} \qquad \forall i \in I, j \in B \tag{C.11}$$

$$y_{jj'} \in \{0; 1\} \qquad \forall j, j' \in B \tag{C.12}$$

Objective function (C.2) minimizes the total weighted earliness over all items. Constraints (C.3) ensure that each item $i \in I$ is loaded exactly onto one truck. Constraints (C.4) take care that no truck $j \in B$ is overloaded. Constraints (C.5) guarantee that no item misses its deadline, i.e., if item i is loaded onto truck j ($x_{ij} = 1$), then truck j cannot depart later than d_i. Constraints (C.6) set the value of decision variable $y_{jj'}$ to 1, if truck j departs after another truck j' has started docking. This way, for each pair of trucks $j, j' \in B$ with overlapping processing times, $y_{jj'} + y_{j'j} = 1$ holds. Constraints (C.7) define the earliness z_i of each item $i \in I$ as the difference between its deadline d_i and the departure time of its truck t_j. Constraints (C.8) ensure that in no period of the time horizon, the amount of required resources exceeds the total amount of available resources Q. Similarly, Constraints (C.9) take care that at no time the number of simultaneously docked trucks exceeds the number of available dock doors D. Constraints (C.10) guarantee that each truck j departs in its time window (i.e., no earlier than a_j and no later than b_j). Finally, Constraints (C.11) - (C.12) define the domain of the variables.

C.3.4.2. Time-indexed model

The number of Constraints (C.8) and (C.9) in model MILP-LO grows exponentially with the input size. In this section we propose an alternative discrete time model formulation for OTSLP with a pseudo-polynomial number of variables and constraints. To construct the model concisely, we divide the time horizon into $T = \{t \in \mathbb{Z}^+ \mid \min_{j \in B}\{a_j\} \leq t \leq \max_{j \in B}\{b_j\}\}$ discrete periods. Note that since we assume that all parameters are integer, this can be done without loss of precision.

Applying the slightly altered notation, summarized in Table C.3, the time-indexed MILP model (denoted as MILP-TI) consists of objective function (C.13), subject to Constraints (C.14) - (C.21).

T	Set of time periods, index $\tau \in T = \{\tau \in \mathbb{Z}^+ \mid \min_{j \in B}\{a_j\} \leq \tau \leq \max_{j \in B}\{b_j\}\}$
$x_{ij\tau}$	Binary variable: 1, if item i is loaded onto truck j and leaves the warehouse in period τ; 0, otherwise
$\hat{\mathcal{X}}$	Set of triple-index x variables: $\hat{\mathcal{X}} = \{x_{ij\tau} \mid i \in I, j \in B, \tau \in [\![a_j, b_j]\!]\}$

Table C.3.: Additional notation for the time-indexed model.

$$\text{Minimize } \mathcal{F}(\hat{\mathcal{X}}, \mathcal{T}) = \sum_{i \in I} \sum_{j \in B_i} \sum_{\tau \in [\![a_j, b_j]\!]} r_i \cdot (d_i - \tau) \cdot x_{ij\tau} \tag{C.13}$$

subject to

$$\sum_{j \in B_i} \sum_{\tau \in [\![a_j, b_j]\!]} x_{ij\tau} = 1 \qquad \forall i \in I \tag{C.14}$$

$$\sum_{\substack{i \in I: \\ j \in B_i}} \sum_{\tau \in [\![a_j, b_j]\!]} w_i \cdot x_{ij\tau} \leq c_j \qquad \forall j \in B \tag{C.15}$$

$$t_j \leq d_i + (1 - \sum_{\tau \in [\![a_j, b_j]\!]} x_{ij\tau}) \cdot M \qquad \forall i \in I, j \in B_i \tag{C.16}$$

$$\sum_{\tau \in [\![a_j, b_j]\!]} \tau \cdot x_{ij\tau} \geq t_j - (1 - \sum_{\tau \in [\![a_j, b_j]\!]} x_{ij\tau}) \cdot M \qquad \forall i \in I, j \in B_i \tag{C.17}$$

$$\sum_{\tau \in [\![a_j, b_j]\!]} \tau \cdot x_{ij\tau} \leq t_j + (1 - \sum_{\tau \in [\![a_j, b_j]\!]} x_{ij\tau}) \cdot M \qquad \forall i \in I, j \in B_i \tag{C.18}$$

$$\sum_{i \in I} \sum_{j \in B_i} \sum_{\tau' = \max\{a_j; \tau\}}^{\min\{b_j; \tau + p_j - 1\}} q_i \cdot x_{ij\tau'} \leq Q \qquad \forall \tau \in T \tag{C.19}$$

$$\sum_{j \in B_i} \sum_{\tau'=\max\{a_j;\tau\}}^{\min\{b_j;\tau+p_j-1\}} x_{ij\tau'} \leq D \qquad \forall i \in I, \tau \in T \tag{C.20}$$

$$x_{ij\tau} \in \{0;1\} \qquad \forall i \in I, j \in B_i, \tau \in [\![a_j, b_j]\!] \tag{C.21}$$

Objective function (C.13) minimizes the total weighted earliness. Constraints (C.14) ensure that each item $i \in I$ is assigned exactly to one truck, which can depart only in its respective time window (i.e., no earlier than a_j and no later than b_j). Constraints (C.15) keep any truck $j \in B$ from being overloaded with respect to its available capacity c_j. Constraints (C.16) - (C.18) define the value of the variable t_j: Constraints (C.16) ensure that no item misses its deadline, i.e., if item i is loaded onto truck j, then departure time t_j of truck j cannot be later than d_i. Constraints (C.17) - (C.18) set the proper binary variable $x_{ij\tau}$ to value 1, i.e., whose third index τ equals the actual departure time t_j of truck j. Constraints (C.19) and (C.20) take care that in no period, the required workforce and dock doors exceed the available amounts Q and D, respectively. Finally, constraints (C.21) define the domain of binary variables.

C.3.4.3. Generalized set covering model

Both proposed models suffer from the presence of many difficult big M constraints, which weaken the LP relaxation and make the use of standard branch-and-bound and branch-and-cut methods as used by default solvers difficult. Moreover, the linear order model MILP-LO suffers from an exponential number of constraints, while the number of variables in the time-indexed model MILP-TI is only pseudo-polynomially bounded in $O(n \cdot m \cdot T)$, which can be too large a number to efficiently handle for practical problem dimensions. We, therefore, reformulate OTSLP as a generalized set covering (GSC) model, which serves as the foundation of a column generation approach, where the variables are added to the model iteratively.

Let $\mathcal{S}^j$ be the set of all partial schedules which are feasible for truck j, and $\mathcal{S} = \bigcup_{j \in B} \mathcal{S}^j$ be the set of all partial schedules. For notational convenience, let j_s be the index of the truck $j \in B$ which the partial schedule $s \in S^j$ is for. A partial schedule $s \in \mathcal{S}$ consists of a departure time $t_s \in \mathbb{Z}^+$, $a_{j_s} \leq t_s \leq b_{j_s}$, of truck j_s, and a set of items $I_s \subseteq I$ such that $j_s \in B_i$ and $d_i \geq t_s$, $\forall i \in I_s$, and $\sum_{i \in I_s} w_i \leq c_{j_s}$. Binary parameter α_{si} indicates if item i is in I_s (i.e., if item i is loaded onto truck j_s) in partial schedule $s \in \mathcal{S}$. Analogously, binary parameter τ_{st} indicates if truck j_s blocks resources during period $t \in T$ in partial

schedule $s \in \mathcal{S}$. The amount of resources required during this time in partial schedule s is encoded in parameter σ_s, calculated as $\sum_{i \in I_s} q_i$. The total weighted earliness e_s of partial schedule $s \in \mathcal{S}$ is calculated as $\sum_{i \in I_s} r_i \cdot (d_i - t_s)$. We define variable $\lambda_s \in \{0, 1\}$ to denote whether ($\lambda_s = 1$) or not ($\lambda_s = 0$) partial schedule $s \in \mathcal{S}$ is selected. The notation is summarized in Table C.4.

$\mathcal{S}^j$	Set of possible departure times and item assignments for truck j, index $s \in \mathcal{S}$
$\mathcal{S}$	Set of possible departure times and item assignments, $\mathcal{S} = \bigcup_{j \in B} \mathcal{S}^j$
j_s	The truck j such that $s \in \mathcal{S}^j$, i.e., the truck j that schedule s is for
α_{si}	Binary parameter: 1, if item i is loaded onto truck j_s in partial schedule $s \in \mathcal{S}$; 0, otherwise
τ_{st}	Binary parameter: 1, if truck j_s is being loaded during period t in partial schedule $s \in \mathcal{S}$; 0, otherwise
σ_s	Integer parameter: amount of resources required to load truck j_s in partial schedule $s \in \mathcal{S}$
e_s	Integer parameter: total weighted earliness of partial schedule $s \in \mathcal{S}$
λ_s	Binary variable: 1, if partial schedule $s \in \mathcal{S}$ is used; 0, otherwise
$\mathbf{\Lambda}$	Set of λ variables: $\mathbf{\Lambda} = \{\lambda_s \mid s \in \mathcal{S}\}$

Table C.4.: Notation for the generalized set covering model.

$$\text{Minimize } \mathcal{F}(\mathbf{\Lambda}) = \sum_{s \in \mathcal{S}} \lambda_s \cdot e_s \tag{C.22}$$

subject to

$$\sum_{s \in \mathcal{S}^j} \lambda_s \leq 1 \qquad \forall j \in B \tag{C.23}$$

$$\sum_{j \in B} \sum_{s \in \mathcal{S}^j} \alpha_{si} \cdot \lambda_s \geq 1 \qquad \forall i \in I \tag{C.24}$$

$$\sum_{j \in B} \sum_{s \in \mathcal{S}^j} \tau_{st} \cdot \sigma_s \cdot \lambda_s \leq Q \qquad \forall t \in T \tag{C.25}$$

$$\sum_{j \in B} \sum_{s \in \mathcal{S}^j} \tau_{st} \cdot \lambda_s \leq D \qquad \forall t \in T \tag{C.26}$$

$$\lambda_s \in \{0; 1\} \qquad \forall s \in \mathcal{S} \tag{C.27}$$

Objective function (C.22) minimizes the total weighted earliness over all trucks. Constraints (C.23) guarantee that for each truck $j \in B$, at most one schedule $s \in \mathcal{S}$ is selected. Constraints (C.24) ensure that each item is assigned to a truck. Note that

these are set covering constraints, i.e., they allow an item to be assigned more than once. However, assigning items multiple times cannot improve the objective value. Because of this and the generally faster convergence of set covering-based algorithms, we opt against set partitioning constraints (e.g., Feillet, 2010). Constraints (C.25) and (C.26) make it impossible to use more resources (logistics workers and dock doors) than are available in any period. Finally, constraints (C.27) define the domain of the variables.

C.4. Branch & Price

The number of variables in the GSC model grows exponentially with the input size. We, therefore, use a branch-and-price scheme to add them iteratively. At each node of the branch-and-bound tree, to obtain a lower bound, the linear relaxation of the GSC model is solved by column generation. This is done by iteratively solving the *restricted master problem* (RMP) and the *pricing problem*. The RMP is the LP relaxation of the GSC model using only a subset of the variables (or columns). The pricing problem consists of identifying variables with a negative reduced cost if any exist. These variables are then added to the RMP, and the process is repeated until no more negative reduced cost columns are found, at which point the RMP has been solved to optimality. Since the RMP is only an LP relaxation, the integrality of the variables may need to be enforced by branching.

First, in Section C.4.1 we describe how the initial columns for RMP are generated. The pricing problem and the branching scheme are described in Sections C.4.2 and C.4.3, respectively. Finally, in Section C.4.4, we describe some implementation details which we use to improve the performance of the branch-and-price algorithm for OTSLP.

C.4.1. Initial column

For proper functionality of the column generation algorithm, it has to be ensured that the RMP (i.e., the LP relaxation of the GSC model) can be solved to feasibility. Otherwise, if no feasible solution for the RMP exists, the optimal solution cannot be found and, hence, no proper dual pricing information can be passed to the pricing problem. By Proposition C.1, deciding feasibility of OTSLP is already NP-hard. Therefore, we generate a dummy column s_0, which assigns the complete set of items I to the first truck ($j_{s_0} = 1$) without taking into account truck compatibility and capacity. Further, we set the amount of

resources required to load the items to zero (i.e., $\sigma_{s_0} := 0$) and the total weighted earliness to a prohibitively large positive number M (i.e., $e_{s_0} := M$). We add s_0 to all nodes of the branch-and-bound tree to ensure that every RMP has a feasible solution. Due to the large penalty factor in the objective, the column is not used unless no other feasible solution exists.

C.4.2. Pricing problem

Given an optimal solution to the RMP for the current set of columns, let $\boldsymbol{u}^v$ be the vector of dual prices associated with Constraints v of the RMP ($v \in \{(\text{C.23}), (\text{C.24}), (\text{C.25}), (\text{C.26})\}$). The reduced cost $\tilde{c}_s$ of variable λ_s is

$$\tilde{c}_s = e_s + u_{j_s}^{(\text{C.23})} - \sum_{i \in I} \alpha_{si} \cdot u_i^{(\text{C.24})} + \sum_{t \in T} \tau_{st} \cdot \sigma_s \cdot u_t^{(\text{C.25})} + \sum_{t \in T} \tau_{st} \cdot u_t^{(\text{C.26})}.$$

To find a variable with negative reduced cost (or to prove that no such variable exists), we can solve the pricing problem for each truck $j \in B$ and possible departure time $\tilde{t}_j \in [\![a_j, b_j]\!]$ independently. For a given truck j and departure time $\tilde{t}_j$, we compute $\bar{u}^{(\text{C.25})}(j, \tilde{t}_j) = \sum_{t=\tilde{t}_j - p_j + 1}^{\tilde{t}_j} u_t^{(\text{C.25})}$ and $\bar{u}^{(\text{C.26})}(j, \tilde{t}_j) = \sum_{t=\tilde{t}_j - p_j + 1}^{\tilde{t}_j} u_t^{(\text{C.26})}$. This way, the pricing problem becomes

$$\text{Minimize } \mathcal{P}(\boldsymbol{\pi}) = u_j^{(\text{C.23})} + \bar{u}^{(\text{C.26})}(j, \tilde{t}_j) + \sum_{i \in \bar{I}} (\bar{e}(i, \tilde{t}_j) - u_i^{(\text{C.24})} + q_i \cdot \bar{u}^{(\text{C.25})}(j, \tilde{t}_j)) \cdot \pi_i$$

subject to

$$\sum_{i \in \bar{I}} w_i \cdot \pi_i \leq c_j \tag{C.28}$$

$$\pi_i \in \{0; 1\} \qquad \forall i \in \bar{I}, \tag{C.29}$$

where $\bar{I} \subseteq I$ denotes the set of items that can be assigned to truck j if it departs in period $\tilde{t}_j$, i.e., $\bar{I} = \{i \in I \mid d_i \geq \tilde{t}_j \wedge j \in B_i\}$; $\bar{e}(i, \tilde{t}_j)$ is the weighted earliness of item i if it is loaded onto truck j that departs in period $\tilde{t}_j$, i.e., $\bar{e}(i, \tilde{t}_j) = (d_i - \tilde{t}_j) \cdot r_i$; and π_i is a binary variable indicating if item i is loaded onto truck j.

The pricing problem hence corresponds to a series of $O(mT)$ knapsack problems, each of which can itself be solved in pseudo-polynomial time (Martello et al., 1999). If a feasible

solution of the pricing problem with a negative objective value is found, the variable λ_s with the truckload $I_s = \{i \in \bar{I} \mid \pi_i = 1\}$ and departure time $t_s = \tilde{t}_j$ is added to the RMP.

C.4.3. Branching

RMP is only an LP relaxation of GSC; therefore, an optimal solution obtained by column generation is not necessarily integral. To ensure the integrality of the final solution, we branch the nodes with fractional optimal solutions using the following branching scheme.

If an RMP solution is fractional, then at least one of the following two cases occur:

- there exists at least one item $i^* \in I$, which is (partially) loaded onto two (or more) distinct trucks;
- there exists at least one truck $j^* \in B$ which is (partially) assigned to two (or more) distinct departure times.

In other words, there exist at least two distinct partial schedules $s, s' \in \mathcal{S}$, $s \neq s'$, with $0 < \lambda_s, \lambda_{s'} < 1$, such that $\exists i^* \in I : i^* \in I_s \wedge i^* \in I_{s'}, j_s \neq j_{s'}$, and/or $\exists j^* \in B : t_s \neq t_{s'}, j_s = j_{s'} = j^*$, holds. That is, s and s' must assign an item to two distinct trucks and/or assign a truck to two distinct departure times. Our branching procedure generates two new nodes, none of which contains both columns s and s'. To do so, we use the following two-stage hierarchical branching scheme:

- *branching on items*: This branching rule is applicable if there exist at least one item $i^* \in I$ and two (or more) partial schedules $s, s' \in \mathcal{S}$ with $\lambda_s > 0$ and $\lambda_{s'} > 0$, which assign i^* to two (or more) distinct trucks $j, j' \in B : j \neq j'$ (i.e., $s \in \mathcal{S}^j, s' \in \mathcal{S}^{j'} : i^* \in I_s \wedge i^* \in I_{s'}$). Let $j^* \in B$ denote one of the trucks which item i^* is partially loaded on, chosen at random. We generate two nodes: in the first node, item i^* is forced to be loaded onto truck j^* ($x_{i^*j^*} = 1$) and in the second node, item i^* is forbidden from being loaded onto truck j^* ($x_{i^*j^*} = 0$). The new branching constraints can be easily integrated into the pricing problem: for the node with branching constraint $x_{i^*j^*} = 0$, we just remove i^* from $\bar{I}$ for all pricing problems and for the node with constraint $x_{i^*j^*} = 1$, we simply add the constant contribution (the weighted earliness of item $r_{i^*} \cdot (d_{i^*} - t_s)$) to the objective value and subtract w_{i^*} from c_{j^*} before solving the pricing problems for truck j^*. Moreover, the branching

constraints are considered in the RMP as follows: for the nodes with the branching constraint $x_{i^*j^*} = 1$, we remove all columns $s \in \mathcal{S}^{j^*}$ that do not assign item i^* to truck j^* (i.e., where $i^* \notin I_s$) and all columns s' that assign i^* to some truck j' other than j^* (i.e., where $i^* \in I_{s'} \wedge j' \neq j^*$). Similarly, the branching constraint $x_{i^*j^*} = 0$ can be considered by removing all columns s that assign item i^* to truck j^* (i.e., where $i^* \in I_s$).

- *branching on time windows*: This branching rule can be applied for the case when there exists at least one truck $j^* \in B$ and two (or more) partial schedules $s, s' \in \mathcal{S}$ with $\lambda_s > 0$ and $\lambda_{s'} > 0$, which assign truck j^* to different departure times $t_s \neq t_{s'}$, more specifically, $t_s < t_{s'}$, (i.e., $s, s' \in \mathcal{S}^j : t_s \neq t_{s'}$). We split the current possible time window of the truck's departure time $[a_{j^*}; b_{j^*}]$ into two sub-intervals, $[a_{j^*}; \lfloor t^* \rfloor]$ and $[\lceil t^* + \epsilon \rceil; b_{j*}]$ where ϵ denotes a sufficiently small positive number and t^* is computed as

$$t^* = \begin{cases} t_s, & \text{if } \frac{a_{j^*}+b_{j^*}}{2} < t_s, \\ \frac{a_{j^*}+b_{j^*}}{2}, & \text{if } t_s \leq \frac{a_{j^*}+b_{j^*}}{2} \leq t_{s'}, \\ t_{s'}, & \text{otherwise.} \end{cases}$$

  To consider these branching constraints in the RMP, we adjust the earliest and latest departure times of truck j^* accordingly (i.e., either $b_j := \lfloor t^* \rfloor$ or $a_{j*} := \lceil t^* + \epsilon \rceil$) and erase all columns violating these modified time windows. Further, we solve the series of pricing problems only with valid departure times $\tilde{t}_j$ that meet the modified time windows.

In preliminary tests, we experimented with various combinations of our branching rules (e.g., select the truck j^*, or item i^* with most fractional values and then, if possible, branch on item, otherwise on time window, or vice-versa, etc.). Among all observed combinations, the following branching scheme has shown the best performance, which we use for our computational experiments: from a set B^c of all candidate trucks which are assigned to at least two distinct departure times, select the truck j^* with the widest time window $j^* = \arg\max_{j \in B'}\{b_j - a_j\}$ and, if possible, branch on time window; if branching on time window is not applicable (because there is no truck which is assigned to more than one departure time to begin with), then branch on item, where the item is selected via the following rule. From a set I^c of all candidate items which are assigned to at least two distinct trucks, select the item i^* with the greatest earliness penalty factor, $i^* = \arg\max_{i \in I^c}\{r_i\}$. Further, to decide the next node to be branched, we use

the minimum lower bound rule, i.e., from all open nodes, we select the node with the smallest optimal objective value of the corresponding RMP.

C.4.4. Implementation details of the branch & price algorithm

In this section, we describe some implementation details, which in the preliminary tests have shown to improve the computational performance of our B&P algorithm.

For **solving the pricing problem**, recall that our pricing problem corresponds to the 0-1 knapsack problem. In each iteration of the column generation algorithm, we compute the input parameters for the corresponding knapsack problem (i.e., *profit* and *weight* of each item $i \in \bar{I}$, the constant contribution to the objective function and the current capacity c_j). Before solving the knapsack problem, we first reduce the problem size by erasing all redundant items from $\bar{I}$ that either have non-negative *profit*, or greater *weight* than the knapsack's capacity. Then, instead of solving the pricing problem to optimality, we first try to find a feasible solution with negative reduced cost by using a quick greedy heuristic that packs items into the knapsack in the order of their utility-to-weight ratio until no more items can be packed into the knapsack without violating the capacity constraint. The objective value of the obtained heuristic solution (including the constant term) represents an upper bound of the reduced cost of the corresponding column. By filling the knapsack with a fraction of the next-best item (by utility-to-weight), we solve the LP relaxation of the pricing problem and, thus, get a lower bound on the reduced cost. We only solve the pricing problem to optimality if the upper bound of the reduced cost is not negative, and the lower bound is. In this case, we solve the pricing problem to optimality in pseudo-polynomial time using the so-called "combo algorithm" proposed by Martello et al. (1999) (the codes are downloaded from `http://hjemmesider.diku.dk/~pisinger/codes.html`) and either find a new column with negative reduced cost or prove that no such column exists.

To accelerate the column generation algorithm for the root node, we add additional **initial columns to the root node** apart from the dummy column s_0. Specifically, for each truck $j \in B$, we generate a column s by assigning truck j to a feasible truckload (i.e., $I_s \subseteq \{i \in I \mid j \in B_i, d_i \geq a_j\}$ such that $\sum_{i \in I_s} w_i \leq c_j$), which is randomly selected, and the feasible departure time $t_s = \min\{\min_{i \in I_s}\{d_i\}; b_j\}$.

Apart from the dummy column s_0, each **child node receives all such columns** $s \in \mathcal{S}$ from the parent node whose optimal value in the parent node is positive and does not

violate the new branching constraint. We do not add such columns to the child nodes that are not selected in the optimal solution of the parent node (i.e., where the corresponding λ variable is zero). This may cause some columns to be generated multiple times but reduces the size of RMP significantly.

Moreover, we apply **logic testing before branching** each node and only branch the nodes with room for improvement, i.e., that satisfy the following criteria.

1. Let F^* denote the optimal objective value of the RMP of some node, which constitutes a lower bound on the objective value of the optimal integer solution. Since all objective parameters (i.e., total weighted earliness of every truck) are integer, the optimal objective value of OTSLP has to be integral. Hence, we only branch such nodes that satisfy $\lceil F^* \rceil < UB$, where UB denotes the objective value of the current best-known solution.

2. If all negative reduced-cost columns have been added, but the dummy variable s_0 still has a positive optimal value (i.e., $\lambda_{s_0} > 0$) in the RMP solution, no feasible solution exists for this node, and it can be fathomed.

Similarly, we apply **logic testing after branching** each node. Some branching constraints may implicitly cause a reduction of the search space of the child node which we take into account.

- Let $x_{i^*j^*} = 1$ be such a branching constraint that forces an item i^* to be loaded onto truck j^* such that $d_{i^*} < b_{j^*}$ holds. Note that all columns s that assign truck j^* to a departure time t_s later than deadline d_{i^*} of item i^* are infeasible. Therefore, after such a branching step, we implicitly restrict the time window of truck j^* by updating $b_{j^*} := d_{i^*}$.

- Analogously, for all nodes that are generated due to a branching constraint $[\lceil t^* + \epsilon \rceil; b_{j*}]$, we check all columns $s \in \mathcal{S}^{j^*}$ from the parent node if the assigned truckload I_s to truck j^* contains any item i with $d_i < \lceil t^* + \epsilon \rceil$. If the truckload I_s contains at least one item i whose deadline d_i is earlier than the new earliest possible departure time $\lceil t^* + \epsilon \rceil$ of truck j^*, then we do not add this column to the child node.

To quickly obtain a good **initial upper bound**, after adding all columns with negative reduced cost to the root node via column generation, we solve the corresponding root-node GSC with all added columns considering the integrality constraints. Although finding a feasible solution to GSC is strongly NP-complete (Garey and Johnson, 1979),

modern commercial solvers can solve even large GSC instances in acceptable computational times to optimality (e.g., Tadumadze et al., 2019, 2020b).

C.5. Computational study

In this section, we explore and compare the computational performance of the proposed solution approaches. Specifically, in Section C.5.2, we compare the computational performance of the mixed-integer linear programs (MILP-LO and MILP-TI) with the proposed B&P algorithm. To the best of our knowledge, there is no established testbed for OTSLP, so that we first describe how we generate new realistic test instances in Section C.5.1. Finally, in Section C.5.3, we explore some practical insights that can be helpful for dispatch warehouse managers in their decision making on a tactical planning level.

C.5.1. Problem instances and computational environment

Our instance generation scheme is geared to the aforementioned dispatch warehouse of a trailer coupling manufacturer that ships couplings to multiple destinations (e.g., different OEMs) just-in-time. We generate OTSLP instances with different sizes whereas the instance size is defined by the number of served OEMs O, which our instance generator receives as an input parameter.

OTSLP is an operational planning problem, which is solved once the definitive part requirements at the OEMs and the trucks' exact time windows (i.e., earliest and latest departure time for each truck) are known. The typical planning horizon for sequencing mixed-model assembly lines and for scheduling trucks is one day or even one shift (Boysen et al., 2009, Tadumadze et al., 2020a). Thus, it is advisable to plan OTSLP for a shorter planning horizon (e.g., one or even half a working shift). A time unit for our OTSLP instances corresponds to five minutes and, thus, a planning horizon consists of $T = 96$ periods, i.e., a standard 8 hours working shift.

Regarding the truck-related parameters, we assume each OEM $o \in \{1, \cdots, O\}$ to be exclusively served by a truck fleet Φ_o of fixed size consisting of trucks of different types, bound for one particular OEM o. Specifically, we consider four different truck types (dubbed S, M, L, and XL). The truck types differ in the trailer capacity (corresponding

to the maximal number of pallets they can load), the docking time, and the arrival frequency at the terminal during the planning horizon (i.e., count of trucks of that type in each fleet Φ_o). The detailed characteristics of the four truck types are displayed in Table C.5. To generate realistic truck capacities c_j, we rely on the maximal number of pallets for each particular trailer, based on the realistic data provided by Goodloading (2019). We assume that the docking time p_j of each truck j depends on the trailer size, i.e., the maximal number of pallets that can fit in the trailer. To generate realistic docking times, we first set the docking time (including the pre- and post-processing times before and after the loading process) for the standard semitrailer (XL) based on the empirical data provided by Burdzik et al. (2014). Then, we re-scale the processing times for smaller truck types accordingly. The earliest possible departure time a_j of each truck $j \in B$ is calculated as $a_j = \eta_j + p_j$, where η_j denotes the arrival time of truck j at the dispatch warehouse. We assume that at the beginning of the planning horizon, from each fleet Φ_o, there is one truck of each type waiting in the terminal yard. This way, we ensure that there are sufficient trucks to meet early deadlines of items. Further, we assume that the inter-arrival time $\eta'_j - \eta_j$ of each pair j and j' of successive trucks of the same type depends on the truck type: the smaller the truck, the shorter the inter-arrival time (i.e., the higher the frequency). As a result, each fleet Φ_o consists of more trucks with a small trailer capacity than the ones with larger trailers. Trucks' inter-arrival time for each truck size is determined by dividing T by the number of that particular trucks (see Table C.5). In total, each OEM $o \in \{1, \cdots, O\}$ is served by identical truck fleets Φ_o, consisting of 15 trucks. Thus, the total number of trucks of an OTSLP instance can be derived as $n = 15 \cdot O$. Finally, the latest possible departure time b_j of each truck j is computed as $b_j = min\{T; a_j + \omega_j\}$ where ω_j denotes the time window of truck j (i.e., maximal duration of its stay at the terminal including the docking time). Unless noted otherwise, the time window ω_j for each truck $j \in B$ is randomly drawn as $\omega_j = rnd(24, 36)$. Note that $\omega_j = rnd(24, 36)$ means that trucks stay at the terminal for between 2 and 3 hours.

Regarding the item related parameters, we recall that in the OTSLP, an item corresponds to an entire shipment or order of parts packed on one or multiple pallets. We assume that each OEM $o \in \{1, \cdots, O\}$ orders a fixed number of 50 items in the planning horizon. Thus, the total number of items m for an OTSLP instance can be derived as $m = 50 \cdot O$. The relative importance of item r_i (i.e., penalty cost in case of early arrival) is randomly

Truck type	Size	Trailer capacity [pallets]	Docking time [TU]	# of trucks per fleet
Solo truck 6T 720x245x240	S	18	7	8
Solo truck 8T 820x245x240	M	20	8	4
Truck with 40ft sea container 1203x235x240	L	24	9	2
Standard semitrailer 1360x245x270	XL	33	11	1

Table C.5.: Truck characteristics per truck type

drawn as $r_i = rnd^{\mathbb{Z}}(1,5)$, where $rnd^{\mathbb{Z}}$ denotes a discrete uniformly distributed random integer number from the interval given in brackets. Similarly, the space w_i required for each item $i \in I$ is also randomly drawn as $w_i = rnd^{\mathbb{Z}}(1,5)$, where a value of 1 can be thought of as an entire pallet. The number of logistics workers required to (prepare and) load order i onto a truck is generated as $q_i = \lfloor w_i \cdot rnd^{\mathbb{R}}[0,1] \rceil$, where $rnd^{\mathbb{R}}$ denotes a uniformly distributed random number from the interval in the argument and $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. This way, the number of workers q_i, on the one hand, depends on the order size w_i and, on the other hand, is randomly re-weighted to account for some other order-specific factors (e.g., shape, weight, packaging, and labeling effort of the order, etc.). The deadline d_i of item i is randomly drawn from the entire planning horizon starting from the earliest possible departure time of any truck, i.e., $d_i = rnd^{\mathbb{Z}}(\min\{a_j\}, 96)$. Finally, the set of compatible trucks B_i for each order i, which has to be shipped to OEM o, consists only of the trucks from the corresponding truck fleet Φ_o.

We assume that the number of dock doors D depends on the size of the warehouse (i.e., the number of served OEMs O) and is set to $D = 3 \cdot O$. The number of workers is computed as $Q = \left\lfloor \alpha \cdot \frac{\sum_{i \in I} q_i}{n} \cdot D \right\rceil$, with $\alpha \in \mathbb{R}^+$ as a predefined parameter determining the desired level of available workforce. Unless noted otherwise, parameter α is set to value 1, leading the available workforce size Q to be calculated as the product of the expected amount $\frac{\sum_{i \in I} q_i}{n}$ of required workforce per truck and the maximal possible number D of trucks that can be docked simultaneously (i.e., the number of dock doors). In Section C.5.3, we observe the effect of the workforce size on the solution quality by comparing solutions of different OTSLP instances by varying the value of parameter α.

We implement our solution procedures in C# 7.0 and apply the commercial solver IBM ILOG CPLEX Optimizer V12.9.0 for solving all (MI)LPs (excluding the ILP for the pricing problem, described in Section C.4.2, which we solve with specialized solution methods as described in Section C.4.4). All tests are executed on an x64 PC with an Intel Core i7-8700K 3.70 GHz CPU and 64 GB RAM.

C.5.2. Computational performance

In this section, we compare the computational performance of the proposed models and solution approaches (MILP-LO, MILP-TI, and B&P) solving OTSLP instances. For this comparison, we generate new OTSLP instances for differently sized dispatch warehouses,

as described in Section C.5.1. In particular, the value for the input parameter O (the number of served OEMs) is varied in the range $O \in \{1, 2, 3, 4\}$, generating OTSLP instances with $n = \{15, 30, 45, 60\}$ trucks and $m = \{50, 100, 150, 200\}$ items, respectively. For each instance size, we generate 10 random OTSLP instances so that for this comparison, in total, 40 new OTSLP instances are available. The time limit for solving the MILP models, as well as the B&P approach, is set to 1800 CPU seconds unless noted otherwise. Moreover, to make a fair comparison between alternative solution approaches, we execute all solution methods (including the default solver) on a single thread.

Tables C.6 and C.7 summarize the computational results. The first column "Name" indicates the instance name consisting of the number of items m, the number of trucks n, and a running index. For each approach, we report the computational runtime (in CPU seconds) (column "Time"), the objective value of the best found solution (column "UB"), the best lower bound (column "LB"), and the relative gap between upper and lower bound (in %), calculated as $gap = \frac{UB-LB}{LB} \cdot 100$ (column "Gap"). Further, since both MILPs find relatively weak lower bounds, we additionally report the relative gap of the best-found solution of each MILP UB^{MILP} to the best found objective value of the B&P approach UB^{BP} calculated as $gap^{BP} = \frac{UB^{MILP}-UB^{BP}}{UB^{BP}} \cdot 100$ (column "Gap^{BP}"). Due to the presence of an exponential number of constraints, CPLEX runs out of memory even before generating the corresponding MILP-LO models for instances with more than $m = 15$ trucks. Thus, we only report the computational results of MILP-LO for the OTSLP instances of the smallest size with $m = 15$ trucks and $n = 50$ items.

	B&P				MILP-TI					MILP-LO				
Name	Time	UB	LB	Gap	Time	UB	LB	Gap	Gap^{BP}	Time	UB	LB	Gap	Gap^{BP}
$O = 1$														
50x15-01	71.6	167	167	0.0	1828.1	174	94.4	84.4	4.2	1871.8	171	54.1	215.9	2.4
50x15-02	9.2	124	124	0.0	185.8	124	124.0	0.0	0.0	1880.4	124	68.4	81.3	0.0
50x15-03	3.1	139	139	0.0	1829.8	148	80.2	84.5	6.5	1882.1	148	50.3	194.1	6.5
50x15-04	1800.1	208	204	2.0	1830.3	217	54.2	300.6	4.3	1849.5	208	21.8	856.1	0.0
50x15-05	9.0	133	133	0.0	66.9	133	133.0	0.0	0.0	1829.4	133	105.6	26.0	0.0
50x15-06	8.0	171	171	0.0	1830.4	175	108.2	61.7	2.3	1831.4	171	79.6	115.0	0.0
50x15-07	38.7	158	158	0.0	1830.0	158	107.5	47.0	0.0	1831.9	158	59.9	163.6	0.0
50x15-08	3.2	185	185	0.0	1853.0	199	77.0	158.3	7.6	1834.1	190	70.7	168.7	2.7
50x15-09	59.5	146	146	0.0	1829.8	146	89.3	63.5	0.0	1829.4	152	57.8	163.2	4.1
50x15-10	36.3	154	154	0.0	1829.3	154	141.6	8.8	0.0	1828.9	154	111.0	38.7	0.0
Mean	*203.9*			*0.2*	*1491.3*			*80.9*	*2.5*	*1846.9*			*202.3*	*1.6*
$O = 2$	0.0													
100x30-01	96.4	315	315	0.0	1830.6	332	169.7	95.6	5.4	-	-	-	-	-
100x30-02	13.7	305	305	0.0	1838.1	339	142.2	138.4	11.2	-	-	-	-	-
100x30-03	130.4	297	297	0.0	1833.7	327	150.8	116.8	10.1	-	-	-	-	-
100x30-04	23.6	341	341	0.0	1818.5	374	195.2	91.7	9.7	-	-	-	-	-
100x30-05	66.8	337	337	0.0	1839.1	375	154.4	142.9	11.3	-	-	-	-	-
100x30-06	9.7	307	307	0.0	1826.3	337	143.2	135.3	9.8	-	-	-	-	-
100x30-07	153.4	284	284	0.0	1849.9	313	130.7	139.4	10.2	-	-	-	-	-
100x30-08	13.2	358	358	0.0	1838.8	402	131.8	205.0	12.3	-	-	-	-	-
100x30-09	637.4	314	314	0.0	1838.1	347	166.3	108.6	10.5	-	-	-	-	-
100x30-10	9.8	328	328	0.0	1840.2	359	147.8	142.9	9.5	-	-	-	-	-
Mean	*115.4*			*0.0*	*1835.3*			*131.7*	*10.0*	-			-	-

Table C.6.: Comparison of B&P with MILP-LO and MILP-TI (part 1)

	B&P				MILP-TI					MILP-LO				
Name	Time	UB	LB	Gap	Time	UB	LB	Gap	Gap^{BP}	Time	UB	LB	Gap	Gap^{BP}
$O = 3$														
150x45-01	42.1	429	429	0.0	1907.3	499	228.0	118.8	16.3	-	-	-	-	-
150x45-02	115.1	446	446	0.0	1920.6	464	176.9	162.3	4.0	-	-	-	-	-
150x45-03	102.0	439	439	0.0	1893.7	479	204.0	134.8	9.1	-	-	-	-	-
150x45-04	140.0	430	430	0.0	1895.9	475	198.4	139.4	10.5	-	-	-	-	-
150x45-05	26.1	428	428	0.0	1861.5	481	170.0	183.0	12.4	-	-	-	-	-
150x45-06	120.0	479	479	0.0	1884.9	566	185.5	205.1	18.2	-	-	-	-	-
150x45-07	28.7	487	487	0.0	1904.8	620	222.6	178.5	27.3	-	-	-	-	-
150x45-08	74.4	409	409	0.0	1894.7	423	220.8	91.5	3.4	-	-	-	-	-
150x45-09	44.4	437	437	0.0	1885.6	491	242.2	102.7	12.4	-	-	-	-	-
150x45-10	31.1	448	448	0.0	1930.1	577	167.7	244.1	28.8	-	-	-	-	-
Mean	*72.4*			*0.0*	*1897.9*			*156.0*	*14.2*	-			-	-
$O = 4$														
200x60-01	120.8	655	655	0.0	1895.0	826	251.1	229.0	26.1	-	-	-	-	-
200x60-02	1759.8	691	691	0.0	1872.5	778	306.2	154.1	12.6	-	-	-	-	-
200x60-03	71.6	602	602	0.0	1830.6	736	273.3	169.3	22.3	-	-	-	-	-
200x60-04	943.1	564	564	0.0	1858.1	652	248.0	163.0	15.6	-	-	-	-	-
200x60-05	103.7	596	596	0.0	1863.2	688	223.9	207.3	15.4	-	-	-	-	-
200x60-06	158.5	630	630	0.0	1856.6	693	278.7	148.7	10.0	-	-	-	-	-
200x60-07	1573.8	586	586	0.0	1896.2	754	260.9	189.0	28.7	-	-	-	-	-
200x60-08	155.4	595	595	0.0	1877.0	662	289.7	128.5	11.3	-	-	-	-	-
200x60-09	858.8	596	596	0.0	1830.8	679	240.3	182.6	13.9	-	-	-	-	-
200x60-10	1005.9	612	612	0.0	1872.2	709	284.1	149.6	15.9	-	-	-	-	-
Mean	*675.1*			*0.0*	*1865.2*			*172.1*	*17.2*	-			-	-

Table C.7.: Comparison of B&P with MILP-LO and MILP-TI (part 2)

The first remarkable result is that B&P outperforms both MILP models, solving in total 39 out of 40 OTSLP instances to optimality within the given time limit. The default solver, on the other hand, struggles with solving instances to proven optimality within the time limit using either proposed model—MILP-TI or MILP-LO. Using the time-indexed model, CPLEX solves only 2 out of 40 instances to proven optimality within a time limit. In the other 38 cases, CPLEX finds a feasible solution, but the gap to the B&P objective value is the more remarkable the larger the OTSLP instance is. The effort of the commercial solver is even vaster when the linear ordering model MILP-LO is applied. Suffering from the presence of an exponentially growing number of constraints, the MILP-LO model instances can be generated only for the ten smallest instances (with $m = 15$), none of which are solved to optimality. Hence, we can conclude that neither MILP model scales well, whereas B&P performs reasonably well even on the largest instances in the test set.

To get a better idea of how the novel B&P approach performs, in Table C.8, we list the computational results in more detail. In particular, apart from the total computational time (presented in column "$Time^{BP}$"), we also report the number of nodes in the branch-and-bound tree, as well as the total number of generated columns (columns "$Nodes$" and "$Columns^{BP}$", respectively). Recall that some columns may be generated multiple times as we do not add every column of parent nodes to their child nodes. Moreover, we report the computational details of the root node of the branch-and-bound tree. Specifically, the computational time and the number of generated columns after solving the root node with column generation are listed in the columns "$Time^{root}$" and "$Columns^{root}$", respectively. Moreover, column "$Time^{firstfeasible}$" contains the total computational time required to find the first feasible solution by solving the RMP model from the root node with integrality constraints. This consists of the computational time to solve the root node via column generation $Time^{root}$ and the computational time to solve the corresponding GSC model.

It is notable that for all 40 observed instances, our procedure manages to obtain a feasible solution in a reasonable time. Even for the largest OTSLP instances, the first feasible solution is found in about 1 minute. The relatively small number of nodes in the branch-and-bound tree (see column "*nodes*") suggests that the quality of the root node solution is already quite good. For 8 out of 40 instances, the root node is directly pruned (without branching), which indicates that the first-found feasible solution is already proven optimal. Moreover, for many of the remaining 32 cases, the first feasible solution is already optimal, and the remainder of the runtime is spent proving optimality.

Table C.8.: Detailed computational results of B&P algorithm

	Total			Root node		
Instance	$Time^{BP}$	Nodes	$Columns^{BP}$	$Time^{root}$	$Columns^{root}$	$Time^{firstfeasible}$
$O = 1$						
50x15-01	71.6	467	525701	1.5	4609	3.1
50x15-02	9.2	25	44707	1.6	4837	2.9
50x15-03	3.1	1	5152	1.8	5152	3.1
50x15-04	1800.1	15263	14355765	1.5	4462	2.6
50x15-05	9.0	51	55271	1.1	3920	2.0
50x15-06	8.0	27	41469	1.8	4975	3.1
50x15-07	38.7	89	185520	1.8	4985	3.4
50x15-08	3.2	1	4946	1.9	4946	3.2
50x15-09	59.5	371	433771	1.5	4806	2.8
50x15-10	36.3	235	266483	1.4	4718	2.7
Mean	*203.9*	*1653*	*1591878.5*	*1.6*	*4741*	*2.9*
$O = 2$						
100x30-01	96.4	169	372043	5.5	8623	10.0
100x30-02	13.7	1	10549	7.6	10549	13.7
100x30-03	130.4	85	333837	7.2	10333	13.2
100x30-04	23.6	21	62981	5.4	8571	9.7
100x30-05	66.8	49	175512	5.8	8832	10.5
100x30-06	9.7	1	8712	5.5	8712	9.7
100x30-07	153.4	205	539715	5.6	8864	10.3
100x30-08	13.2	1	10383	7.3	10383	13.2
100x30-09	637.4	669	1992607	6.5	9495	11.8
100x30-10	9.8	1	8690	5.3	8690	9.8
Mean	*115.4*	*120.2*	*351502.9*	*6.2*	*9305.2*	*11.2*
$O = 3$						
150x45-01	42.1	7	43290	13.6	13465	25.5
150x45-02	115.1	45	192390	14.3	13844	26.9
150x45-03	102.0	37	162409	17.3	15301	32.7
150x45-04	140.0	29	194438	15.5	14443	29.2
150x45-05	26.1	1	13559	14.2	13559	26.1
150x45-06	120.0	27	169051	15.2	14271	28.6
150x45-07	28.7	1	14296	15.4	14296	28.7
150x45-08	74.4	17	91672	17.2	15054	31.9
150x45-09	44.4	5	38819	15.5	14366	29.0
150x45-10	31.1	1	15042	16.6	15042	31.1
Mean	*72.4*	*17*	*93496.6*	*15.5*	*14364.1*	*29.0*
$O = 4$						
200x60-01	120.8	7	79149	33.3	20275	64.0
200x60-02	1759.8	339	2261776	27.6	18146	52.6
200x60-03	71.6	3	34783	29.8	18810	57.6
200x60-04	943.1	129	1034901	29.4	18469	56.0
200x60-05	103.7	7	66673	32.4	19642	61.9
200x60-06	158.5	15	134791	29.6	18912	57.0
200x60-07	1573.8	207	1706956	28.2	18224	53.7
200x60-08	155.4	15	129153	27.1	17751	51.8
200x60-09	858.8	117	934842	29.9	18803	57.0
200x60-10	1005.9	115	1018125	33.5	20284	65.1
Mean	*675.1*	*95.4*	*740114.9*	*30.1*	*18931.6*	*57.7*

This indicates that truncated versions of our B&P may also serve as good heuristics.

C.5.3. Managerial insights

In this section, we explore some practical insights, such as the impact of the truck time window management policy as well as workforce size on the delivery punctuality. The results can be helpful from a managerial viewpoint at the tactical planning level when deciding on the time window management policy or the workforce size to achieve the desired service level.

So far, for all generated OTSLP instances used in the previous section, the time window ω_j for each truck $j \in B$ was randomly drawn between 2 and 3 hours. However, by changing the time window management policy for the trucks, the terminal managers can affect the delivery punctuality: tighter time windows make the planning process easier for the logistics providers and probably increase truck utilization, but come at the cost of decreasing flexibility for the dispatch warehouse, which has fewer truck departure times to choose from, possibly making deliveries less just-in-time. To explore the impact of the time window management on the solution quality (i.e., delivery punctuality), we generate additional OTSLP instances by varying the width of the trucks' time windows. We solve them using the B&P algorithm and compare the objective value (i.e., total weighted earliness) of the instances with different time window widths of the trucks.

As a benchmark set of instances, we use 10 OTSLP instances with $n = 45$ trucks and $m = 150$ items from the previous study (i.e., instances: 150x45-01-150x45-10). Each benchmark instance is modified by resetting the truck time window for three scenarios: $\omega_j = rnd^{\mathbb{Z}}(0, 12)$, $\omega_j = rnd^{\mathbb{Z}}(12, 24)$ and $\omega_j = rnd^{\mathbb{Z}}(36, 48)$, leading to an additional 30 instances with maximal docking duration at the terminal between 0 and 1 hour, 1 and 2 hours and 3 and 4 hours, respectively. Note that, apart from the trucks' time windows (or, more precisely, the trucks' latest possible departure dates b_j) the modified OTSLP instances are identical to the original benchmark instances. Note that for the tightest scenario (i.e., $\omega_j = rnd^{\mathbb{Z}}(0, 12)$), if a randomly drawn time window ω_j of some truck j is shorter than its docking time p_j, we set the time window to p_j. This means that truck j cannot wait at the terminal and, hence, either has to be directly docked at the door or cannot be loaded at all.

Figure C.3 displays the aggregated results of our study (including the original 2-3 hour scenario). Each point in the figure represents the average results of the 10 OTSLP

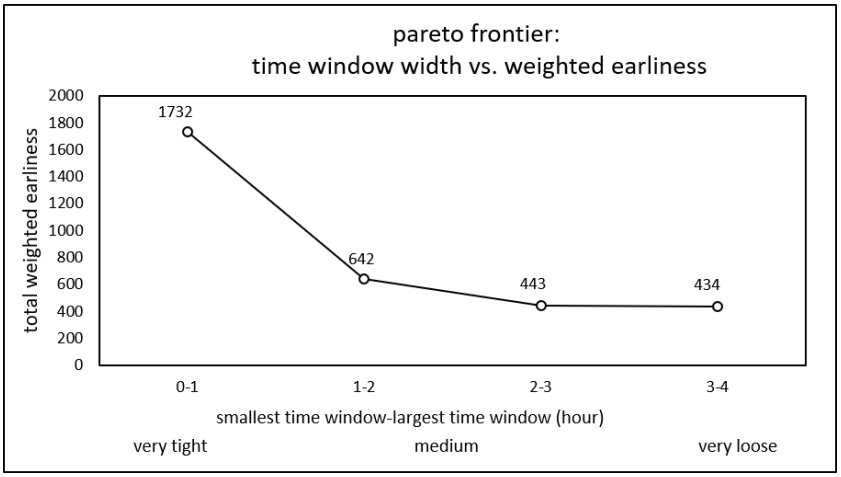


Figure C.3.: Impact of time window management on the delivery punctuality

instances of the same scenario. We would expect a strict time window management (i.e., tighter time windows) to make it hard to punctually deliver orders to the OEMs. And indeed, this is what the results indicate. The decreasing curve of the chart indicates an improvement in delivery punctuality (i.e., a decrease of total weighted earliness) when the length of the trucks' time windows grows. This is only true up to a point, however. The tightening of the time window management policy from 2-3 hours to 1-2 (or even 0-1) hours leads to an increase of the total weighted earliness by about 43% (or even by 292%). On the other hand, loosening the trucks' time windows to 3-4 hours only leads to improvements in delivery punctuality quality by 2%, which is probably not worth the additional planning headache for logistics providers, who have to have their trucks on standby for up to four extra hours.

Analogously, we observe the impact of the workforce size (i.e., number of logistics workers Q) on the delivery punctuality by modifying the benchmark instances once more. This time, we change the available workforce size Q by varying the parameter α in the range $\alpha \in \{0.6, 0.7, 0.8, 0.9\}$ (leading to an additional 40 OTSLP instances). The value $\alpha = 0.6$ can be interpreted such that in peak hours, i.e., when all dock doors might be busy with trucks, the amount of available logistics workers Q is about equal to the 60% of expected required workers. As a result, the shortage of logistics workers may lead to delays (or even canceling) of some trucks' processing, which can negatively affect the solution quality. The aggregated results (averaged per workforce scenario) are depicted in Figure C.4. Note that for this study, not all problem instances can be solved to proven optimality. Therefore, to get a better idea of the relation, we display the best upper

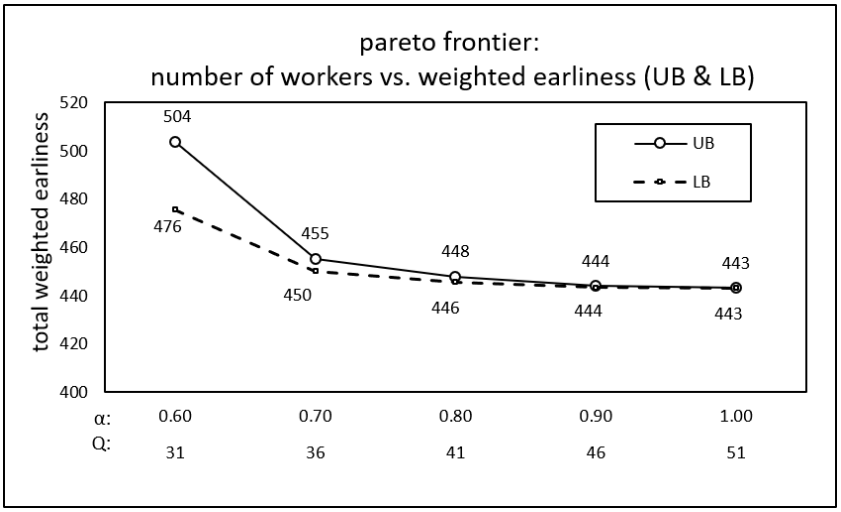


Figure C.4.: Impact of workforce size on the delivery punctuality

and lower bounds within the time limit. As one would expect, the average weighted total earliness goes down with increasing workforce size. By far the greatest gains are made when going from $\alpha = 0.6$ to $\alpha = 0.7$, while going from $\alpha = 0.9$ to $\alpha = 1.0$ makes little difference. This implies that terminal managers should take care not to run their dispatch warehouse with a bare skeleton crew: while it may still be feasible to ship all orders, the punctuality will suffer. On the other hand, there is little point in padding the workforce beyond a certain level. Note that for scenarios with an even tighter workforce ($\alpha = 0.5$), many instances become inherently infeasible.

C.6. Conclusion

In this paper, we introduce the novel outbound truck loading and scheduling problem (OTSLP), which deals with two issues: assignment of orders to outbound trucks and scheduling the loading of the trucks at the dock doors such that the orders arrive at the customers as late as possible but no later than their deadline, considering the trucks' capacity and time window restrictions as well as the available resources at the dispatch warehouses (i.e., dock doors and logistics workers). We formalize the problem and provide three alternative mixed-integer linear programming models. Furthermore, we develop an exact branch-and-price algorithm (B&P). We compare the novel B&P algorithm with a commercial solver using both mixed-integer linear programming models. B&P performs quite well, solving most of the large OTSLP instances to optimality

within the given time limit of 30 minutes.

For terminal managers, we derive some practical insights:

- Offering tight processing time windows to logistics providers, while making their own planning easier, costs a lot of flexibility, which is reflected in decreased punctuality of the shipments. On the other end of the spectrum, very large time windows offer only little benefit in terms of punctuality but probably decrease truck utilization significantly.
- Employing only a bare-minimum workforce in the dispatch area is not a good idea as this can severely delay just-in-time shipments. However, padding the workforce beyond a certain point is not helpful, either.

Future research should focus on integrating detailed workforce scheduling into the OTSLP, e.g., considering individual workers switching trucks once their order is loaded and considering their walking times between two successive tasks. Furthermore, integration of the vehicle routing problem into the OTSLP may be a rewarding research task, especially for those cases where the dispatch warehouse is relatively far from the OEMs and/or multiple suppliers are visited on milk-runs.

Chapter D.

Yard truck scheduling*

Abstract: Spotters (also denoted as switchers) are specialized truck tractors, which are dedicated to the rapid maneuvering of semitrailers between parking lot and dock doors in large trailer yards. This chapter is dedicated to spotter scheduling, i.e., the assignment of predefined trailer movements to a given fleet of spotters. The limited number of dock doors for loading and unloading is often the scarce resource during trailer processing, so that idle time of the bottleneck, e.g., caused by unforeseen delay in the yard, is to be avoided. In this setting, we aim to insert time buffers between any pair of subsequent jobs assigned to the same spotter, so that small delays are not propagated and subsequent jobs can still be executed in a timely manner. We formalize two versions of the resulting robust spotter scheduling problem and provide efficient algorithms for finding optimal solutions in polynomial time. Furthermore, we simulate delays during the execution of spotter schedules and show that the right robustness objective can greatly improve yard performance.

*This chapter has been published as: Tadumadze, G., Boysen, N., Emde, S. (2020): Robust spotter scheduling in trailer yards. *OR Spectrum* 42(4), 995–1021

D.1. Introduction

Increasing freight traffic in many regions of the world (e.g., in Europe, see Statista, 2018) does not only burden the edges (streets) of road networks but also many nodes. Examples of huge terminals where up to several hundred trucks are to be loaded and unloaded each day are cross docks (Ladier and Alpan, 2016), automobile plants (Battini et al., 2013), distribution centers of food retailers (Bodnar et al., 2017), freight airports (Ou et al., 2010), and hub terminals in the postal service industry (Boysen et al., 2017). In many of these nodes, especially during peak hours, considerable waiting times for an (un-)loading of trucks occur. The large German transport cooperative Elvis with 10,643 trucks, for instance, reports that their average daily waiting times of 3.5 h per truck accumulate to yearly waiting costs of about €400 million (VRS, 2012). In an empirical study, 45.9% of the surveyed German truck drivers quantify their average waiting time per stop to exceed 1 h (VRS, 2011). Existing ideas on how to resolve this problem mainly address the demand side. Novel software solutions propagate an internet-based booking of time windows, so that truck arrivals can be spread out over the day (Descartes, 2019), and the current scientific research aims at a coordination via auction mechanisms (Karaenke et al., 2019). We, however, address the internal processes and show that robust schedules for the movement of semitrailers within yards also have the potential to considerably reduce waiting times.

D.1.1. Trailer yard operations and literature review

An important decision to make in the aforementioned nodes is to either let the trucks transporting semitrailers to and from the terminal directly approach the dock doors or to apply dedicated spotters for the intra-terminal trailer movement. Spotters (also denoted as switchers or terminal tractors, see Yano et al., 1998, Berghman and Leus, 2015) are specialized truck tractors which are exclusively applied for the rapid maneuvering of semitrailers between their parking positions and the dock doors where trailers are loaded and unloaded. A spotter is depicted in Figure D.1(a).

Applying extra spotters allows accelerated maneuvering and earlier release of the trucks. Spotters are more agile than conventional trucks, specifically designed for a rapid (un-) coupling process of semitrailers, and due to learning effects, their drivers are much better trained in backing into a dock. A terminal manager of a large German trailer yard we visited told us that spotters cut transportation time in the yard at least in half.

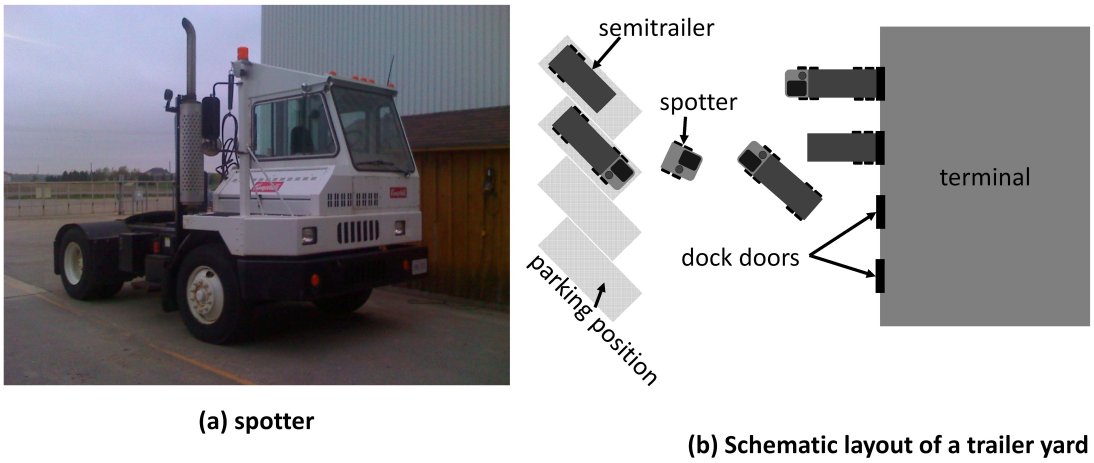


Figure D.1.: Spotter[†]

Moreover, spotters allow for a decoupling of trailer processing at the dock doors and trailer transportation (beyond the yard). Inbound trucks can leave their trailer on some open parking space in the trailer yard and can directly head toward their successive appointment. They need not wait for the processing time window of their trailer, which may considerably improve truck utilization. On the outbound side, dock doors are not blocked by already loaded trailers waiting for a delayed truck to pick them up. On the downside, spotters incur additional investment and operational costs.

When applying spotters for trailer movement, the yard processes are organized as follows. On the inbound side, an incoming truck reaching a trailer yard is registered at the terminal gate and appointed an open parking space in the parking lot of the yard. Figure D.1b depicts the schematic layout of a trailer yard. Once the trailer is positioned and uncoupled, the truck can directly leave the terminal. The arrival of the loaded trailer is registered in the yard's information system, e.g., by a yard operator controlling the parking lot with video surveillance. Then, the trailer is assigned a dock door and a processing time window. In the literature, the decision on the *where* and *when* of (un-)loading a given set of trailers is denoted as truck scheduling. A survey on the numerous solution procedures for this decision task is provided by Boysen and Fliedner (2010). Once such a schedule is available, loaded trailers need to be delivered in a timely manner to their dedicated docks by spotters. The spotter drivers are informed either via radio communication by the yard operator or via a display mounted on the

[†]Spotter (Figure D.1a is published under the Creative Commons Attribution Share Alike 3.0 Unported License. The author of the picture is Exit2DOS2000.

dashboard. Once docked, trailers are unloaded, e.g., by forklifts or directly onto a telescopic conveyor, while the spotter heads to the next trailer on its list. When a trailer is completely unloaded, the status is transferred to the information system and a new transport request is generated as well as assigned to a spotter, which is to return the empty trailer to a parking space. On the outbound side, these steps are analogously executed in reverse order.

This paper is dedicated to spotter scheduling. A given set of trailer movements either from the parking lot to some dock door (i.e., loaded inbound trailers or empty outbound trailers) or vice versa (i.e., empty inbound trailers or loaded outbound trailers) are to be assigned to a given fleet of spotters. Truck scheduling problems from the literature almost without exceptions typically ignore the workforce scheduling aspect. However, Tadumadze et al. (2019) propose integrated truck and workforce scheduling problem but their workforce is dedicated to the intra-terminal operations (i.e., unloading goods from trailers at dock door) while spotter scheduling treats operations in the yard, i.e., outside of the terminal. The only other papers treating spotter scheduling are the ones by Berghman et al. (2014) and Berghman and Leus (2015). Both papers integrate truck scheduling, i.e., the decision on the where and when of truck processing, with spotter scheduling and model the complete process as a three-stage flexible flow shop problem. On the first stage, spotters deliver trailers to docks, where they are processed (stage two) and finally delivered back to the parking lot (stage three). A peculiarity is that stages one and three share resources, i.e., the spotter fleet. Mixed integer models for the resulting problem are provided in Berghman et al. (2014) and heuristic solutions for a real-world case at a Toyota warehouse in Belgium are presented in Berghman and Leus (2015).

Clearly, truck and spotter scheduling are heavily interdependent, so that integrating both planning tasks into a holistic problem seems a good choice for many settings. However, the resulting problem is quite bulky and difficult to solve. Therefore, it may be convenient to decompose it if a single stage of the three-stage process is the main bottleneck. This is, for instance, the case at the aforementioned yard of a large German postal service provider we visited. Over the years especially the increasing volumes of e-commerce have led to a much higher workload to be processed at the terminal. While the spotter fleet can conveniently be adapted to changing capacity situations, the terminal cannot be expanded that easily, so that (especially during peak hours) the dock doors are the unique bottleneck resource. In such a setting, truck schedules can be derived independently of the spotters, such that the docks are efficiently utilized. Once such a

truck schedule is given, the aim of the spotter schedule reduces to realizing the given trailer transports in a timely manner such that the docks never run idle.

Instead of tightly coupling trailer processing and spotter scheduling, which may lead to a propagation of delays whenever unforeseen events occur during plan execution, the focus should rather be on robustness. Although the spotter fleet is sufficiently large per se, from time to time unforeseen delays occur in our terminal. For instance, unloading a trailer may take a little bit more time than expected or some trucks may arrive late at the terminal. These unforeseen events that regularly occur in any terminal lead to an idle time of the actual bottleneck resource, i.e., the dock doors. Therefore, we aim to determine, for the given truck schedules, spotter schedules that are robust in the face of unforeseen disturbances.

Although robust optimization is a wide and ever-expanding field of research and there exist quite a few (partially conflicting) definitions, a general and widely accepted description goes like this (Briskorn et al., 2011): Robust optimization is the task of finding a feasible solution to an optimization problem that is not necessarily optimal but remains feasible (solution-robustness) and has a good performance (quality-robustness) if parameter values of the problem change. Surveys on the vast amount of literature on robust optimization are, for instance, provided by Ben-Tal and Nemirovski (2002, 2008) and Kouvelis and Yu (2013). Robust scheduling in particular is surveyed by Aytug et al. (2005). Among the manifold approaches to reach robust schedules, a convenient way, which often leads to still comparatively compact problems, is the insertion of time buffers. A buffer between two jobs executed by the same resource protects the start time of the latter job against delays of the completion time of the former one. Time buffers have, for instance, been considered in project scheduling (e.g., Herroelen and Leus, 2004, Van de Vonder et al., 2005) and machine scheduling (e.g., Leus and Herroelen, 2007, Briskorn et al., 2011). We, however, have fixed execution intervals for our jobs (i.e., trailer movements), so that time buffers are only influenced by their assignment to resources (i.e., spotters). Fixed processing intervals are the topic of interval scheduling (see Kolen et al., 2007, for a survey), where to the best of our knowledge robustness and time buffers have not been considered yet.

The resulting robust interval scheduling problems are general problems, which are not bound to spotter scheduling, but can be applied whenever predefined (transport) jobs with given processing intervals have to be executed in a timely manner by a given set of machines (or vehicles). However, in the following, we stick to the case of spotter

scheduling because in this area of application the need to derive robust plans was brought to our attention.

D.1.2. Contribution and paper structure

This paper treats robust spotter scheduling by inserting time buffers between job pairs successively executed by the same spotter. Specifically, we assign transport jobs with given processing intervals to a fixed fleet of spotters under two robustness objectives: (i) maximize the minimum time buffer and (ii) maximize the sum of time buffers. The resulting two optimization problems are defined in Section D.2. Polynomial-time algorithms for solving both problems to optimality are provided in Section D.3. Our computational study (Section D.4) simulates different disturbances during plan execution and evaluates the ability of our two robustness approaches to avoid turmoil in the yard. Our results show that robust schedules, especially if optimized according to objective (i), can successfully avoid that delays of single trailer movements or delayed external trucks lead to a propagation effect causing further delays for subsequent yard processes. In this way, the average waiting time per truck and the average idle time per dock door reduces by more than 1 and 5 h, respectively, compared to non-robust solutions. Finally, Section D.5 concludes the paper.

D.2. Problem description

Consider a set $J = \{1, \ldots, n\}$ of jobs, each representing a fixed transport request defined by its completion time C_j, a processing time p_j, and a weight w_j. Jobs represent transport requests either from a parking position to a dock or vice versa. The input parameters of our problem can directly be derived from the output of the truck scheduling problem, which we assume to be pre-defined. The truck schedule for each truck determines when and where it has to be (un-)loaded, i.e., it assigns each trailer to a specific dock door and fixed processing time interval. As we assume that the parking positions for all trailers are also given, we can preprocess the travel time it takes a spotter to transport the trailer between the given dock and the respective parking position (i.e., p_j). A transport job of a trailer toward a dock consists of coupling a trailer to a spotter at the initial parking position and moving it toward the dock, where it is uncoupled. A retrieval of the same trailer after its processing reverts this flow: a spotter is coupled

to the trailer at its dock, moves it toward the parking position, and uncouples it. It is thus easy to derive the processing time p_j and the completion time C_j of each job j so that it is executed in a timely manner in order to meet the fixed predefined (un-)loading time intervals. The weights w_j denote the importance of a job being processed on time. E.g., an important truck that is fully loaded with high-priority freight may receive a greater weight than non-urgent deliveries. Furthermore, we have sequence-dependent setup times $\delta_{jj'}$, which represent the deadheading time it takes a spotter to move from the target position of job j to the start position of job j'. Note that we assume that the first job of each spotter can be processed without any driving time. As a matter of convenience, we assume jobs being numbered according to non-decreasing start times, i.e., $C_1 - p_1 \leq C_2 - p_2 \leq \cdots \leq C_n - p_n$.

Given job set J and a fixed fleet of spotters $S = \{1, \ldots, m\}$, a schedule is defined by a partition $\{R_1, \ldots, R_m\}$ of J and a permutation ω_s of R_s, $\forall s \in S$, specifying the order in which the jobs assigned to spotter s are executed. Let $\omega_s(l) \in R_s$ denote the l-th job of spotter s, i.e., $\omega_s = \langle \omega_s(1), \ldots, \omega_s(l), \ldots, \omega_s(|R_s|) \rangle$. Moreover, let $\eta(j) \in S$ be the spotter job $j \in J$ is assigned to, and let $\theta(j) \in \{1, \ldots, |R_{\eta(j)}|\}$ be the sequence position of job j. Then we define the buffer time $b(j)$ as the amount of idle time between job j and its predecessor $\omega_{\eta(j)}(\theta(j) - 1)$. If j is the first job executed by a spotter (i.e., $\theta(j) = 1$), it does not have a predecessor, therefore we set the buffer time to the start time of that job. Hence, we have

$$b(j) = \begin{cases} C_j - p_j, & \text{if } \theta(j) = 1 \\ C_j - p_j - \delta_{\omega_{\eta(j)}(\theta(j)-1),j} - C_{\omega_{\eta(j)}(\theta(j)-1)}, & \text{else} \end{cases},$$

$\forall j \in J$.

For simplicity's sake, we neglect the spotters' initial driving times ahead of their first job. Thus, we assume that each spotter is initially directly available at its first job at the beginning of the planning horizon. Note that if for each spotter $s \in S$ a specific initial state (i.e., an earliest availability time e_s and an initial driving time δ^s_{0j} toward each job $j \in J$) has to be considered, the buffer time of each initial job j (i.e., if $\theta(j) = 1$) can be computed as $b(j, s) = C_j - p_j - \delta^s_{0j} - e_s$ for each spotter $s \in S$. Such a setup would allow multiple schedule updates throughout the day, e.g., when planning on rolling planning horizons. In our baseline model, however, we do not explicitly take the spotters' initial states into account which remain a valid task for future research.

We say that a schedule is feasible if it satisfies the following conditions.

1. Each spotter is assigned at least one job, i.e., $R_s \neq \emptyset$, $\forall s \in S$. Note that it can obviously not be optimal with regard to robustness to have unused spotters. Also note that this presupposes that $n \geq m$, which can always be imposed by "culling" excess spotters from the instance if necessary.

2. The buffer time between any two jobs must not be negative (no spotter can perform two jobs at the same time), i.e., $b(j) \geq 0$, $\forall j \in J$.

The main idea of a buffer time $b(j)$ between job j and its predecessor $j' = \omega_{\eta(j)}(\theta(j)-1)$ is to protect the start time of j. If an unforeseen prolongation of unloading some truck leads to a delay of job j', then the start time of its successor j is not affected if the delay is not greater than $b(j)$. Otherwise, j cannot start as planned but is delayed less than it would be without a buffer. This way, a diffusion effect, i.e., the propagation of a delay at one dock to other docks via delayed spotters, is avoided or at least mitigated. Specifically, we consider the time buffers in two different robustness objectives.

- Objective max-$\sum$ maximizes the total weighted buffer time, i.e., $F^{sum} = \sum_{j \in J} w_j \cdot b(j)$.

- While the former objective maximizes the overall buffer time, it does not preclude individual buffers from being small (or even zero). To afford each job some level of protection, objective max-min maximizes the minimum buffer size, i.e., $F^{min} = \min\{b(j)/w_j \mid j \in J; \theta(j) > 1\}$. Note that, for this objective, only the buffer times between jobs (as opposed to the start time of the first job) are considered because the buffer before the first job only depends on its given start time and thus should not influence the objective.

Example: Consider an example problem with $n = 4$ transport jobs to be processed by $m = 2$ spotters. The processing (p_j) and completion (C_j) times as well as the weights (w_j) of each job are in Table D.1(a). The driving time from the end point of some job j to the starting point of any other job j', i.e., $\delta_{jj'}$, can be found in Table D.1(b). For this problem, the optimal solution with regard to objective max-$\sum$ is $\omega_1 = \langle 1, 2, 3]$ and $\omega_2 = \langle 4]$, that is, one spotter performs job 1, then job 2 and job 3, while the other spotter handles job 4. This leads to buffer times of $b_1 = 3$, $b_2 = 0$, $b_3 = 0$, and $b_4 = 11$, and hence to a total objective value of $F^{sum} = 1 \cdot 3 + 2 \cdot 0 + 1 \cdot 0 + 3 \cdot 11 = 36$. The solution is depicted as a Gantt chart in Figure D.2(a). Using the max-min objective, the

optimal solution becomes $\omega_1 = \langle 1, 3]$ and $\omega_2 = \langle 2, 4]$, implying buffer times of $b_1 = 3$, $b_2 = 6$, $b_3 = 1$, and $b_4 = 2$, and hence, an objective value of $F^{min} = \min\{1/1, 2/3\} = 2/3$. Figure D.2(b) shows the corresponding Gantt chart.

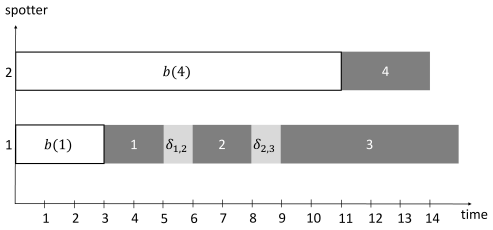
Table D.1.: Example problem: input parameters

j	1	2	3	4
p_j	2	2	6	3
C_j	5	8	15	14
w_j	1	2	1	3

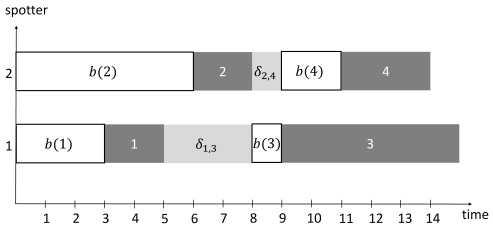
(a) Jobs' characteristics

$\delta_{jj'}$	1	2	3	4
1	–	1	3	2
2	1	–	1	1
3	3	1	–	4
4	2	1	3	–

(b) Driving times



(a) Gantt chart for max-$\sum$ example.



(b) Gantt chart for max-min example.

Figure D.2.: Example problem: optimal solutions

D.3. Algorithms

In this section, we will present polynomial-time exact algorithms for the robust spotter scheduling problem (RSSP), both under the max-$\sum$ as well as the max-min objective.

D.3.1. Objective max-$\sum$

Maximizing the total buffer time, RSSP can be reduced to maximum weight perfect matching in a bipartite graph.

Let $G(U \cup V, E, c)$ be a bipartite graph, where U and V are two disjoint vertex sets, E is a set of edges, with each edge connecting one vertex in U to one vertex in V, and $c : E \to \mathbb{R}_+$ is a weight function. $U = \{u_1, \ldots, u_n, s_1, \ldots, s_m\}$ and $V = \{v_1, \ldots, v_n, e_1, \ldots, e_m\}$

each contain one vertex for each of the n elements in job set J and m additional "dummy" vertices. We say that vertices v_j and u_j correspond to job j, $\forall j = 1, \ldots, n$, and vertices s_l and e_l correspond to spotter l (note that spotters are homogeneous and the indices for the latter are therefore interchangeable).

Each "dummy" vertex s_l in set U is connected to each job vertex v_j in set V by an edge $(s_l, v_j) \in E$, $\forall l = 1, \ldots, m; j = 1, \ldots, n$. Edge (s_l, v_j) indicates that job j is the first job to be performed by spotter l. Consequently, the weight of that edge is $c(s_l, v_j) = w_j \cdot (C_j - p_j)$.

Each job vertex u_j in set U is connected to a job vertex $v_{j'}$ in set V by an edge $(u_j, v_{j'}) \in E$ if and only if job j' can be performed after job j by the same spotter, i.e., if $C_{j'} - p_{j'} - \delta_{j,j'} - C_j \geq 0$. Such an edge stands for one spotter performing job j' immediately after job j. In that case, the edge weight is $c(u_j, v_{j'}) = w_{j'} \cdot (C_{j'} - p_{j'} - \delta_{j,j'} - C_j)$, i.e., the weighted time buffer between jobs j and j', given that they are processed in direct succession by the same spotter.

Finally, each job vertex u_j in set U is connected to each "dummy" vertex e_l in set V by an edge $(u_j, e_l) \in E$, $\forall j = 1, \ldots, n; l = 1, \ldots, m$, indicating that job j is the last job performed by a spotter. The corresponding edge weight is then $c(u_j, e_l) = 0$.

This definition of G allows stating the following proposition.

Lemma D.1. *A maximum weight perfect matching in graph G corresponds to a feasible RSSP solution which is optimal with regard to the* max-$\sum$ *objective and vice versa.*

Proof. A feasible RSSP solution can easily be constructed from a perfect matching in G: If some edge (s_l, v_j) is part of the matching, it means that spotter l starts off with job j. If then there is an edge $(u_j, v_{j'})$ in the matching, that means that job j' is the direct successor of j. Finally, if there is an edge $(u_j, e_{l'})$, it means that job j is the last job of spotter l. Note that l' need not necessarily be equal to l. Also, note that every job must be performed by exactly one spotter, and each spotter must perform at least one job, else the matching would not be perfect. Finally, seeing that edge weights correspond to weighted buffer times, it is clear that a maximum weight matching implies optimality with regard to F^{sum}.

By the same logic, this transformation can also be applied in reverse to create a maximum weight perfect matching in G from a feasible RSSP solution which is optimal with regard to F^{sum}. □

Example (cont.): The maximum weight matching for our example is graphically depicted in Figure D.3, corresponding to the same solution presented in Figure D.2a.

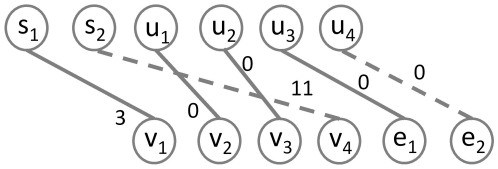


Figure D.3.: Optimal maximum weight matching in the example.

As the number of (potentially useful) spotters m is bounded by the number of jobs n, the total number of vertices in G is bounded by $O(n)$ and the total number of edges by $O(n^2)$. Consequently, using, for example, the improved Hungarian algorithm (e.g., Burkard et al., 2009, Ch. 4), we get the following proposition.

Proposition D.1. *RSSP with* max-$\sum$ *objective can be solved in* $O(n^3)$ *time.*

D.3.2. Objective max-min

We split solving RSSP with F^{min} objective into two parts. First, we will discuss how to find a feasible solution to RSSP (if one exists) for some given minimum time buffer $\overline{F}^{min}$, i.e., a feasible solution where $b(j)/w_j \geq \overline{F}^{min}$, $\forall j \in J; \theta(j) > 1$.

This problem can again be reduced to finding a perfect matching in a bipartite graph. Let $\mathcal{G}(U \cup V, \mathcal{E})$ be a bipartite graph, where U and V are two disjoint vertex sets and $\mathcal{E}$ is a set of edges, each connecting a node in U to a node in V. Sets U and V are set up exactly as described in Section D.3.1.

Set $\mathcal{E}$ contains edges (s_l, v_j) as well as (u_j, e_l), $\forall l = 1, \ldots, m; j = 1, \ldots, n$, indicating that job j is the first or last job executed by a spotter, respectively. Moreover, there is an edge $(u_j, v_{j'})$ if and only if $(C_{j'} - p_{j'} - \delta_{j,j'} - C_j)/w_{j'} \geq \overline{F}^{min}$, meaning that job j' can only be performed right after job j by the same spotter if the weighted time buffer between these two jobs is not below the given minimum $\overline{F}^{min}$.

Lemma D.2. *There is a perfect matching in* $\mathcal{G}$ *if and only if there is a feasible RSSP solution where* $b(j)/w_j \geq \overline{F}^{min}$, $\forall j \in J; \theta(j) > 1$.

Proof. If there is a perfect matching in $\mathcal{G}$, it can be transformed to a feasible RSSP

solution exactly as explained in the proof of Lemma D.1. In such a solution $b(j)/w_j \geq \overline{F}^{min}$, $\forall j \in J; \theta(j) > 1$, must hold because of the way set $\mathcal{E}$ is constructed.

Conversely, if there is a feasible solution where $b(j)/w_j \geq \overline{F}^{min}$, $\forall j \in J; \theta(j) > 1$, holds, there must also be a perfect matching in the corresponding graph $\mathcal{G}$. In such an RSSP solution, every job must clearly be the first or the last job of exactly one spotter, and/or it must be the successor of exactly one other job, such that the weighted time buffer between those jobs does not fall below $\overline{F}^{min}$. In either case, the corresponding edge is part of $\mathcal{E}$. Since this holds for every job in the RSSP solution, $\mathcal{G}$ must permit a perfect matching. □

Input: instance of RSSP
1 $B := \emptyset$; `// set of different buffer values`
2 **for** $j = 1$ **to** $n - 1$ **do**
3 | **for** $j' = j + 1$ **to** n **do**
4 | | **if** $C_{j'} - p_{j'} - \delta_{j,j'} - C_j \geq 0$ **then**
5 | | | $B := B \cup \{(C_{j'} - p_{j'} - \delta_{j,j'} - C_j)/w_{j'}\}$;
6 | | **end**
7 | **end**
8 **end**
9 $\mathcal{b} :=$ sorted list containing the elements in B in ascending order;
10 $q^{min} := 1$;
11 $q^{max} := |B|$;
12 **repeat**
13 | $q := \lfloor (q^{min} + q^{max})/2 \rfloor$;
14 | $\overline{F}^{min} :=$ the q-th element from $\mathcal{b}$;
15 | try to find a perfect matching using $\overline{F}^{min}$ as per Lemma D.2;
16 | **if** *perfect matching exists* **then**
17 | | $q^{min} := q + 1$;
18 | | $\Omega :=$ RSSP solution derived from perfect matching;
19 | **else**
20 | | $q^{max} := q - 1$;
21 | **end**
22 **until** $q^{min} > q^{max}$;
Output: RSSP solution Ω that is optimal with regard to the max-min objective

Algorithm 1: Solving RSSP with max-min objective.

Solving the matching problem as it is outlined above allows constructing a feasible solution for a given objective value of $\overline{F}^{min}$. If the optimal objective value F^{min*} was known, it would, therefore, be possible to generate a corresponding optimal solution. To find the optimal objective value, we propose the scheme outlined in Algorithm 1. This procedure is essentially a binary search scheme: The optimal max-min objective value must be one of the possible weighted time buffers between jobs, i.e., $F^{min*} =$

$(C_{j'} - p_{j'} - \delta_{j,j'} - C_j)/w_{j'}$, for some $j, j' \in J; j' > j$. In Algorithm 1, the set of these potentially optimal $\overline{F}^{min}$ values is denoted as B. Sorting these values in ascending order (list $\mathcal{b}$ in Algorithm 1), finding the greatest feasible and hence optimal $\overline{F}^{min}$ can be implemented as a binary search on $\mathcal{b}$. If a given $\overline{F}^{min}$ value does not permit a feasible solution, the $\overline{F}^{min}$ value must be too ambitious and is lowered in the next iteration. On the other hand, if a feasible solution can be constructed, then the current $\overline{F}^{min}$ may be too low and it is increased in the next iteration.

Example (cont.): The potentially optimal values, sorted in ascending order, for $\overline{F}^{min}$ are $\mathcal{b} = \langle 0, 2/3, 1, 4/3]$. The greatest feasible value from $\mathcal{b}$ is $2/3$. The perfect matching for $\overline{F}^{min} = 2/3$ is depicted in Fig. D.4, corresponding to the same solution presented in Figure D.2b.

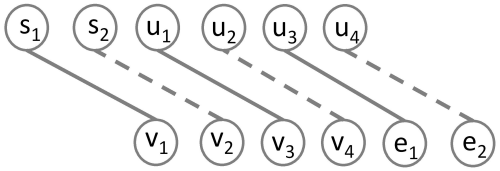


Figure D.4.: Perfect matching for $\overline{F}^{min} = 2/3$.

This leads us to the following proposition.

Proposition D.2. *RSSP with* max-min *objective can be solved in* $O(n^3 \log n)$ *time.*

Proof. The set of possible $\overline{F}^{min}$ values, B, can contain at most $O(n^2)$ distinct values. Since binary search halves the search space in each iteration, it can sift through these values in $O(\log n^2)$ iterations, and in each iteration, a perfect matching problem in a bipartite graph has to be solved, which can be done in $O(n^3)$ time. Consequently, the total asymptotic runtime comes to $O(n^3 \log n)$. □

D.4. Computational study

The objective of our computational study is to answer the following research questions. First, what is the computational performance of our proposed solution methods with regard to both objectives and different yard sizes (Section D.4.2)? Second, do our surrogate objectives indeed promote robustness to operational disturbances in a yard terminal? Since no established testbed for RSSP instances exists, we describe how our

test instances are generated in Section D.4.1. To address the second research question, we apply a terminal simulation, whose setup is described in Section D.4.3. In the simulation study, we investigate the robustness of RSSP solutions in case of unexpected disturbances (Section D.4.4). Specifically, we observe the effects of internal delays, i.e., unexpectedly prolonged (un-)loading times at docks, and external delays, i.e., delayed arrivals of trailers at the terminal.

D.4.1. Instance generation

Our instance generation scheme is geared to the hub terminal of the aforementioned postal service provider. Specifically, we first generate realistic truck schedules, and then, based on those truck schedules, the corresponding RSSP instances are derived.

RSSP is an operational planning problem which has to be solved once truck schedules (i.e., the where and when of trailer processing at the dock doors) are known. The typical planning horizon for the truck scheduling problem is one day (Boysen et al. (2017)). Thus, it is typically pointless to plan spotter schedules more than one day in advance. Moreover, to reduce forecasting errors of input parameters (e.g., truck arrival times, amount, and type of trailer loads, and their (un-)loading durations), it may be advisable to schedule spotters for an even shorter planning horizon (e.g., one or even half a work-shift). Given the operational character of RSSP, we aim to generate precise spotter schedules. Therefore, to avoid rounding errors, a time unit in an RSSP instance corresponds to one second of realtime.

Our trailer yard consists of a terminal and a parking lot. The terminal contains a set D of dock doors where the goods are to be unloaded from incoming trailers and/or loaded into outgoing trailers. The parking lot is divided into L parking lines, each of which is located parallel to the docking wall and contains $|D|$ parking positions. Thus, a set Π of parking positions in the terminal yard consists of $|\Pi| = |D| \cdot L$ positions. Our instance generator receives the number of dock doors $|D|$ and the number of parking lines L as input parameters that define the trailer yard size according to the dimensions of our real-world terminal.

We assume that there are two-way roads in the whole yard so that spotters can move in both directions everywhere. This way, we can easily derive the distance $d_{\alpha,\beta}$ between any pair of positions α and β in the trailer yard ($\alpha, \beta \in \{D \cup \Pi\}$). The schematic layout of a trailer yard with $|D| = 5$ dock doors and $L = 3$ parking lines is depicted in Figure

D.5. Note that a detailed description of how the distances in the yard are calculated is given in Appendix D.6.

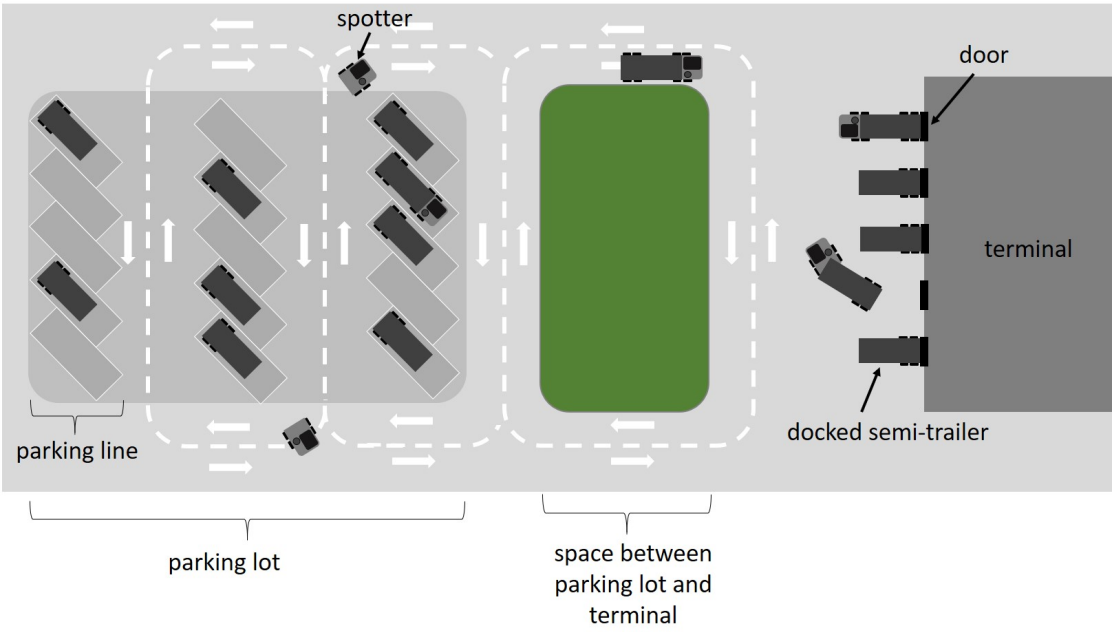


Figure D.5.: Schematic layout of the trailer yard for instance generation.

In line with the planning parameters applied at our real-world case, we assume that a spotter without attached semi-trailer moves with a constant speed of $v^f = 5$ m/s (i.e., 18 km/h) while the speed of a spotter coupled with a semi-trailer is assumed to be $v^b = 3$ m/s (i.e., 10.8 km/h). Furthermore, we assume that semitrailer coupling and uncoupling (including parking and pulling out) always last $\Delta^c = 30$ and $\Delta^u = 20$ s, respectively.

As a first step, to generate a feasible truck schedule for a given set of trucks I, for each truck $i \in I$, we randomly determine its arrival time a_i, parking position π_i in the yard, and handling time h_i at the dock door (i.e., duration of the (un-)loading of the trailer). For the sake of simplicity, we assume that the number of trucks $|I|$ in the planning horizon is equal to the total number of parking positions in the parking lot ($|I| = |\Pi|$). This way, each truck $i \in I$ is associated with exactly one parking space and vice versa. Note that most large trailer yards are located in rural areas where land is not that costly. Thus, this simplifying assumption should seldom be a shortcoming. We assign each trailer i to a parking position π_i randomly while taking into account that to each parking position π_i, exactly one trailer is assigned ($\pi_i \neq \pi_{i'} | \forall i, i' \in I; i \neq i'$).

It is well known that the truck scheduling problem can be modeled as a parallel machine scheduling problem (Tadumadze et al., 2019, 2020b). Thus, we generate arrival times and (un-)loading times of trucks similarly to the instance generation scheme proposed by Hall and Posner (2001). Specifically, handling time h_i for each trailer $i \in I$ is drawn from a normal distribution with an expected handling time of 30 min and variance of 6 min (i.e., $\mu = 1800, \sigma^2 = 6$). Furthermore, to generate arrival times a_i for each truck $i \in I$, the inter-arrival times of two successive trucks are randomly drawn from an exponential distribution with mean $\lambda = \frac{\mu}{|D|}$ (starting at $a_1 = 1800$). Note that all input parameters are assumed to have deterministic values.

To derive a truck schedule, we first sort trailers according to their arrival times a_i and then assign each trailer $i \in I$ to the next free dock door $g_i \in D$ and (un-)loading time window "first come first serve." As a result, for each truck $i \in I$, start (s_i) and end ($e_i = s_i + h_i$) of (un-)loading is determined.

Finally, for each trailer $i \in I$, the relative importance r_i of its truckload is drawn randomly with a uniform distribution from the interval $[1, 5]$.

Each trailer i is associated with exactly two jobs for spotters j and j':

- j: transport of trailer i from parking position π_i to dock door g_i with processing time $p_j = \Delta^c + \frac{d_{\pi_i,g_i}}{v^b} + \Delta^u$ (i.e., transport time of the trailer from the parking position to the dock door) and completion time $C_j = s_i$ (i.e., the trailer transport job is completed when the trailer has been set down at the door and its (un-)loading starts),
- j': return the trailer from g_i to π_i with $p'_j = \Delta^c + \frac{d_{g_i,\pi_i}}{v^b} + \Delta^u$ and $C'_j = e_i + p'_j$.

Setup times $\delta_{jj'}$ for each pair of jobs $j, j' \in J$ are determined by dividing $d_{\beta_j,\alpha_{j'}}$ (the distance between the end position β_j of job j and the start position $\alpha_{j'}$ of job j') by constant speed v^f of an empty spotter, i.e., $\delta_{jj'} = \frac{d_{\beta_j,\alpha_{j'}}}{v^f}$. The weight of each job $j \in J$ is derived as $w_j = r_i \cdot rnd[0.5, 2]$ where rnd denotes a uniformly distributed random number from the interval in the argument. This way, the weight of the job on the one hand depends on the relative importance of the corresponding trailer and, on the other hand, it is randomly re-weighted in respect of some other job-specific issues (e.g., urgency at the corresponding dock door, or a pressing truck departure time).

We have implemented our algorithms, including the shortest augmenting path algorithm by Jonker and Volgenant (1987) for finding a (maximum weighted) perfect matching in a

bipartite graph, as well as the simulation model in C# 6.0. All tests have been executed on an x64 PC with an Intel Core i7-8700K 3.70 GHz CPU and 64 GB RAM. The generated RSSP instances are available and can be downloaded using the following DOI: 10.5281/zenodo.3925356.

D.4.2. Computational performance

In this section, we investigate and compare the computational performance of the proposed algorithms (i.e., max-$\sum$ and max-min). To do so, we generate RSSP instances with our instance generator. Specifically, we assume three differently sized terminals consisting of $|D| = 20$, $|D| = 50$, and $|D| = 200$ doors (in the rest of the text the three different terminal sizes are dubbed S, M, and L). For each terminal size, we presuppose four different numbers $L \in \{2, 3, 4, 5\}$ of parking lines which, together with $|D|$, determine the size of the parking yard $|\Pi|$, the number of trucks $|I|$, and the number of transport jobs $n = 2 \cdot |I|$ in the planning horizon. Furthermore, for each terminal size, we assume three different spotter fleet sizes m. The applied input parameters are summarized in Table D.2. For each parameter combination, our instance generator generates 10 RSSP instances, so that in total there are 360 RSSP instances for the computational performance test.

Table D.2.: Parameters for instance generation of computational performance test

	Terminal size		
Parameter	S	M	L
$\|D\|$	20	50	200
L	(2,3,4,5)	(2,3,4,5)	(2,3,4,5)
$\|I\|$	(40,60,80,100)	(100,150,200,250)	(400,600,800,1000)
n	(80,120,160,200)	(200,300,400,500)	(800,1200,1600,2000)
m	(20,30,40)	(50,75,100)	(200,300,400)

Table D.3 summarizes the aggregated computational results of the algorithmic performance test. Each row in Table D.3 contains the results averaged over the 10 problem instances of the same size. For each algorithm (i.e., max-$\sum$ and max-min), we report the average computational time that is required to obtain the optimal solution (column "CPU sec"). For comparison, we also evaluate both objective values F^{min} and F^{sum} (columns "F^{min}" and "F^{sum}," respectively), even if the other objective function is pursued. Thus, we solve RSSP under the max-$\sum$ objective and evaluate the resulting optimal solution

Table D.3.: Results of computational performance test

Instance size		max-$\sum$			max-min		
m	n	CPU sec	F^{sum}	F^{min}	CPU sec	F^{sum}	F^{min}
Terminal size: S							
20	80	0.01	673,732.80	0.10	0.02	529,380.10	109.30
30	80	0.01	8877,59.20	0.20	0.02	754,772.10	259.33
40	80	0.01	1,067,874.70	2.50	0.02	976,983.00	284.50
20	120	0.02	957,800.50	0.03	0.05	688,786.00	107.75
30	120	0.02	1,363,439.70	0.00	0.05	1,071,680.30	221.38
40	120	0.02	1,616,270.30	0.00	0.06	1,333,432.80	278.46
20	160	0.04	1,130,077.30	0.00	0.12	777,390.70	90.10
30	160	0.04	1,640,356.70	0.00	0.12	1,194,765.80	210.72
40	160	0.04	2,065,274.30	0.00	0.13	1,617,540.10	277.67
20	200	0.06	1,316,235.40	0.00	0.24	834,793.70	61.13
30	200	0.07	2,030,708.50	0.00	0.26	1,442,855.30	213.05
40	200	0.07	2,577,453.60	0.00	0.27	1,947,120.30	281.76
Mean (S)		*0.03*	*1,443,915.25*	*0.24*	*0.11*	*1,097,458.35*	*199.60*
Terminal size: M							
50	200	0.08	2,801,623.40	0.35	0.27	2,234,450.30	271.04
75	200	0.08	3,384,937.90	0.65	0.29	2,958,384.40	261.79
100	200	0.08	3,999,989.70	1.68	0.32	3,732,210.60	286.27
50	300	0.22	3,895,340.30	0.00	0.95	2,843,258.20	268.39
75	300	0.24	5,458,433.00	0.18	1.07	4,367,426.00	288.90
100	300	0.25	6,300,740.30	0.33	1.09	5,420,223.30	280.31
50	400	0.46	5,142,676.80	0.00	2.41	3,567,202.20	277.59
75	400	0.49	7,124,042.70	0.03	2.53	5,354,420.40	285.28
100	400	0.53	8,992,119.00	0.00	2.70	7,223,840.40	298.59
50	500	0.83	6,435,699.20	0.00	4.97	4,265,949.10	251.84
75	500	0.90	9,238,427.70	0.08	5.28	6,603,872.90	266.86
100	500	0.96	11,516,821.00	0.08	5.56	8,846,761.80	297.87
Mean (M)		*0.43*	*6,190,904.25*	*0.28*	*2.29*	*4,784,833.30*	*277.89*
Terminal size: L							
200	800	4.09	32,045,392.00	0.00	26.34	25,975,279.00	272.16
300	800	4.40	38,253,184.00	0.13	27.60	34,427,745.00	278.55
400	800	4.54	46,529,532.00	0.25	30.00	44,087,069.00	266.96
200	1200	11.97	53,432,412.00	0.00	93.21	39,669,969.00	274.18
300	1200	13.21	68,786,194.00	0.00	97.63	56,381,906.00	276.93
400	1200	14.17	79,158,212.00	0.00	100.31	69,536,510.00	265.67
200	1600	26.16	73,286,205.00	0.00	226.35	51,825,850.00	265.62
300	1600	29.02	102,827,090.00	0.00	241.58	79,509,343.00	286.44
400	1600	30.82	119,421,480.00	0.00	246.04	98,797,397.00	263.93
200	2000	48.74	96,623,679.00	0.00	456.86	65,889,463.00	264.14
300	2000	52.79	132,979,290.00	0.00	473.17	97,712,393.00	284.98
400	2000	56.47	158,843,290.00	0.00	481.95	125,592,390.00	269.38
Mean (L)		*24.70*	*83,515,496.67*	*0.03*	*208.42*	*65,783,776.17*	*272.41*

with the objective function of the max-min problem (and vice versa). Note that the F^{min} values are only optimal when the max-min problem is solved; analogous for F^{sum} and max-$\sum$.

Our experiment shows that both algorithms are well suited for solving RSSP to optimality in short computational time. Even in the most extreme cases, the computational times of the hardest problem instances for the large terminal yard (scheduling $n = 2,000$ jobs on $m = 300$ or $m = 400$ spotters) do not exceed 1 min when total weighted buffer time (i.e., objective max-$\sum$) is maximized and 8 min in case of objective max-min, respectively. For all problem instances, the computational times under the max-$\sum$ objective are always shorter compared to the max-min case. On the other hand, maximizing the total weighed buffer time apparently comes at the cost of low-weighted jobs receiving very short or even no buffer times. This can be seen in the F^{min} values of the solutions where the min $-\sum$ objective is applied, which are always close to zero (column "F^{min}" for the max-$\sum$ objective). This may damage the robustness of solutions, because in case of unforeseen events during plan execution, delays can propagate. We investigate this further in Section D.4.4.

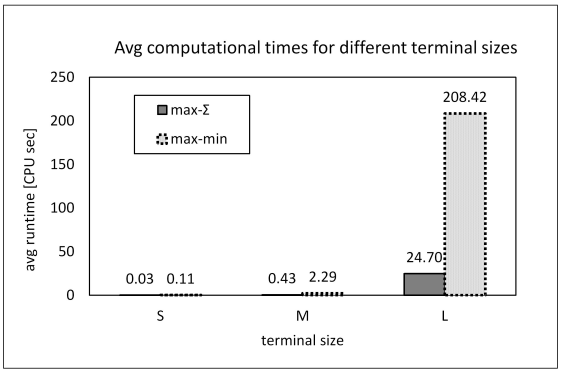


Figure D.6.: Impact of terminal size on computational times

Unsurprisingly, the terminal size strongly affects the computational times of both algorithms. The average computational times of both algorithms for the three terminal sizes (i.e., S, M, and L) are depicted in Figure D.6. We further observe the impact of n (i.e., the number of transport jobs) and m (i.e., the number of spotters) on problem complexity. According to the experimental results, the spotter fleet size m has only a negligible impact on the problem complexity, while the number of transport jobs n seems to have a considerable influence on the computational time. Figure D.7 depicts the average computational times for instances with varying numbers of jobs n for each terminal

size. In all cases, with the increase in parameter n, the average computational times of both algorithms increase super-linearly, although max-$\sum$ scales better than max-min.

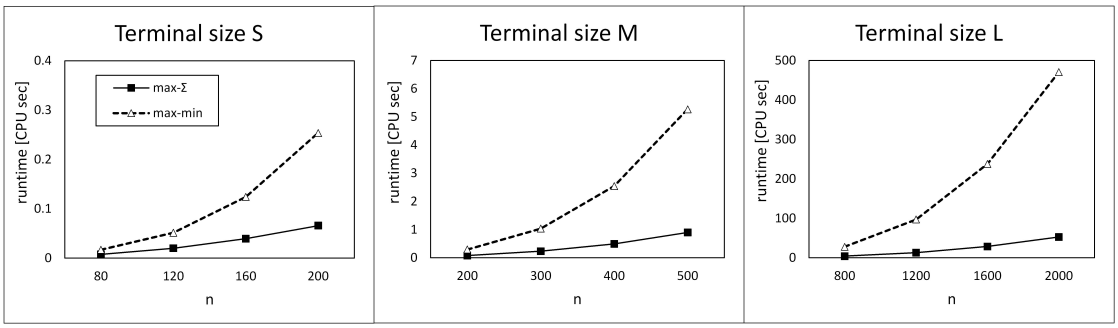


Figure D.7.: Impact of n on computational times

It can be concluded from the performance test that our solution algorithms seem well suited for practical applications. They deliver optimal solutions quickly even for very large yards, and the solution times are short enough to promptly return solutions whenever a replanning may be required during daily operations.

D.4.3. Setup of simulation study

In this section, we investigate whether RSSP solutions are actually robust against unforeseen delays that may occur during the execution of spotter schedules. To do so, we build a simulation model where some random events are simulated and the robustness of the solutions is evaluated. In the following, we describe the setup of our simulation model.

As a prototype terminal of our simulation model we select a medium-sized terminal with $|D| = 50$ doors, $|\Pi| = 300$ parking positions, and $|I| = 300$ trucks leading to $n = 600$ transport jobs. Furthermore, we vary parameter m (i.e., the number of spotters) between the following nine different values: $m \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$. Again for each fleet size, we generate ten RSSP instances, so that for the simulation study we generated 90 RSSP instances in total.

We aim to observe the robustness of both objectives against unexpected disturbances. Therefore, for each generated instance, we simulate some random delays. In particular, we consider the following two kinds of disturbances:

- *External disturbance* a^+: Some trucks may arrive later at the terminal yard than

expected. The original arrival time a_i of delayed truck i is modified to $a'_i = a_i + a_i^+$, where a_i^+ denotes the amount of delay of truck i.

- *Internal disturbance* h^+: The (un-)loading of some semitrailers may last longer than expected so that the handling time of such a trailer i at the dock is prolonged $h'_i = h_i + h_i^+$ with h_i^+ defining the amount of delay.

For both kinds of disturbances our simulation model receives two predefined input parameters:

- the probability that a disturbance for a truck occurs: $p(a^+)$ and $p(h^+)$ as well as
- the maximal amount of delay [in min] that can occur: $max(a^+)$ and $max(h^+)$.

Given these parameters, for each truck i the simulation model first randomly decides whether or not it arrives with delay. If so, the amount of delay a_i^+ is drawn randomly from a uniform distribution in the interval $[0, max(a_i^+)]$. Internal delays are determined analogously. To observe the effects of delays, we vary the input parameters in the following ranges $p(a^+) \in \{5, 10, 20\}$ [%] and $max(a^+) \in \{10, 20, 30, 40, 50, 60\}$ [min] as well as $p(h^+) \in \{5, 10, 20\}$ [%] and $max(h^+) \in \{10, 20, 30, 40, 50, 60\}$ [min]. The input parameters of our simulation study are summarized in Table D.4.

Table D.4.: Parameters for instance generation of simulation study

Parameter [unit]	Value(s)
$\|D\|$	50
L	6
$\|I\|$	300
n	600
m	(20,30,40,50,60,70,80,90,100)
$p(a^+)$ [%]	(5,10,20)
$max(a^+)$ [min]	(10,20,30,40,50,60)
$p(h^+)$ [%]	(5,10,20)
$max(h^+)$ [min]	(10,20,30,40,50,60)

After generating these random disturbances, an original RSSP instance P can be modified to P'. To do so, we reassign each truck $i \in I$ to the new (un-)loading time window according to the modified parameters a'_i and h'_i. Note, however, that we keep the door assignment of each truck as derived in the original truck schedule. This is because changing the door assignment on short notice triggers new problems inside the terminal. For

instance, the goods dedicated to a specific outbound trailer are typically assembled (e.g., picked from a warehouse) over a longer period of time and placed in the outbound area directly in front of the respective dock door. Changing the dock door on short notice would require to move these goods to the new dock, which produces double handling and increases the probability of misplaced shipments.

Once our instance generator has produced an RSSP instance P, we solve it with regard to both objectives and store the corresponding optimal spotter schedules $\Omega(P)^*_{sum}$ and $\Omega(P)^*_{min}$. Moreover, to benchmark our robust spotter schedules against non-robust schedules, we additionally solve P to feasibility (by finding a perfect matching in the corresponding initial bipartite graph without considering any objectives) and store the resulting solution $\Omega(P)_{feas}$ too. Subsequently, our (simulated) disturbances occur, which leads to the modified RSSP instance P'. We evaluate the modified instance P' with the three fixed schedules (i.e., $\Omega(P)^*_{sum}$, $\Omega(P)^*_{min}$, and $\Omega(P)_{feas}$) and observe whether the new job completion times C'_j of P' are violated. Specifically, each spotter $s \in S$ executes the transport jobs in the given sequence ω_s according to the fixed spotter schedule ($\Omega(P)^*_{sum}$, $\Omega(P)^*_{min}$, or $\Omega(P)_{feas}$), which represent the schedules derived prior to the occurrence of disturbances. Given these fixed spotter schedules originally determined for P, we can derive whether our two robustness measures protect us against the disturbances of P'.

To compare the robustness of the schedules, we compute the lateness $l_j = \max\{0; \frac{C'_j - t_j}{60}\}$ of each job $j \in J$ [measured in min], where t_j denotes the actual execution time of job j (i.e., the execution time of job j when P' is evaluated according to original spotter schedule $\Omega(P)^*_{sum}$, $\Omega(P)^*_{min}$, or $\Omega(P)_{feas}$). Then based on the lateness of the job, we derive the following three performance measures, which evaluate the solution quality (i.e., robustness):

- *Average weighted lateness*: We define the average weighted lateness AWL of all jobs as the weighted sum of every jobs' lateness divided by the total number of jobs n: $AWL = \frac{\sum_{j \in J} w_j \cdot l_j}{n}$.

- *Truck-related lateness*: To reduce the delay of goods, terminal yards aim to provide (external) trucks with their (loaded or empty) semitrailers in a timely manner just as agreed. Thus, our external performance measure, average truck-related lateness TRL, calculates the total lateness of those jobs that are associated with bringing trailers from the terminal back to their parking position divided by the number of

trucks $|I|$.

- *Dock-related lateness*: While the previous performance measure is geared to the external stakeholders of the yard, our third performance measure, average dock-related lateness DRL, is primarily relevant for the internal processes within the terminal. Specifically, DRL measures the total lateness of those transport requests whose target position is a dock door divided by the number of dock doors $|D|$. Such lateness leads to unforeseen waiting times at the dock doors and delays the processing of incoming goods inside the terminal.

The lower these three lateness measures for the schedules, the larger their protection level against unforeseen disturbances, which is evaluated in the following section.

D.4.4. On the impact of robust spotter scheduling

In this section, we benchmark the robustness of our two objectives max-$\sum$ and max-min if unforeseen disturbances occur during the execution of a spotter schedule. To do so, we solve the 90 generated RSSP instances to feasibility as well as with regard to both objectives and store the solutions. Then, we simulate disturbances, evaluate the modified data with the original spotter schedules derived for the initial (undisturbed) problem instance, and calculate our three lateness measures.

First, we observe robustness in case of only *external disturbances*, i.e., when some trucks arrive later than expected. For this, we simulate each problem instance in 18 different scenarios, where the values of external disturbance parameters, $p(h^+)$ and $max(h^+)$, are varied in the ranges given in Table D.4. Recall that the higher the former (latter), the more often external disturbances occur (the larger the delay of a late truck). Afterward, we address exclusively *internal disturbances*, i.e., some semitrailers require a longer processing time than expected. Here, we vary the probability $p(h^+)$ of the prolongation of a trailer's handling time at a dock and the maximal duration $max(h^+)$ of the delay. Finally, we investigate the most realistic scenario when both kinds of disturbances—i.e., internal and external delays—occur simultaneously. Specifically, we simulate both kinds of delays at the same time by varying all parameters $p(a^+) = p(h^+)$, and $max(a^+) = max(h^+)$ in the ranges given in Table D.4. As a result, each of the generated 90 instances is simulated in 54 different scenarios, each of which is solved and evaluated three times (to feasibility and with regard to both objectives).

Aggregated results of our simulation study are visualized in Figures D.8, D.9 and D.10. Specifically, Figures D.8 and D.9 display the results for either external or internal disturbances, respectively, while Figure D.10 shows the results when both kinds of disturbances occur simultaneously.

Compared with non-robust solutions, applying the right robustness measure, i.e, the max-min objective, allows to protect against disturbances, and the lateness can be almost completely avoided (see slightly rising slopes of all three performance measures using the $\max - \min$ objective). The optimal schedules with regard to our max-$\sum$ objective do not deliver results as good as those optimized using the max-min objective. This is due to objective max-$\sum$ generally favoring solutions where high-priority jobs (i.e., those with large weight w_j) receive generous buffers, while some or many other transport jobs have (almost) no buffer. Since any delay of a single job—even if it is a low-priority job—can lead to cascades of further delays of successive jobs, it is helpful to protect all jobs with buffer time, which is what the max-min objective promotes.

In general, the rising curves in all figures indicate that the robustness of initial schedules gets worse when the expected delay duration increases. Furthermore, the steeper slopes of the curves in scenarios with higher delay probabilities show deteriorating initial schedules when delays occur more frequently. Improvements of the robust schedules, i.e., optimized with regard to the max-min objective, compared with non-robust schedules are more remarkable for the scenario when both kinds of disturbances occur rather than when either exclusively internal or external delays are simulated. This can be seen in the steeper slopes of the curves in Figure D.10 than the slopes from Figures D.8 and D.9. Note that the different graphics apply varying scaling factors and bounds on the y-axis.

Spotter schedules optimized with regard to the $\max - \min$ objective can effectively protect the terminal from dramatic cascades of delays. Even for the most extreme scenario, i.e., when both kinds of disturbances occur with the largest probability and the longest delay duration ($p(h^+) = p(a^+) = 0.2$ and $max(h^+) = max(a^+) = 60$), the average weighted lateness is still reasonable when spotter schedules are optimized according to the max-min objective: $AWL \leq 44$. Thus, in spite of the disturbances, external trucks can be processed almost always on time as planned ($TRL \leq 15$ min) and our bottleneck resource, the dock doors, do not suffer considerably from delayed jobs ($DRL \leq 63$ min). Non-robust random solutions, however, lead to considerable delays. For the same scenarios, our lateness measures exceed $AWL \geq 277$, $TRL \geq 81$, and $DRL \geq 376$. In other

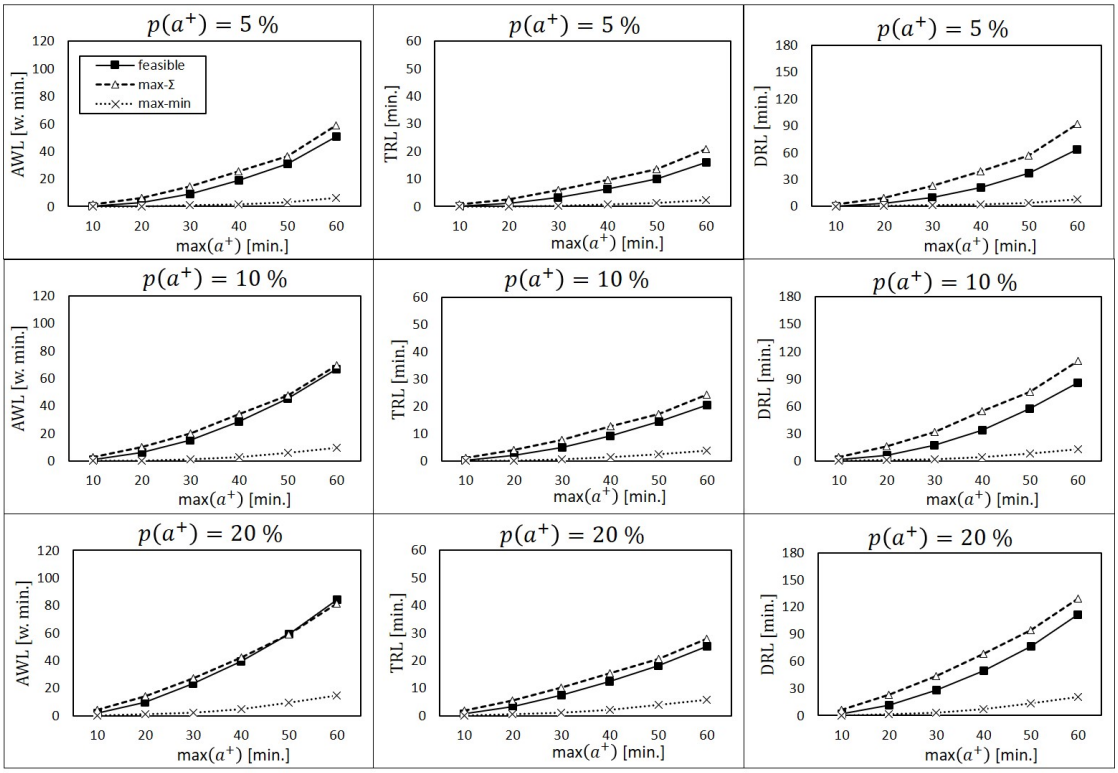


Figure D.8.: Comparison of solution robustness in case of external delays

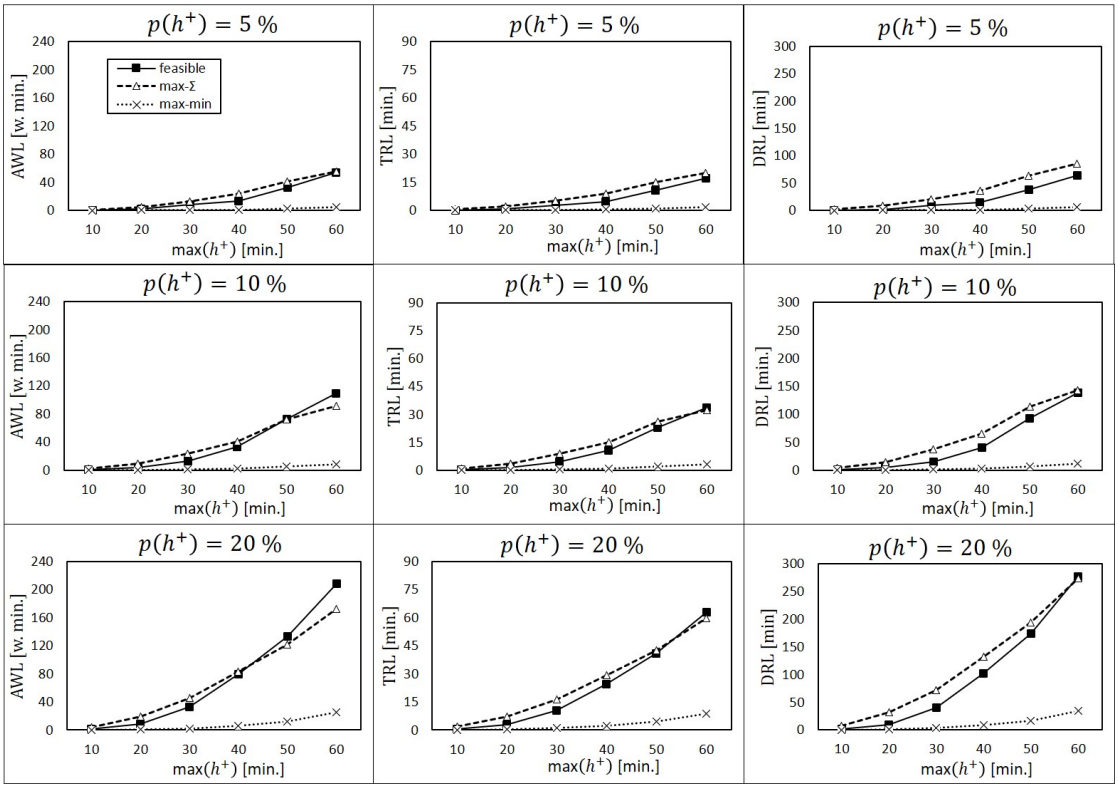


Figure D.9.: Comparison of solution robustness in case of internal delays

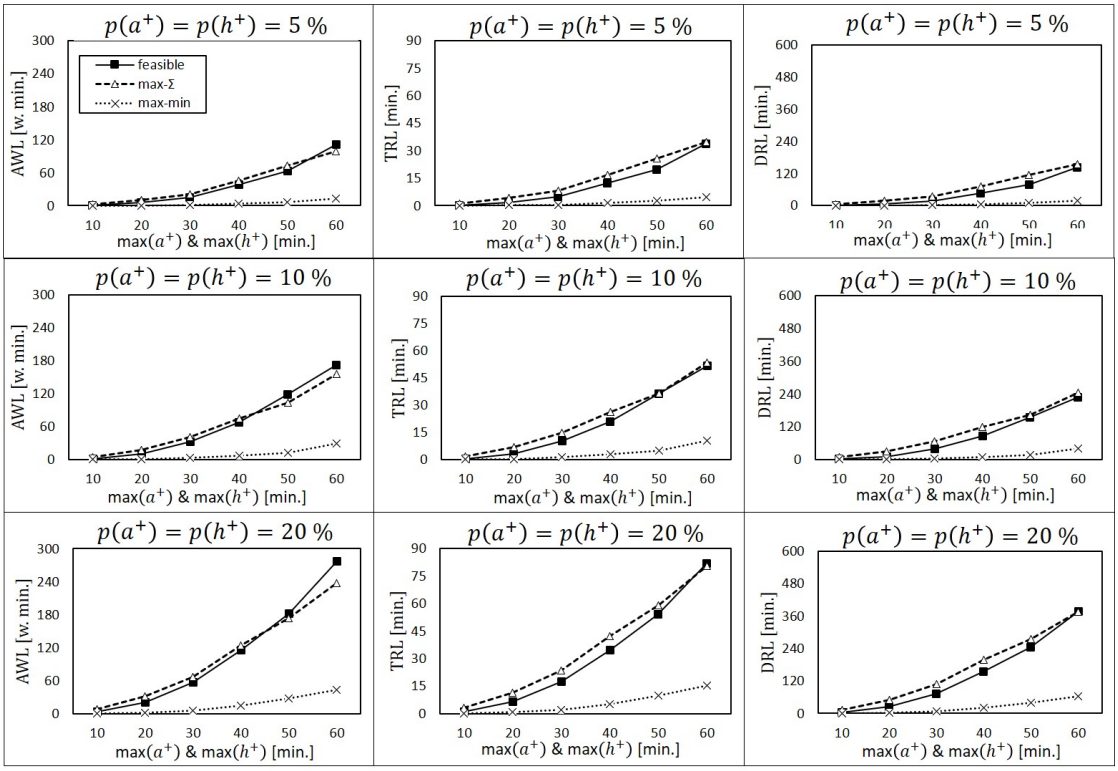


Figure D.10.: Comparison of solution robustness in case of internal and external delays

words, the average waiting time of trucks can be reduced by about 80% (from more than 81 min to less than 15 min) and the average waiting time of a dock door can be reduced by more than 83% (from more than 376 min to less than 63 min) if instead of non-robust schedules a suited robustness measure (i.e., the max-min objective) is applied.

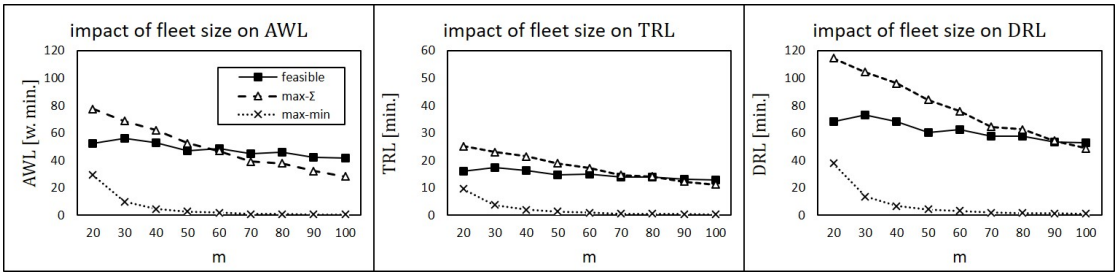


Figure D.11.: Impact of spotter fleet size on solution quality

Moreover, we explore the impact of the spotter fleet size m on robustness. In particular, we observe the average values of our three performance measures when executing the spotter schedules derived for the original instances P on the disturbed instances P' with different fleet sizes. The aggregate results are displayed in Figure D.11. We would expect additional spotters to make it easier to increase buffer times since there is more flexibility to shift jobs among the vehicles. And indeed, this is what the results indicate. The decreasing curves of all three charts show an improvement of solution quality (i.e., less lateness) when the spotter fleet size grows. Note that this observation is clearly visible for the robust schedules (i.e., $\max - \min$ and $\max - \sum$), while the spotter fleet sizes seem to have less impact on non-robust (i.e., random) schedules. These results can be helpful from a managerial viewpoint at the tactical planning level when deciding on the fleet size. For larger trailer yards, which require a greater spotter fleet anyway, the additional flexibility of swapping jobs among spotters enables sufficient protection against disturbances (when planning with the right robustness objective). For smaller yards and smaller spotter fleets, the pooling effect is much smaller, so that additional stand-by spotters should be procured to support the permanent fleet during peak-hours.

To get a better idea of how objectives $\max - \sum$ and $\max - \min$ are interrelated, we maximize the weighted sum of buffer times for a given aspiration level of F^{min}. Specifically, we first forbid all solutions with an F^{min} value of less than the given aspiration level, deleting the matching edges from the corresponding graph as described in Section D.3.2. Then, we solve the RSSP instance with regard to our $\max - \sum$ objective.

To illustrate this approach, we solve a random RSSP instance with $n = 500$ jobs, $m = 150$

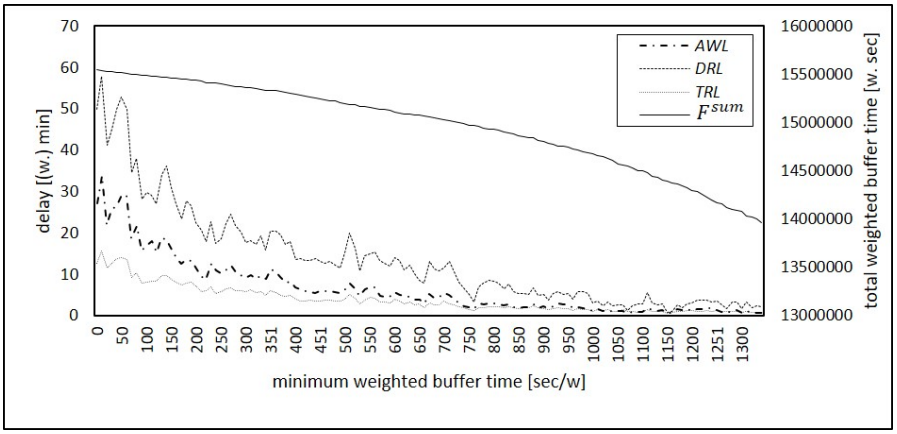


Figure D.12.: Simulation results using $\max - \sum$ objective with given aspiration level for F^{min}

spotters, and $D = 50$ dock doors. The aspiration level F^{min} is iteratively increased from 0 by 10, until it reaches the point where the RSSP instance cannot be solved to feasibility. Further, we simulate internal and external disturbances with our simulation model using the parameters $p(h^+) = p(a^+) = 0.2$ and $max(h^+) = max(a^+) = 60$ and evaluate robustness by measuring AWL, DRL, and TRL.

Figure D.12 visualizes the Pareto frontier for the RSSP instance (i.e., optimal objective values F^{sum} for given aspiration levels F^{min}) and the robustness measures of each Pareto optimal solution. As expected, F^{sum} and F^{min} are inversely correlated: the higher F^{min}, the lower F^{sum}, because a high minimum buffer constrains the feasible search space among which to pick the best F^{sum} solution. However, although a higher value of F^{min} leads to lower (i.e., worse) F^{sum} values, it promises better scores on each of three performance measures AWL, TRL, and DRL. These results support the previous findings that our $\max - \min$ objective yields spotter schedules that are more robust than those of the $\max - \sum$ objective.

D.5. Conclusion and future research

This paper aims to reduce turmoil in large trailer yards resulting in considerable waiting times for the trucks up to several hours in many real-world applications (for examples see Section D.1). Existing solution approaches primarily address the demand side and aim to spread out truck arrivals over the day, e.g., by applying internet booking plat-

forms. We, however, show that robust internal schedules can successfully avoid that unforeseen delays propagate among successive jobs and considerable waiting times accumulate. We specifically address the assignment of transport jobs (i.e., semitrailers to be moved between the trailer yard and the dock doors of a terminal) to a fleet of spotters. By introducing time buffers among successive jobs assigned to the same vehicle, our computational study shows that waiting times of trucks and idle times at the dock doors of the terminal can be reduced considerably. In a simulation study, especially the max-min objective, which maximizes the minimum time buffer among all successive job pairs, is shown to successfully protect schedules against unforeseen delays. We present a solution procedure for the resulting robust optimization problem that is solvable in polynomial time so that even for very large trailer yards and long planning horizons, optimal solutions can be obtained quickly.

Our robust solution approaches are not bound to spotter scheduling in trailer yards, so that future research should evaluate other applications of our robust interval scheduling. A systematic benchmark study including multiple applications and alternative robustness measures would be a valid contribution to further promote the integration of time buffers to protect against uncertainty in a simple and straightforward manner. Thanks to the short computational times of the proposed algorithms, they can also be applied in a dynamic environment where unforeseen events require a repeated (re-)planning on rolling horizons. The only adaption of our solution approaches for such an application is that some spotters are still blocked at the beginning of the current planning horizon by jobs not yet completed. While the algorithmic adaption is truly straightforward, future research should evaluate our solution approaches when planning on rolling horizons. Further computational tests should also compare the proposed approaches with online scheduling, stochastic optimization, and alternative robust optimization techniques. Another important future research task may integrate spotter scheduling with truck scheduling in a holistic approach since these two problems are clearly heavily interdependent. Recall that the proposed algorithms solve the spotter scheduling problem in polynomial time and may thus be part of a decomposition scheme.

Appendix

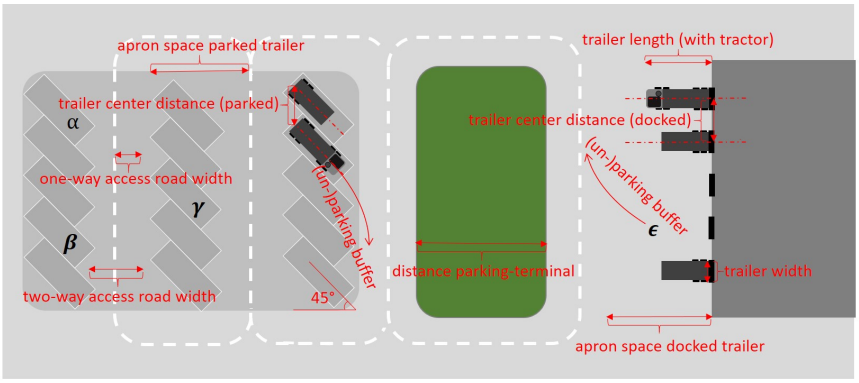
D.6. Appendix: Distance calculations in the trailer yard

This appendix provides a more detailed explanation of how the distances in the yard within our test instances are calculated. It, thus, complements Section D.4.1.

When calculating distances, we take into consideration dock and yard safety distances, apron spaces between docked and parked trailers, minimum road widths, and parking angles as defined by the widespread dock planning standards elaborated in NTI (2013). Table D.5 summarizes the values of the applied safety distances in the yard, which are visualized for an example layout in Figure D.13. We derive the shortest distances between any pairs of positions in the trailer yard as the sum of total horizontal and vertical distances including the (un-)parking buffers. Figure D.13 includes some example distance calculations between four selected positions $\alpha, \beta, \gamma, \epsilon \in \{D \cup \Pi\}$.

Table D.5.: Input (safety) parameters for trailer yard

Parameter	Values	
	Foot(') / inch (")	Meter
Trailer length (with tractor)	53'	16.1544
Trailer width	96"	2.4384
Trailer center distance (docked)	12'	3.6576
Trailer center distance (parked)	12'	3.6576
One-way access road width	13'	3.9624
Two-way access road width	26'	7.9248
Apron space docked trailer	159'	48.4632
Apron space parked trailer	88'	26.8224
Distance parking - terminal	328.084'	100



$d_{\alpha\beta}$ = 2 × (un-)parking buffer + 3 × trailer center distance (parked)
vertical distance
vertical distance

$d_{\alpha\gamma}$ = 2 × (un-)parking buffer + 6 × trailer center distance (parked) + 1 × (apron space parked trailer + one-way access road width)
horizontal distance

vertical distance

$d_{\alpha\epsilon}$ = 2 × (un-)parking buffer + 5 × trailer center distance (parked) +
+ 2 × (apron space parked trailer + 6 × one-way road width + distance parking-terminal) + apron space docked trailer-trailer length
horizontal distance

Figure D.13.: Yard layout with safety parameters and distance calculations

Chapter E.

Unrelated parallel machine scheduling*

Abstract: This chapter addresses scheduling a set of jobs with release dates and deadlines on a set of unrelated parallel machines to minimize some minmax objective. This family of problems has a number of applications, e.g., in discrete berth allocation and truck scheduling at cross docks. We present a novel exact algorithm based on logic-based Benders decomposition as well as a heuristic based on a set partitioning reformulation of the problem. We show how our approaches can be used to deal with additional constraints and various minmax objectives common to the above-mentioned applications, solving a broad class of parallel machine scheduling problems. In a series of computational tests both on instances from the literature and on newly generated ones, our exact method is shown to solve most problems within a few minutes to optimality, while our heuristic can solve particularly challenging instances with tight time windows well in acceptable time.

*This chapter has been published as: Tadumadze, G., Emde, S., Diefenbach, H. (2020): Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. *OR Spectrum* 42, 461–497.

E.1. Introduction

Scheduling problems concern themselves with assigning tasks to limited resources (processors / machines) over time. As such, they play a central role in most logistics and production processes. In this context, this paper deals with variations of the following basic problem.

Given are a set of jobs $J = \{1, \ldots, n\}$, where each job is associated with a release date $r_j \in \mathbb{R}^+$ and a deadline $\overline{d}_j \in \mathbb{R}^+$, and a set of processors (or machines) $P = \{1, \ldots, m\}$, where it takes $p_{ij} \in \mathbb{R}^+$ time units to process job j on machine i. The machines are unrelated, meaning that some jobs may be processed faster on some machines than on others, and preemption is not allowed. On what processor and at what time should each job start processing such that no two distinct jobs are processed at the same time on the same machine, the time windows are observed, and some minmax objective is optimized? In classic machine scheduling literature, using the notation introduced by Graham et al. (1979), this problem is denoted as $[R|r_j, \overline{d}_j|\cdot]$.

A schedule S for $[R|r_j, \overline{d}_j|\cdot]$ is defined as a set of triples $(i, j, t) \in S$, indicating that job $j \in J$ is assigned to processor $i \in P$ starting at time $t \in \mathbb{R}^+$. We say that a schedule is feasible if it meets the following conditions.

- Each job is assigned exactly once, i.e., for each job $j \in J$, there is exactly one triple $(i, j, t) \in S$.
- No job is scheduled before its release date or finishes processing after its deadline, i.e., $\forall (i, j, t) \in S$, it must hold that $r_j \leq t \leq \overline{d}_j - p_{ij}$.
- No two jobs may occupy the same processor at the same time, i.e., for each pair of triples $(i, j, t), (i', j', t') \in S$, $j \neq j'$, it must hold that $i \neq i'$ or $t + p_{ij} \leq t'$ or $t' + p_{i'j'} \leq t$.

Scheduling with time windows on unrelated parallel machines has multiple practical applications. One prime example is berth allocation at maritime container terminals, where calling vessels are assigned time and space at the quay wall. We can equate ships with jobs and berths with processors. Another example is truck scheduling at distribution centers or manufacturing plants, where incoming trucks have to be assigned to dock doors. In this context, incoming trucks correspond to the jobs and dock doors to processors. In both of these applications, it is very common to have given arrival

and departure times (i.e., time windows) for each transport device (i.e., vessel or truck). These applications are discussed in more detail in Section E.2.

Inspired by these applications, we discuss the following generalizations of the base problem.

- *Processing set restrictions:* Especially in the berth allocation context, it is possible that certain ships are incompatible with certain berths due to draft limitations (e.g., Xu et al., 2012). In machine scheduling parlance, this means that certain jobs are incompatible with certain machines.
- *Machine availability:* Especially in a rolling horizon framework, some machines may not be ready before a given time (e.g., Imai et al., 2001, in the berth scheduling context). In this case, no job may be scheduled on a given machine before the machine is ready.
- *Tails:* Jobs may have tails (also called delivery times), where they do not block any machine but still remain active. This is especially relevant for truck scheduling applications, where it may not be enough for a truck to finish processing at a door before the due date, but the shipments need to actually arrive at their destinations (e.g., another door for transshipment) to be considered on-time (e.g., Tadumadze et al., 2019).

We investigate three alternative minmax objectives that are relevant for both of the above-mentioned applications.

- *Makespan:* It is typically desirable to quickly clear the terminal (i.e., have the last job finish as soon as possible) for successive planning runs (e.g., Boysen and Fliedner, 2010). Consequently, a schedule S is optimal with regard to makespan if it minimizes
$$C(S) = \max_{(i,j,t)\in S}\{t + p_{ij}\}. \tag{E.1}$$
- *Maximum (weighted) flow time:* It is important for terminals to serve every customer (i.e., transport device) in a timely manner. The flow time of a job corresponds to the total service time of the transport device in the terminal (i.e., the time from its arrival until it finishes processing) and is calculated as the difference between its completion time C_j and its release date r_j. In the literature, it is common to minimize the (weighted) sum of flow times $\sum w_j F_j$, where each job

$j \in J$ may additionally have a priority weight w_j ($w_j > 0$), indicating its relative importance (e.g., Cordeau et al., 2005, Boysen, 2010). While $\sum w_j F_j$ leads to low (weighted) service times for the average customer, it does not preclude long service times for some individual customers. We therefore consider the maximum (weighted) flow time $wF_{\max}$ which aims to minimize the service time of the worst served customer, defined as:

$$wF_{\max}(S) = \max_{(i,j,t)\in S}\{w_j \cdot (t + p_{ij} - r_j)\}.$$

- *Maximum (weighted) lateness:* In real life, it is not always possible to achieve a feasible solution without violating any time windows. In such cases, a more useful objective may be to minimize the maximum violation of the due dates. In other words, deadlines $\overline{d}_j$ become due dates d_j and the corresponding constraint becomes soft. The lateness of job j is defined as the difference between its completion time C_j and its due date d_j. For many terminals, it may be reasonable to minimize the maximum (weighted) lateness $wL_{\max}$, defined for our problem as follows:

$$wL_{\max}(S) = \max_{(i,j,t)\in S}\{w_j \cdot (t + p_{ij} - d_j)\}.$$

The main contribution of this paper is a novel logic-based branch and Benders cut scheme, which is shown to solve many realistic problem instances to optimality in reasonable time, decisively outperforming a commercial optimization solver. To the best of our knowledge, it is the first such algorithm to be proposed for this family of unrelated parallel machine scheduling problems with time windows. For even larger and/or harder instances, we propose a heuristic procedure based on a generalized set partitioning formulation. Our exact and heuristic approaches are very flexible and can be easily adapted to account for problem generalizations and special cases as well as different minmax objectives, such that a broad class of parallel machine scheduling problems with minmax objectives can be solved. Our computational tests show that hundreds of realistically sized instances from both the literature and systematically generated ones can be solved to optimality in acceptable time.

The remainder of this paper is organized as follows. In Section E.2, we review the pertinent literature and discuss real-world applications of our scheduling problem in more detail. In Section E.3, we introduce a novel branch & Benders cut scheme as

well as a heuristic column selection algorithm for this problem, respectively. We extend our approaches to cope with additional constraints and alternative minmax objectives in Section E.4. In Section E.5 performance of our algorithms is tested on benchmark instances in a computational study. Finally, Section E.6 concludes the paper.

E.2. Literature review and applications

In the literature, the problem of scheduling a set of jobs with given release dates and deadlines on unrelated parallel machines to minimize the makespan is expressed by the triple $[R|r_j, \overline{d}_j|C_{\max}]$. For an overview on machine scheduling in general, see Pinedo (2016).

Concerning scheduling on unrelated parallel machines with a makespan objective in particular, it is extensively studied by Lenstra et al. (1990), who show that the approximation ratio for $[R||C_{\max}]$ is at least 3/2 unless $P = NP$. The same authors also develop a polynomial time 2-approximation scheme. An improved $(2-1/m)$-approximation scheme with computational time $O(m^2)$ is developed by Shchepin and Vakhania (2005). Furthermore, for the special case where each job can be assigned to at most two machines with the same processing time, Ebenlendr et al. (2014) proposed a 1.75-approximation algorithm. More recently, Knop and Koutecky (2017) introduced a method to solve the problem in $\Theta^{O(\Theta^2)} \cdot n^{O(1)}$ time, where $\Theta = \max_{i \in P, j \in J}\{p_{ij}\}^K$, and K is the number of different types of machines. Mokotoff and Chrétienne (2002) develop an exact and a heuristic algorithm based on a cutting plane scheme to solve $[R||C_{\max}]$. Another successful solution procedure to solve this problem based on a recovering beam search heuristic is proposed by Ghirardi and Potts (2005). In contrast to traditional beam search, their algorithm contains a recovering phase, which allows substitution of dominated solutions by alternative more promising solutions from the previous stage. Moreover, Fanjul-Peyro and Ruiz (2010, 2011) develop heuristics for $[R||C_{\max}]$ based on iterated greedy local search methods and size reduction heuristics. Further metaheuristics for $[R||C_{\max}]$ are presented by Lin et al. (2011) and Sels et al. (2015).

Regarding the inclusion of release dates, Lancia (2000) proposes a branch and bound scheme to schedule jobs with release dates and tails on two unrelated parallel machines, minimizing the makespan $[R2|r_j, q_j|C_{\max}]$. A similar scheduling problem for identical parallel machines subject to job release dates and tails minimizing the makespan is investigated by Gharbi and Haouari (2002). They develop branch and bound algorithms

and propose a new tight branching scheme for $[P|r_j, q_j|C_{\max}]$. In Section E.4, we discuss how our own algorithms can be extended to account for nonzero delivery times.

Regarding logic-based Benders decomposition for parallel machine scheduling, Jain and Grossmann (2001) propose a decomposition approach which combines solving a mixed-integer linear programming (MILP) model and a constraint programming (CP) model to solve an unrelated parallel machine scheduling problem with time windows to minimize the total cost, which depends on the job-machine assignment. Their approach, which is similar to logic-based Benders decomposition, is tested on instances with up to $n = 20$ jobs and $m = 5$ machines. Tran et al. (2016) study the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times and a makespan objective $[R|sds|C_{\max}]$. They develop a logic-based Benders decomposition approach where the master problem (i.e., assignment of jobs to the machines) is solved via MILP techniques and the subproblems (i.e., sequencing jobs) via a specialized solver for the traveling salesman problem (TSP). In a computational study, instances with up to $n = 60$ jobs and $m = 5$ machines are solved to optimality, while instances with up to $n = 120$ jobs and $m = 8$ machines are solved heuristically. Gedik et al. (2016) develop two logic-based Benders decomposition algorithms for a parallel unrelated machine scheduling problem with sequence dependent setup times. Their problem additionally considers availability intervals, and as each job is associated with a given profit and cost, their objective is to maximize the total profit. The authors compare their decomposition algorithms to extant ILP and CP models for instances with $n \in \{32, 57, 116\}$ jobs and $m \in \{10, 15, 20, 25, 30\}$ machines, where not all instances can be solved to optimality within a 3-hour time limit. Hooker (2007) develops another logic-based Benders decomposition algorithm for several versions of the unrelated parallel machine scheduling problem with regard to three different objectives (makespan, total tardiness and total costs). They additionally consider machine-job specific resource consumption rates c_{ij} and allow more than one job to be processed concurrently on machine i as long as the total resource consumption does not exceed the given resource limit C_i. They solve problems with up to $n = 50$ jobs, where not all instances can be solved to optimality within 2 hours.

In summary, most papers employing logic-based Benders decomposition for parallel machine scheduling use a constraint programming solver for the subproblems, the exception being Tran et al. (2016), who use a TSP solver. In this paper, we develop a novel bounded dynamic programming-based approach, which is shown to perform quite well and can easily be adapted to a broad range of machine scheduling problems (see Section E.4).

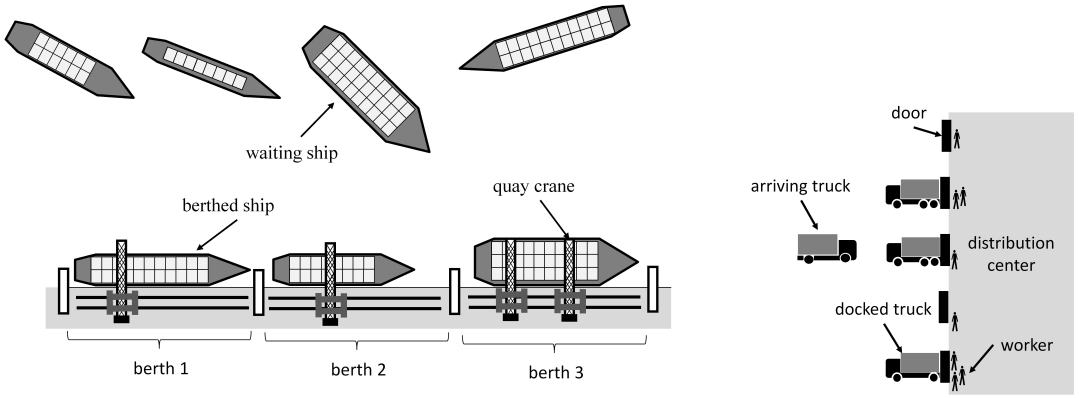
E.2.1. Berth allocation

Apart from a short dip during the financial crisis of 2008, international seaborne trade has been steadily on the rise for the past decades (UNCTAD, 2016). While container vessels are growing more numerous and larger, capacities (e.g., quay cranes and space at the quay) at container terminals are hard to expand due to physical and geographical constraints. For this reason, operations research methods have become very prominent in making the best use of limited resources at many container terminals (e.g., Steenken et al., 2004, Vacca et al., 2007). One of the most important problems in this context is the berth allocation problem, which aims to assign a set of calling vessels to both a berthing time and space at the quay wall, which is surveyed in Bierwirth and Meisel (2010, 2015). In this context, our problem is equivalent to the following basic berth allocation problem: Given a set of calling vessels (jobs) and a discrete set of berths (processors), which ship should moor at which berth at what time, such that no vessel berths before its arrival time, every vessel can depart before its deadline, and the last vessel departs as early as possible? In practice, berths may differ in term of the resources (e.g., quay cranes) and each vessel may have an "optimal berthing point" (e.g., depending on the storage location for its containers in the terminal). As a result, the ships' handling times at the quay wall may vary depending on where they are berthed, which makes berths equivalent to unrelated machines. A schematic representation of the discrete berth allocation problem is given in Figure E.1(a).

In the berth allocation context, minmax objectives have so far only been tackled with regard to very specific applications. The first application of a makespan objective in the berth allocation context is provided by Li et al. (1998), who consider a continuous quay wall, where ships can berth at any arbitrary position as long as they do not collide. Emde et al. (2014) investigate the berth allocation problem at ports that have a so-called mobile quay wall, which can enclose ships berthed at the normal, fixed quay wall. Similarly, the same performance criterion for a combined berth allocation and quay crane scheduling (or assignment) problem is applied by Lee and Qiu Wang (2010) and Blazewicz et al. (2011), respectively.

E.2.2. Truck scheduling

At many cargo handling facilities, like distribution centers, manufacturing plants or cross docks, calling trucks need to be assigned to a given set of dock doors for (un-)loading.



(a) Berth allocation. (b) Truck scheduling.

Figure E.1.: Applications of $[R|r_j, \overline{d}_j|C_{\max}]$.

This problem is commonly referred to as truck scheduling, which is surveyed by Boysen and Fliedner (2010). The existing literature almost without exception assumes that all dock doors process trucks equally fast, which is not always the case (e.g. Tadumadze et al. (2019), Konur and Golias (2017)). Since dock doors may differ with regard to staff levels and equipment, processing times may also depend on which truck is processed at which door. Critical trucks can be processed faster at dock doors with more workers, although not every truck profits equally from additional workers (depending on size and load). In this regard, dock doors can be interpreted as unrelated parallel machines. It is very common that trucks must be scheduled within fixed time windows. A distribution center handling incoming trucks is schematically depicted in Figure E.1(b).

E.3. Solution procedures

In this section, we present novel exact and heuristic algorithms to solve $[R|r_j, \overline{d}_j|C_{\max}]$. We later generalize these approaches to include additional constraints (Section E.4.1) and alternative minmax objectives (Sections E.4.2.1, E.4.2.2). The exact algorithm, based on logic-based branch and Benders cut, is presented in Section E.3.1. Since machine scheduling with time windows is well known to be strongly NP-hard for any $|P| \geq 1$ (Lenstra et al., 1977), we also propose a heuristic column selection approach in Section E.3.2.

E.3.1. Branch and Benders cut for $[R|r_j, \overline{d}_j|C_{\max}]$

Our exact logic-based branch & Benders cut (B&BC) algorithm is based on the classic Benders decomposition. The original algorithm proposed by Benders (1962) decomposes a problem into a master model, which is usually a relaxed version of the original model, and a slave model, which is used to iteratively generate feasibility and optimality cuts to be added to the master model. In our approach, we deviate from the classic Benders scheme by adding so-called logic-based (or combinatorial) Benders cuts in the spirit of Codato and Fischetti (2006) and Hooker (2011). Our master model is a type of bin packing problem with the goal of assigning jobs to processors. The slave problem consists of sequencing a given set of jobs on given processors, where the assignment of jobs to processors is determined by the current master solution, such that the makespan is minimal.

E.3.1.1. Master problem

Our master model contains only binary variables y_{ij}, which define the assignment of jobs to machines: $y_{ij} = 1$ if and only if job j is assigned to machine i. Moreover, we define $N(j) = \{j' \in J \mid r_{j'} \geq r_j\}$ as the set of jobs that are released no sooner than job j and $\overline{d}_{\max}(j) = \max\{\overline{d}_{j'} \mid j' \in N(j)\}$ as the latest deadline of the jobs from $N(j)$. Finally, let z be an auxiliary continuous variable denoting a lower bound on the completion time of the latest machine. Our master model is a relaxed version of the original problem, where we assume that a no-wait earliest release date (ERD) schedule is feasible, which it may not be. It is defined as the following mixed integer linear program.

$$\text{[Master] Minimize } F^M(Y, Z) = z \tag{E.2}$$

subject to

$$\sum_{i \in P} y_{ij} = 1 \qquad \forall j \in J \tag{E.3}$$

$$r_j + \sum_{j' \in N(j)} y_{ij'} \cdot p_{ij'} \leq z \qquad \forall i \in P; j \in J \tag{E.4}$$

$$r_j + \sum_{j' \in N(j)} y_{ij'} \cdot p_{ij'} \leq \overline{d}_{\max}(j) \qquad \forall i \in P; j \in J \tag{E.5}$$

$$y_{ij} \in \{0; 1\} \qquad \forall i \in P, j \in J \tag{E.6}$$

Objective (E.2) minimizes the makespan of the relaxed problem. Constraints (E.3) make sure that each job is assigned to exactly one machine. Valid inequalities (E.4) enforce that the lower bound z on the makespan cannot be less than lower bound on the makespan of each processor $i \in P$, which is the total processing time of the jobs assigned to processor i starting after time r_j. Valid inequalities (E.5) forbid such job-to-machine assignments that obviously lead to the violation of deadlines, i.e., when even the lower bound on the makespan on a machine exceeds a potential latest deadline $\overline{d}_{\max}(j)$ of the jobs on that machine. Constraints (E.6) define the binary variables. Note that it is possible to solve the model as a pure feasibility problem, i.e., without variable z and valid inequalities (E.4) and (E.5). However, it is expedient to add these valid inequalities to drive the search into the promising regions of the solution space.

We solve [Master] using a commercial black box solver. Whenever the solver hits upon a candidate integer solution $\bar{Y}$, the slave problem is solved to derive a feasible solution from $\bar{Y}$ (if any) by sequencing the jobs optimally for the given assignment. Therefore, from the viewpoint of the slave problem, the assignment of jobs to processors is already given; what remains is determining the optimal schedule of jobs on each machine.

E.3.1.2. Slave problem

To solve the slave problem, first off, the problem clearly decomposes along the processors: how we schedule jobs assigned to one processor does not affect how we schedule jobs assigned to another processor. Given an integer solution $\bar{Y}$ for model [Master], the set of jobs assigned to machine i is $\Gamma_i = \{j \in J \mid \bar{y}_{ij} = 1\}$. Now, scheduling jobs Γ_i on machine i is equivalent to solving a single machine scheduling problem with time windows to minimize the makespan, i.e., $[1|r_j, \overline{d}_j|C_{\max}]$. This problem is well known to be NP-hard in the strong sense (Lenstra et al., 1977). It stands to reason, however, that at least in the applications discussed in Section E.2, the number of jobs per processor will not be too high. Therefore, we propose the following bounded dynamic programming (BDP) scheme based on the general methodology introduced by Held and Karp (1962).

The dynamic program consists of $|\Gamma_i| + 1$ stages (index $l = 0, \ldots, |\Gamma_i|$), each stage containing states (Σ), where $\Sigma \subseteq \Gamma_i$ is the set of jobs already scheduled. Starting in dummy stage $l = 0$ from state ($\emptyset$), successors in stage $l + 1$ are reached by appending one of the not yet scheduled jobs to the schedule, i.e., the successor states of some state (Σ) are ($\Sigma \cup \{j\}$), $\forall j \in \Gamma_i \backslash \Sigma$. Let $\mathcal{C}(\Sigma)$ be the makespan of the (partial) schedule of

state (Σ) which can be recursively computed as

$$\mathcal{C}(\Sigma) = \min_{j \in \Sigma} \{\max\{\mathcal{C}(\Sigma \backslash \{j\}), r_j\} + p_{ij}\},$$

where $\mathcal{C}(\emptyset) := 0$. Further, for each state (Σ), we compute the lower bound on the makespan $LB(\Sigma)$ as follows:

$$LB(\Sigma) = \max_{j \in \Gamma_i \backslash \Sigma} \left\{ \max\{\mathcal{C}(\Sigma), r_j\} + \sum_{j' \in N(j) \cap \{\Gamma_i \backslash \Sigma\}} p_{ij'} \right\},$$

the idea being that the remaining jobs are appended in a no-wait ERD schedule, which is optimal but may be infeasible due to the deadlines.

In the dynamic programming graph, the number of nodes (i.e., states) increases exponentially with the number of jobs to be scheduled. However, we can significantly reduce its size by generating only such nodes that satisfy the following criteria.

1. It is still possible to schedule every remaining job with regard to its respective time window, i.e., $\mathcal{C}(\Sigma) + p_{ij} \leq \overline{d}_j$, $\forall j \in \Gamma_i \backslash \Sigma$, must hold.

2. The remaining jobs can at least theoretically still be feasibly scheduled without violating the latest deadline, i.e., $\mathcal{C}(\Sigma) + \sum_{j \in \Gamma_i \backslash \Sigma} p_{ij} \leq \max_{j \in \Gamma_i \backslash \Sigma} \{\overline{d}_j\}$ must hold.

3. Let UB be the objective value of the best known feasible solution, i.e., the global upper bound. Then it must hold that $LB(\Sigma) < UB$.

Moreover, if the optimal solution of the relaxed problem, obtained while computing $LB(\Sigma)$, is feasible for the original problem (i.e., every job j from Γ_i is scheduled within its time window), we fathom the state and, if applicable, store the found solution as the new best known feasible solution and the corresponding objective value as the local upper bound $UB(\Sigma)$ on this machine. If the local upper bound of the state is smaller than the global upper bound (i.e., if $UB(\Sigma) < UB$ holds), we replace UB with $UB(\Sigma)$ in criterion 3.

The optimal makespan of processor i given Γ_i equals $\mathcal{C}(\Gamma_i)$. The corresponding sequence of jobs is obtained by backward recovery along the optimal path. To determine the start times of the jobs, each job must be scheduled in the given sequence as early as possible, i.e., either when its predecessor ends or at its release date, whichever comes last. If no final state (Γ_i) exists—either because it is impossible to schedule the jobs in Γ_i without

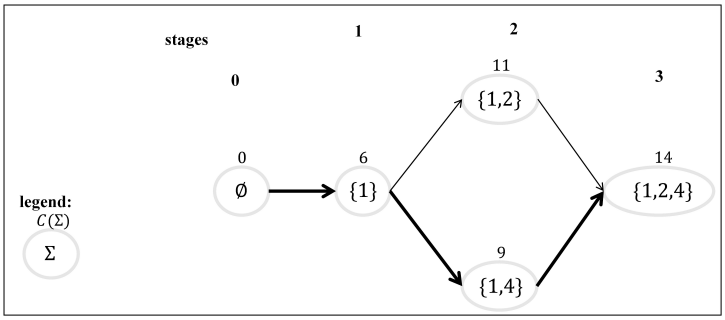
violating any time windows or because there is no schedule with a makespan of less than UB—we consider the "optimal makespan" for this processor to be $\mathcal{C}(\Gamma_i) := UB$.

Regarding the asymptotic runtime, BDP has $O(2^{|\Gamma_i|})$ states and $O(|\Gamma_i|2^{|\Gamma_i|})$ transitions, making the runtime exponential. However, as mentioned above, in many applications, the number of jobs per processor $|\Gamma_i|$ can be expected to be rather low; hence, solution times may still be acceptable for many practical uses.

Example: Consider the example data given in Figure E.2(a), consisting of $n = 4$ jobs and $m = 2$ processors. Assume that for some [Master] solution $\Gamma_2 = \{1, 2, 4\}$, i.e., jobs 1, 2, and 4 are assigned to processor 2. The resulting dynamic programming graph is in Figure E.2(b). One of the two optimal solutions is bold, corresponding to job sequence $\langle 1, 4, 2]$.

j	1	2	3	4
$p_{1,j}$	5	6	8	4
$p_{2,j}$	5	5	9	3
r_j	1	3	4	2
d_j	8	19	13	25

(a) Example problem data.



(b) Dynamic programming graph for the slave problem in the example.

Figure E.2.: Example data and dynamic programming.

Let $\mathcal{C}(\Gamma_i)$ be the optimal makespan of machine i given Γ_i, determined for each machine via BDP. Let $i^* = \arg\max_{i \in P}\{\mathcal{C}(\Gamma_i)\}$ be the machine with the greatest makespan and hence the machine that determines the makespan for the schedule as a whole. If $\mathcal{C}(\Gamma_{i^*})$ is lower than the current best upper bound UB, then a new best solution has been found. The solution is stored and the upper bound is updated, i.e., $UB := \mathcal{C}(\Gamma_{i^*})$. Moreover, the following cut is added to program [Master]:

$$z \leq \text{UB} - \epsilon,$$

where ϵ is a sufficiently small positive number. This way, solutions to model [Master], whose lower bound z is not less than the best upper bound UB, will be fathomed. Note that the upper bound can initially be set to $UB := \max_{j \in J}\{\overline{d}_j\}$ or to the best objective

value of some heuristic procedure.

Regardless of whether a new upper bound has been found, we determine the set $I = \{i \in P \mid \mathcal{C}(\Gamma_i) \geq \text{UB}\}$. To find a better solution with a lower makespan, at least one of the jobs on each of the machines in I must be moved to another machine. To enforce this, we add the following cuts to program [Master].

$$\sum_{j \in \Gamma_i} y_{ij} \leq |\Gamma_i| - 1 \qquad \forall i \in I.$$

Once the cuts are added as constraints, the solver continues solving model [Master] with these new cuts. When it hits upon another integer solution, the slave problem is again solved, new cuts are generated, and so on, until no more unexplored and unfathomed solutions remain for the master model. At that time, the search terminates and the best upper bound is optimal. This procedure is commonly referred to as (logic-based) branch & Benders cut (Rahmaniani et al., 2017, Emde, 2017). In Section E.5.2, we investigate whether it is beneficial to add cuts as so-called lazy or regular constraints.

E.3.2. Heuristic column selection based on generalized set partitioning

As a first step to solve $[R|r_j, \overline{d}_j|C_{\max}]$ heuristically, we formulate the problem as a generalized set partitioning problem (GSPP). This model formulation assumes all time related parameters (i.e., r_j, $\overline{d}_j$, and p_{ij}) to be integer. For practical purposes, this assumption can be imposed by scaling and rescaling the time unit before and after using the solution procedure, respectively. We divide the entire planing horizon in discrete time periods $t = 1, \ldots, T$. Note that $T = \max_{j \in J}\{\overline{d}_j\}$ is a sufficiently large value for the length of time horizon. Once the $[R|r_j, \overline{d}_j|C_{\max}]$ problem is formulated as a discrete time-indexed scheduling problem, it can be straightforwardly transformed to a GSPP as described below. Similar problem formulations and techniques have also proven successful, for instance, for discrete berth allocation (Buhrkal et al., 2011, Lalla-Ruiz et al., 2016) and for truck scheduling problems (Boysen et al., 2017, Tadumadze et al., 2019).

In the GSPP model, a column represents a feasible assignment of a job j to a machine i starting in period t. Let $M = \{(i, j, t) \in P \times J \times \{1, \ldots, T\} \mid r_j \leq t \leq \overline{d}_j - p_{ij}\}$ be the set of all columns. The following restrictions have to be considered to derive a feasible partition:

- For each job $j \in J$ exactly one column (i, j, t) from set M has to be selected, and
- On each processor $i \in P$ and in each period t ($t = 1, \ldots, T$), at most one job j can be processed.

Among all feasible partitions, we seek one that minimizes the makespan.

M	Set of columns
T	Number of periods (index $t = 1, \ldots, T$)
v_m	Binary variable: 1, column m is selected; 0, otherwise

Table E.1.: Additional notation for the GSPP model.

We define a binary variable $v_m \in \{0, 1\}$, which is 1 if column $m = (i, j, t)$ is selected. This way, the problem reduces to selecting one column m for each job j while taking into account that in no period t, more than one job is processed on each processor i. Additional notation for the GSPP model formulation is summarized in Table E.1. We formulate our GSPP as the following integer program.

$$\text{[GSPP] Minimize } G(V) = \max_{(i,j,t)=m \in M} \{v_m \cdot (t + p_{ij})\} \tag{E.7}$$

subject to

$$\sum_{\substack{(i,j',t)=m \in M: \\ j'=j}} v_m = 1 \qquad \forall j \in J \tag{E.8}$$

$$\sum_{\substack{(i',j,t')=m \in M: \\ i'=i \wedge t' \leq t < t'+p_{ij}}} v_m \leq 1 \qquad \forall t = 1, \ldots, T, i \in P \tag{E.9}$$

$$v_m \in \{0; 1\} \qquad \forall m \in M \tag{E.10}$$

Objective function (E.7) minimizes the makespan by minimizing the completion time of the latest job. Constraints (E.8) ensure that for each job exactly one column is selected. Furthermore, inequalities (E.9) take care that in no period more than one job is processed on the same processor. Finally, (E.10) sets the domain of the binary variables.

Finding a feasible solution to GSPP is strongly NP-hard (Garey and Johnson, 1979). However, the GSPP is an extensively studied problem and it offers some modeling advantages. This formulation allows us to heuristically reduce M to a small subset of

preselected columns. For all $i \in P$ and $j \in J$, let $M_{ij} = \{(i', j', t) \in M \mid i = i' \wedge j = j'\}$ be the subset of columns relevant for job j on machine i. As a simple approach to heuristically reduce M_{ij}, we only select the first and every μth column from the sorted set of columns, where they are sorted in ascending order according to their completion time $t + p_{ij}$, and μ is a predefined integer number ($\mu \geq 1$). As a result, the number of columns reduces to $|M_{ij}| = \left\lceil \frac{\bar{d}_j - r_j - p_{ij} + 1}{\mu} \right\rceil$. This way, by varying the value of parameter μ we can reduce the solution space to the desired size: A greater value of μ reduces $|M_{ij}|$ and the GSPP becomes easier to solve. On the other hand, smaller $|M_{ij}|$ increase the risk that the optimal (or promising) columns are not considered. Note that our heuristic does not guarantee finding a feasible solution (especially if μ has a high value). In such a case, μ must be reduced to a lower value. We conduct our own series of tests with regard to the best value of μ in Section E.5.2. Also note that if $\mu = 1$, model [GSPP] is equivalent to the original problem (assuming that all parameters are integer). We further investigate this trade-off in our computational study (Section E.5.2). Note that this simple heuristic column selection scheme is shown to be very successful in related problems and superior among other heuristic rules studied by Tadumadze et al. (2019).

Once the GSPP model is solved, we derive the job sequences for all machines and improve the machine schedules by starting all jobs as early as possible considering the given sequence. To do so, we set the start time of each job either to its release date or to the completion time of its predecessor job on the same machine. The ensuing makespan may be lower than the original GSPP objective value.

E.4. Extensions

The proposed algorithms can be easily modified for a broader class of parallel machine scheduling problems. In this section, some extensions of our approaches are presented. First, we discuss how the algorithms have to be adjusted if the solution space of the problem is changed, i.e., the problem contains some new or misses some existing constraints. Afterwards, we generalize our approaches for other minmax objectives than the makespan.

E.4.1. Special cases and generalizations

Since unrelated machine scheduling is a generalization of parallel machine scheduling, the proposed approaches can be also applied for solving its special cases: related (or uniform) machine scheduling with time windows $[Q|r_j, \overline{d}_j|C_{\max}]$ and identical parallel machine scheduling with time windows $[P|r_j, \overline{d}_j|C_{\max}]$. In the case of relaxed deadlines, i.e., $\overline{d}_j = \infty$, $\forall j \in J$, the problem becomes $[R|r_j|C_{\max}]$, which is still NP-hard in the strong sense (Garey and Johnson, 1979). However, in this case, for a given set Γ_i of jobs to be processed on machine i, the slave problem of our B&BC approach becomes $[1|r_j|C_{\max}]$, which can be solved as a single machine scheduling problem with the goal of minimizing the maximum lateness, i.e., $[1||L_{\max}]$, in polynomial time (Lawler, 1973). Analogously, the slave problem of special case $[R|\overline{d}_j|C_{\max}]$ (i.e., when release dates are not considered) becomes $[1|\overline{d}_j|C_{\max}]$ and can be solved to optimality by applying the earliest deadline rule (EDD). Note that the set $N(j) := \{j' \in J|\overline{d}_j \leq \overline{d}_{j'}\}$ for [Master] has to be redefined as the set of jobs whose deadlines are not earlier than $\overline{d}_j$. If neither release dates nor deadlines are considered, (i.e. $r_j = 0$ and $\overline{d}_j = \infty$, $\forall j \in J$), the resulting problem $[R||C_{\max}]$ reduces to the decision of assigning jobs to processors. Sequencing jobs on the processors (i.e., the slave problem) becomes trivial as any no-wait schedule is optimal. It hence suffices to only solve the master model.

Regarding generalizations, our approaches allow us to easily include some application specific constraints. For example, we can consider so-called *processing set restrictions*, surveyed by Leung and Li (2008). According to this restriction, for each job j, there is a given set of incompatible processors $G(j) \subset P$ on which job j cannot be processed. For instance, in a berth allocation context, some ships may not be compatible with certain berths due to insufficient water depth (e.g., Tong et al., 1999). The corresponding problem is denoted as $[R|r_j, \overline{d}_j, M_j|C_{\max}]$. This restriction can be considered by adding constraints $y_{ij} = 0$, $\forall j \in J; i \in G(j)$, to [Master]. For the GSPP heuristic, the processing set constraint can be considered by generating sets of columns M_{ij} only for compatible machine and job pairs (i.e., $M_{ij} = \emptyset$, $\forall j \in J, i \in G(j)$).

Our approaches can also be easily extended to include *machine availability* restrictions, i.e., for the case when each processor i cannot process any job before its start time s_i or after its end time e_i. In the scheduling literature, the resulting problem is denoted as $[R, NC|r_j, \overline{d}_j|C_{\max}]$ (e.g., Sanlaville and Schmidt, 1998). Due to its practical relevance, many versions of existing discrete berth allocation problems consider berth availability times (e.g., Imai et al., 2001, Cordeau et al., 2005, Buhrkal et al., 2011). Especially,

consideration of machine start times enables using a rolling scheduling environment.

For B&BC, the machine availability times can be taken into account by modifying inequalities (E.4) and (E.5) as $\max\{r_j, s_i\} + \sum_{j' \in N(j)} y_{ij'} \cdot p_{ij'} \leq z$, $\forall i \in P; j \in J$, and $\max\{r_j, s_i\} + \sum_{j' \in N(j)} y_{ij'} \cdot p_{ij'} \leq \min\{\overline{d}_{\max}(j), e_i\}$, $\forall i \in P; j \in J : s_i \leq \overline{d}_{\max}(j) \wedge e_i \geq r_j$, respectively. This tightens the current time window of job j on machine i if the machine availability times are more limiting than the job time windows. Analogously, BDP for the slave problem on a given machine i can be adjusted by setting $\mathcal{C}(\emptyset) := s_i$ and tightening criteria 1 and 2 from Section E.3.1.2 as $\mathcal{C}(\Sigma) + p_{ij} \leq \min\{e_i; \overline{d}_j\}$, $\forall j \in \Gamma_i \backslash \Sigma$ and $\mathcal{C}(\Sigma) + \sum_{j \in \Gamma_i \backslash \Sigma} p_{ij} \leq \min\{e_i, \max_{j \in \Gamma_i \backslash \Sigma}\{\overline{d}_j\}\}$, respectively.

For the GSPP heuristic, the set M of all columns has to contain only such columns that satisfy both job time window and machine availability-related restrictions (i.e., $M = \{(i, j, t) \in P \times J \times \{1, \ldots, T\} \mid \max\{r_j; s_i\} \leq t \leq \min\{\overline{d}_j, e_i\} - p_{ij}\}$).

We can further extend our problem by considering so called *tails* (or delivery times) when, for each job $j \in J$, a nonzero delivery time q_j is given. Tail q_j is defined as the amount of time which job j has to spend in the system after completion on a machine. During the delivery step, job j does not occupy any machine but still affects the makespan, i.e., objective function (E.1) becomes $C(S) = \max_{(i,j,t) \in S}\{t + p_{ij} + q_j\}$.

To modify B&BC for this extension, we add the smallest delivery time $q_{min} = \min_{j \in J}\{q_j\}$ to the left-hand sides of inequalities (E.4) and (E.5) of [Master] to tighten the valid inequality. Moreover, BDP for the slave problem is adjusted as follows. We extend the state space by considering states $(\Sigma, \mathcal{C}(\Sigma))$, i.e., a state is defined by the set Σ of jobs already scheduled and the makespan $\mathcal{C}(\Sigma)$, when the last job leaves the machine. Each state $(\Sigma, \mathcal{C}(\Sigma))$ has successor states $(\Sigma \cup \{j\}, \max\{\mathcal{C}(\Sigma), r_j\} + p_{ij})$, $\forall j \in \Gamma_i \setminus \Sigma$. Let $\Xi(\Sigma, \mathcal{C}(\Sigma))$ be the set of states from which a transition to state $(\Sigma, \mathcal{C}(\Sigma))$ exists. We calculate the objective value of a state as

$$\mathcal{C}^{\text{tails}}(\Sigma, \mathcal{C}(\Sigma)) = \min_{(\Sigma', \mathcal{C}(\Sigma)') \in \Xi(\Sigma, \mathcal{C}(\Sigma))} \left\{ \max\left\{ \mathcal{C}^{\text{tails}}(\Sigma', \mathcal{C}(\Sigma)'); \max\left\{ \mathcal{C}(\Sigma)', r_j \right\} + p_{ij} + q_j \right\} \right\},$$

which is the completion time, including the tails, of the (partial) schedule (note that $j = \Sigma \backslash \Sigma'$). Conditions 1 through 3 are similarly adapted to account for tails. Note that if the deadlines are relaxed (i.e., $\overline{d}_j = \infty$, $\forall j \in J$), then the slave problem $[1|r_j, q_j|C_{\max}]$

can be solved by the procedure proposed by Carlier (1982).

To modify our GSPP heuristic, we substitute the objective function of the GSPP model (Eq. (E.7)) with objective function minimize $G(V) = \max_{(i,j,t)=m \in M}\{v_m \cdot (t + p_{ij} + q_j)\}$. Moreover, we only consider columns that satisfy the deadlines, i.e., $M = \{(i,j,t) \in P \times J \times \{1,\ldots,T\} \mid r_j \leq t \leq \overline{d}_j - p_{ij} - q_j\}$.

E.4.2. Alternative minmax objectives

Apart from changing the constraint set, our approaches can also be used for different minmax objectives. Representatively, we discuss two alternative performance criteria, namely the maximum (weighted) flow time and the maximum (weighted) lateness, in the following.

E.4.2.1. Minimizing the weighted maximum flow time

To adjust [Master] for the case when the maximum weighted flow time is to be minimized, we define sets $N_k(j) \subseteq N(j)$, containing the k jobs with the earliest release dates from $N(j)$ ($k \in \{1,\ldots,|N(j)|\}$). We replace valid inequalities (E.4) of [Master] with (E.4.2.1)

$$\min_{j' \in N_k(j)}\{w_{j'}\} \cdot \left(r_j + \sum_{j' \in N_k(j)} y_{ij'} \cdot p_{ij'} - \max_{j' \in N_k(j)}\{r_{j'}\} \right) \leq z \qquad \forall i \in P; j \in J; k \in \{1,\ldots,|N(j)|\} \tag{E.11}$$

Valid inequalities (E.4.2.1) set the value for the lower bound z on the maximum flow time by relaxing deadlines and weights. If the deadlines and weights of the jobs are not considered, an ERD ordering of jobs is optimal, which is what underlies inequalities (E.4.2.1): If the k successors of some job j are processed in ERD order, they cannot finish processing sooner than $r_j + \sum_{j' \in N_k(j)} y_{ij'} \cdot p_{ij'}$. The latest release date of these jobs is $\max_{j' \in N_k(j)}\{r_{j'}\}$; hence, the maximum flow time on processor i cannot be less than the left side of inequality (E.4.2.1).

The corresponding slave problem $[1|r_j, \overline{d}_j|wF_{\max}]$ for processor i and a given set Γ_i can be solved by adapting our BDP. A state is characterized by tuple $(\Sigma, \mathcal{C}(\Sigma))$, where Σ is the set of completed jobs and $\mathcal{C}(\Sigma)$ is the completion time of the state. Starting from

dummy state $(\emptyset, 0)$, there are transitions from some state $(\Sigma', \mathcal{C}(\Sigma)')$ to successor states $(\Sigma, \mathcal{C}(\Sigma))$, such that $\Sigma = \Sigma' \cup \{j\}$ and $\mathcal{C}(\Sigma) = \max\{\mathcal{C}(\Sigma)', r_j\} + p_{ij}$, $\forall j \in \Gamma_i \backslash \Sigma'$.

Let $\Xi(\Sigma, \mathcal{C}(\Sigma))$ be the set of all states from which a transition to $(\Sigma, \mathcal{C}(\Sigma))$ exists. The (partial) objective value of a state $(\Sigma, \mathcal{C}(\Sigma))$ is

$$\mathcal{F}_{\max}(\Sigma, \mathcal{C}(\Sigma)) = \min_{(\Sigma', \mathcal{C}(\Sigma)') \in \Xi(\Sigma, \mathcal{C}(\Sigma))} \left\{ \max \left\{ \mathcal{F}_{\max}(\Sigma', \mathcal{C}(\Sigma)'); w_j \cdot (\mathcal{C}(\Sigma) - r_j) \right\} \right\},$$

where $j = \Sigma \backslash \Sigma'$, and the objective value of the initial state is $\mathcal{F}_{\max}(\emptyset, 0) := 0$.

Moreover, the calculation of the lower bound of state $(\Sigma, \mathcal{C}(\Sigma))$ is modified as follows:

$$LB(\Sigma, \mathcal{C}(\Sigma)) = \max \left\{ \mathcal{F}_{\max}(\Sigma, \mathcal{C}(\Sigma)); \max_{j \in \Gamma_i \backslash \Sigma; k \in \{1, \ldots, |N(j)|\}} \left\{ \min_{j' \in N_k(j) \cap (\Gamma_i \backslash \Sigma)} \{w_{j'}\} \cdot \left(\max\{\mathcal{C}(\Sigma); r_j\} + \sum_{j' \in N_k(j) \cap (\Gamma_i \backslash \Sigma)} p_{ij'} - \max_{j' \in N_k(j) \cap (\Gamma_i \backslash \Sigma)} \{r_{j'}\} \right) \right\} \right\}.$$

We say that a state $(\Sigma', \mathcal{C}(\Sigma)')$ is dominated by a different state $(\Sigma, \mathcal{C}(\Sigma))$ if the following criteria are satisfied:

- $\Sigma = \Sigma'$, i.e., both states consider the same set of completed jobs,
- $\mathcal{C}(\Sigma) \leq \mathcal{C}(\Sigma)'$, i.e., the makespan of $(\Sigma, \mathcal{C}(\Sigma))$ is not longer than that of $(\Sigma', \mathcal{C}(\Sigma)')$, and
- $\mathcal{F}_{\max}(\Sigma, \mathcal{C}(\Sigma)) \leq \mathcal{F}_{\max}(\Sigma', \mathcal{C}(\Sigma)')$, i.e., the weighted maximum flow time of the dominating state is at least as good as that of the dominated state.

Dominated states do not have to be considered for the next stage of BDP since they will never lead to any improvement. Otherwise, the BDP can run as described in Section E.3.1.2.

Example (cont.): Consider the example from Section E.3.1.2, where a slave problem for processor $i = 2$ with $\Gamma_2 = \{1, 2, 4\}$ is to be solved. Additionally, the following weights for jobs 1, 2, and 4 are given: $w_1 = 1$, $w_2 = 2$, $w_4 = 1$. The resulting DP graph for minimizing the weighted maximum flow time on machine 2 is shown in Figure E.3. The optimal solution is bold, corresponding to job sequence $\langle 1, 2, 4]$.

To adjust our GSPP heuristic, only objective function (E.7) of the GSPP model has to

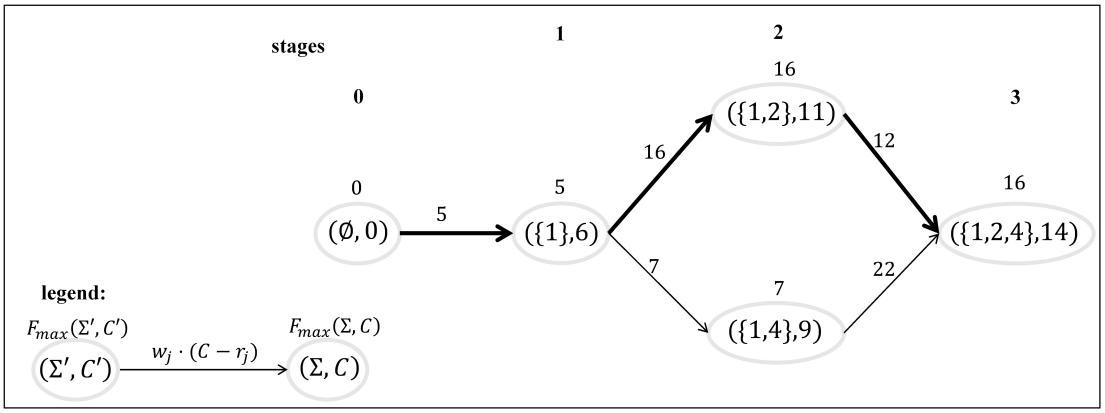


Figure E.3.: Dynamic programming graph for the slave problem in the example minimizing $wF_{\max}$.

be substituted by (E.12).

$$\text{Minimize } G(V) = \max_{(i,j,t)=m \in M} \{w_j \cdot v_m \cdot (t + p_{ij} - r_j)\} \tag{E.12}$$

E.4.2.2. Minimizing maximum weighted lateness

Similar considerations as in the previous section can be applied for minimization of the *maximum weighted lateness* $wL_{\max}$.

To modify model [Master] for $[R|r_j|wL_{\max}]$, for each job j, we define the set $V(j) = \{j' \in J | d_{j'} \leq d_j\}$ of jobs whose due dates are not later than the due date of job j. Further, we define sets $V_k(j) \subseteq V(j)$, containing the k jobs with the latest due dates from $V(j)$ ($k \in \{1, \ldots, |V(j)|\}$). This way we can replace valid inequalities (E.4) of [Master] with (E.4.2.2)

$$\min_{j' \in V_k(j)} \{w_{j'}\} \cdot \left(\min_{j' \in V_k(j)} \{r_{j'}\} + \sum_{j' \in V_k(j)} y_{ij'} \cdot p_{ij'} - d_j \right) \leq z \qquad \forall i \in P; j \in J; k \in \{1, \ldots, |V(j)|\}, \tag{E.13}$$

which determines the lower bound on the weighted maximum lateness. The idea behind inequalities (E.4.2.2) is to relax the weights and the release dates, so that the relaxed slave problem per processor becomes $[1||L_{\max}]$ which can be easily solved to optimality by the EDD rule (see Section E.4.1). Thus, the optimal objective value of the relaxed

slave problem $[1||L_{\max}]$, weighted with the smallest weight $\min_{j' \in V_k(j)}\{w_{j'}\}$, is a valid lower bound for the objective function of $[1|r_j|wL_{\max}]$ which is what inequalities (E.4.2.2) mimic.

To solve the corresponding slave problem $[1|r_j|wL_{\max}]$, BDP has to be adapted similarly as for $wF_{\max}$ objective with the following differences:

- The (partial) objective value of a state $(\Sigma, \mathcal{C}(\Sigma))$ is calculated as

$$\mathcal{L}_{\max}(\Sigma, \mathcal{C}(\Sigma)) = \min_{(\Sigma', \mathcal{C}(\Sigma)') \in \Xi(\Sigma, \mathcal{C}(\Sigma))} \{\max\{\mathcal{L}_{\max}(\Sigma', \mathcal{C}(\Sigma)'); w_j \cdot (\mathcal{C}(\Sigma) - d_j)\}\},$$

  where $j = \Sigma \backslash \Sigma'$.

- The lower bound of state $(\Sigma, \mathcal{C}(\Sigma))$ is computed as follows:

$$LB(\Sigma, \mathcal{C}(\Sigma)) = \max\left\{\mathcal{L}_{\max}(\Sigma, \mathcal{C}(\Sigma)); \max_{j \in \Gamma_i \backslash \Sigma; k \in \{1,\ldots,|V(j)|\}} \left\{\min_{j' \in V_k(j) \cap (\Gamma_i \backslash \Sigma)} \{w_{j'}\} \cdot \left(\max\left\{\mathcal{C}(\Sigma); \min_{j' \in V_k(j) \cap (\Gamma_i \backslash \Sigma)} \{r_{j'}\}\right\} + \sum_{j' \in V_k(j) \cap (\Gamma_i \backslash \Sigma)} p_{ij'} - d_j\right)\right\}\right\},$$

- While generating the states for BDP, criteria 1 and 2 from Section E.3.1.2 have to be ignored because the deadlines are no longer strict.

Example (cont.): The dynamic programming graph for the example slave problem from Section E.3.1.2, minimizing the maximum weighted lateness is depicted in Figure E.4.

Note that if the problem does not consider any weights, then the corresponding slave problem $[1|r_j|L_{\max}]$ is a textbook single-machine scheduling problem and can be solved by an exact algorithm from the literature; for an overview, we refer to Leung (2004, Chap. 10) and Pinedo (2016, Chap. 3.2).

To adjust the GSPP model for the new objective, only the objective function (E.7) has to be substituted by new objective function

$$\text{Minimize } G(V) = \max_{(i,j,t)=m \in M} \{w_j \cdot v_m \cdot (t + p_{ij} - d_j)\}.$$

However, note that GSPP formulation with lateness objective can only be applied if the

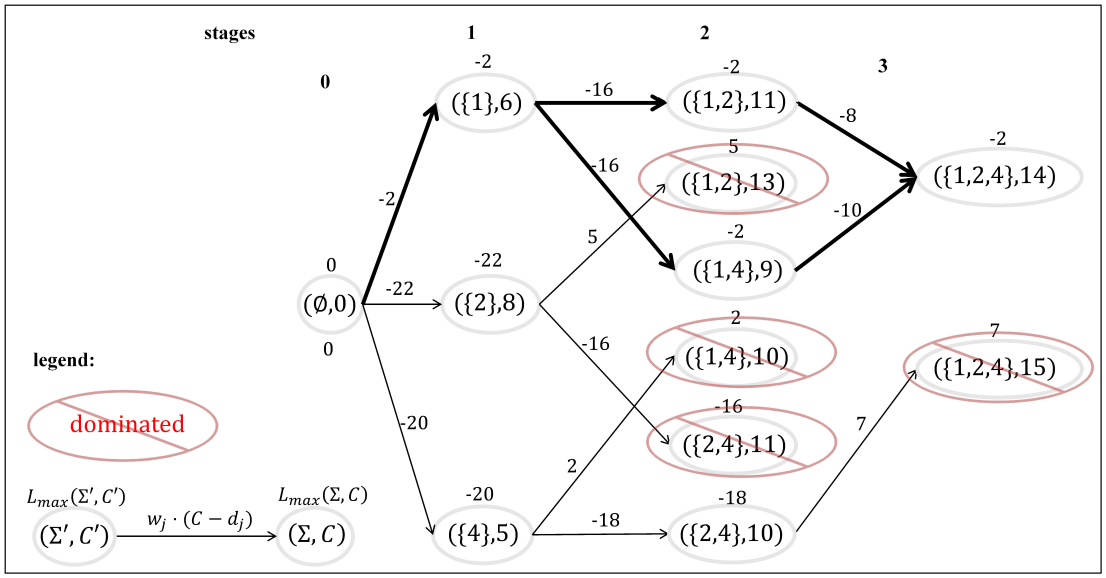


Figure E.4.: Dynamic programming graph for the slave problem in the example minimizing $L_{\max}$.

planning horizon is bounded from above by another restriction, e.g. end availability time of the machines or an upper bound on the schedule length.

E.5. Computational study

In this section, we compare the computational performance of the three presented solution approaches, namely a default solver solving the original mixed integer program (MIP-OP), branch and Benders cut (B&BC), and heuristic column selection based on a generalized set partitioning formulation (GSPP). A mixed-integer linear programming model [MIP-OP] for $[R|r_j, \overline{d}_j|C_{\max}]$, based on the model formulation for the classic discrete berth allocation problem proposed by Imai et al. (2001) and Monaco and Sammarra (2007), is presented in Appendix E.7. We use it as a benchmark. Experiments are performed on newly generated random problem instances as well as the benchmark instances from the literature for both applications (i.e., berth allocation and truck scheduling). We first describe how new problem instances have been generated and then describe the instances from the literature. Afterwards, we present the computational results of our solution approaches on both instance sets.

E.5.1. Benchmark instances and computational environment

To observe the computational performance of our algorithms, we test them on different kinds of problem instances. First, we generate new random instances for the basic problem $[R|r_j, \overline{d}_j|C_{\max}]$ according to the instance generation scheme proposed by Hall and Posner (2001). Specifically, the processing time p_{ij} of each job $j \in J$ on each machine $i \in P$ is randomly drawn from the following normal distribution: $p_{ij} \sim N(30, 6)$ (truncated below to positive lower bound). The release dates r_j are generated with the following scheme: $r_1 = 0$ and $r_j = r_{j-1} + X_j$, $\forall j \in 2, \ldots, n$, where $X_j \sim exp(\lambda)$ is an exponentially distributed random number with mean $\lambda = \frac{30}{|P|}$. This way, the jobs' arrival represents a Poisson process whose rate matches the total processors' job capacity. Finally, we draw the deadline of job j with the following equation: $\overline{d}_j = r_j + k \cdot E[p_{ij}]$, where $E[p_{ij}] = \sum_{i \in P} \frac{p_{ij}}{|P|}$ and $k \geq 1$. Thus, by varying parameter k, we receive instances that either have tight time windows (low k) or have more flexibility (high k). All time related parameters are rounded to the next integer value. Note that this instance generation scheme does not prevent infeasible instances from being generated. However, instances with a higher value of k are more likely to have feasible solutions.

To observe the impact of the problem size on the computational performance of our algorithms we generate different sized sets of problem instances. Specifically, we generate small instances consisting of $n = 20$ jobs and $m \in \{4, 6, 8\}$ machines (dubbed S) and larger instances with $n = 80$ jobs and $m \in \{15, 20, 25\}$ machines (dubbed L). Moreover, we vary parameter $k \in \{2, 3, 4\}$. Apart from this, for the parameter tuning tests we use medium-sized instances with $n = 50$ jobs, $m = 10$ machines and $k = 2.5$ (dubbed M).

Apart from our new problem instances, we test the performance of our approaches on extant problem instances from the literature for both aforementioned applications: berth allocation and truck scheduling.

In the context of berth allocation, we use the problem instances from sets "$I2$" and "$I3$" originally proposed by Cordeau et al. (2005). These instances were generated based on statistical traffic and berth allocation data at the Port of Gioia Tauro. $I2$ contains 50 instances with the following five instance sizes: 25 ships (i.e., jobs) with 5, 7 and 10 berths (i.e., processors); 35 ships with 7 and 10 berths. $I3$ consists of 30 larger instances with 60 ships and 13 berths.

As benchmark instances from the truck scheduling context, we use the problem in-

stances generated by Tadumadze et al. (2019) that can be downloaded using the following DOI: `https://doi.org/10.5281/zenodo.1487845`. Specifically, we use the problem instances from the set "ITWS-DC-M" that contains 30 larger problem instances with 100 trucks (i.e., jobs) and 25 dock doors (i.e., processors). Furthermore, these instances vary in the relative width of time windows of the trucks, which is controlled by the parameter $\Omega_{\max} \in \{1, 2, 3\}$, where instances with lower $\Omega_{\max}$ tend to have tighter time windows (similar to parameter k described above). Table E.2 summarizes all the instances used.

Proposed by	Dataset name	# Instances	n	m	Additional characteristics
This paper (based on Hall and Posner (2001))	S	90	$\{20\}$	$\{4, 5, 6\}$	$k \in \{2, 3, 4\}$
	M	10	$\{50\}$	$\{10\}$	$k = 2.5$
	L	90	$\{80\}$	$\{15, 20, 25\}$	$k \in \{2, 3, 4\}$
Cordeau et al. (2005)	I2	50	$\{25, 35\}$	$\{5, 7, 10\}$	-
	I3	30	$\{60\}$	$\{13\}$	-
Tadumadze et al. (2019)	ITWS-DC-M	30	$\{100\}$	$\{25\}$	$\Omega_{\max} \in \{1, 2, 3\}$

Table E.2.: Summary of benchmark instance sets

Note that there is, to the best of our knowledge, no dataset in the literature for $[R|r_j, \overline{d}_j|\ C_{max}]$ and its variants discussed in this paper. The literature datasets of Cordeau et al. (2005) and Tadumadze et al. (2019) have a feasible solution space that is covered by our approaches, but the original papers consider a different objective function—minimization of total (weighted) flow time $\sum w_j F_j$, which differs from all minmax objectives considered in this work. The numerical results can therefore not be directly compared. However, the performance of our algorithms on these datasets should nonetheless give a good idea of the applicability of our approaches.

We have implemented our solution procedures in C# 6.0 and applied off-the-shelf solver IBM ILOG CPLEX Optimizer V12.8.0 for solving our MILP models, including the master model of our B&BC approach. All tests have been executed on an x64 PC with an Intel Core i7-8700K 3.70 GHz CPU and 64 GB RAM. The time limit for solving the MILP models is set to 300 CPU seconds. Note that we also experimented with larger time limits in preliminary tests but found that the best upper bounds are usually found within the first five minutes of computation. Moreover, to make a fair comparison between alternative solution approaches, we execute all solution methods (including the default solver) on a single thread. The problem instances as well as the detailed computational results for every instance are available from the following DOI: `https://doi.org/10.5281/zenodo.3696775`.

E.5.2. Parameter tuning

We calibrate both of the proposed approaches in a pretest. Specifically, in the case of B&BC, we compare two ways of adding cuts to the master model: either as lazy constraints, injecting them into the branch & bound tree as it grows, or as regular constraints, such that the master model is resolved from scratch every time a new cut is added. Similarly, the performance of our heuristic column selection approach strongly depends on the value of the parameter μ. A low value of μ leads to a high number of generated columns which, on the one hand, can be beneficial for smaller size problem instances. On the other hand, it can be counterproductive for larger instances, where solving the resulting large GSPP model within a given time limit may lead to poor solutions. The proper value of μ clearly depends on the instance size; thus, we derive the value of μ from the number of jobs $|J|$ with the equation $\mu = \lceil |J| \cdot \delta \rceil$ and the predefined parameter δ. Note that a high value of δ leads to compact GSPP models with a smaller search space.

For our pretest, we use 10 medium-sized instances with $|J| = 50$, $|P| = 10$, and $k = 2.5$. We solve them with regard to makespan as an objective function applying both versions of B&BC (i.e., applying either lazy or regular cuts) and the heuristic GSPP approach, applying five different values of $\delta \in \{0.02, 0.1, 0.2, 0.3, 0.4\}$.

Table E.3 shows the results for the parameter tuning test with regard to $C_{\max}$ (i.e., makespan) as an objective function. Column "$C^*_{\max}$" denotes the best known makespan for that instance, found by B&BC, either applying lazy or regular cuts, where an asterisk (*) after the value indicates that this objective value is optimal. For each approach (i.e., B&BC with both kinds of cuts and GSPP heuristics for each value of δ), we report the average relative gap (in %) of the objective value val$'$ found by that approach to the best known objective value for that instance val*, calculated as gap $= \frac{\text{val}' - \text{val}^*}{\text{val}^*} \cdot 100$ and the required computational time (in CPU seconds).

First of all, the results tend to be better when adding cuts to [Master] as lazy constraints. For each of the 10 instances, B&BC with lazy cuts obtains an optimal solution within the given time limit. On the other hand, B&BC with regular cuts struggles to solve 2 out of 10 instances to proven optimality within 300 seconds. This is likely due to the computation time that is expended on solving the master model from scratch in every iteration. This leads to the solver wasting a lot of time re-exploring regions of the search space it already explored in previous iterations. Note, however, that regular cuts are

		B&BC		GSPP				
		Lazy cuts	Regular cuts	$\delta = 0.02$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.4$
Instance	$C^*_{\max}$	Gap/time	Gap/time	Gap/time	Gap/time	Gap/time	Gap/time	Gap/time
50x10-01	175*	0.0/0.6	0.0/0.1	9.1/300.8	0.0/11.4	0.0/1.9	0.0/0.2	0.0/0.2
50x10-02	230*	0.0/0.0	0.0/0.1	2.6/300.8	0.0/6.2	0.0/3.9	0.0/0.2	0.0/0.1
50x10-03	153*	0.0/0.5	0.0/22.7	23.5/300.7	1.3/150.5	3.3/6.7	4.6/4.4	7.8/2.7
50x10-04	201*	0.0/0.4	0.0/0.1	0.0/250.7	0.0/16.4	0.0/4.1	0.0/0.1	0.0/0.1
50x10-05	189*	0.0/1.3	0.0/300.0	-/300.7	1.1/52.2	3.2/8.1	2.7/2.4	3.2/2.0
50x10-06	153*	0.0/0.1	0.0/0.1	-/300.7	0.0/8.0	0.0/0.4	0.7/2.3	3.3/1.5
50x10-07	163*	0.0/1.0	0.0/141.4	-/300.7	1.2/15.4	1.8/6.8	1.8/6.3	3.1/2.4
50x10-08	170*	0.0/114.0	1.8/300.0	-/300.7	1.2/46.4	3.5/5.1	4.7/5.0	3.5/2.1
50x10-09	200*	0.0/0.0	0.0/0.1	10.0/300.8	0.0/19.0	0.0/0.4	0.0/0.2	0.0/0.2
50x10-10	176*	0.0/0.2	0.0/1.8	26.7/300.7	0.0/128.7	0.0/0.3	2.8/1.8	2.8/0.8
Mean		0.0/11.8	0.2/76.6	12.0/295.7	0.5/45.4	1.2/3.8	1.7/2.3	2.4/1.2

* indicates that the value is optimal

- indicates that in the time limit no feasible solutionhas been found

Table E.3.: Calibration of B&BC (type of cuts) and GSPP heuristic (value of δ).

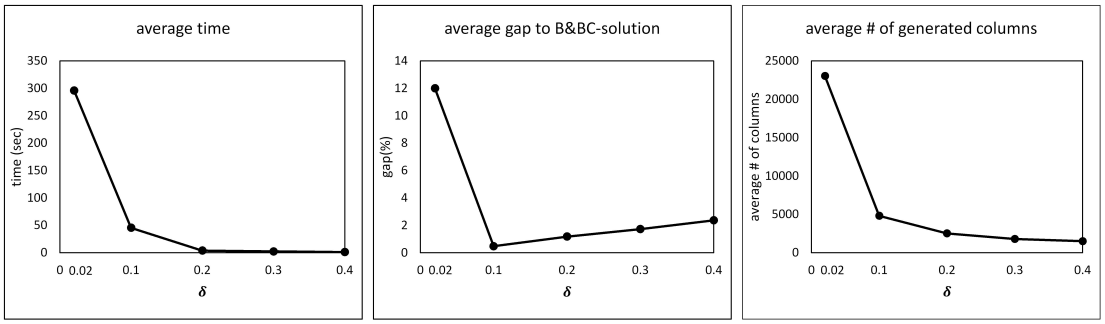


Figure E.5.: Influence of δ on the solution quality of the GSPP heuristic.

sometimes marginally superior to lazy constraints, probably because the presolver is more aggressive in the absence of lazy constraints and the root node relaxation is stronger in later iterations, when multiple cuts have already been added. Nonetheless, in light of the performance stability of lazy cuts, adding constraints lazily seems advisable.

Figure E.5 summarizes the average results in terms of runtime and relative gap to the optimal objective value (obtained by B&BC) for each value of parameter δ used to select columns for the GSPP heuristic. The leftmost and rightmost graphics of Figure E.5 present an apparently exponential decrease in runtime and number of generated columns when the value of δ rises. For $\delta = 0.02$, only one out of 10 instances could be solved in less than 300 seconds (note that $\delta = 0.02$ leads to $\mu = 1$, and the resulting GSPP, containing all columns, is equivalent to the original problem). For higher values of δ, all resulting GSPP model instances could be solved within the time limit. However, raising the value of δ makes it less probable that the GSPP contains potentially good columns, possibly lowering the solution quality. From our results, $\delta = 0.1$ turns out to be the most promising compromise of solution quality and runtime. For $\delta = 0.1$, in 6 out of 10 cases, GSPP obtains the optimal solution for the corresponding original problem while maintaining acceptable computational times. Also, for the remaining 4 instances, GSPP finds near-optimal solutions (with a maximal optimality gap of less than 1.4%) in reasonable computational times. Therefore, for the next computational experiments we apply $\delta = 0.1$ when using our GSPP heuristic and we add cuts as lazy constraints to the master model of B&BC. Note that setting $\delta = 0.1$ does not guarantee that a feasible solution is in the GSPP search space. If no feasible solution can be found, δ must be lowered to expand the number of columns in the model. However, in our computational experiments, this proves to be unnecessary; we are able to solve all instances to feasibility with $\delta = 0.1$.

E.5.3. Computational performance on new instances

In this section, we compare the computational performance of the three proposed approaches (B&BC, GSPP and MIP-OP) solving $[R|r_j, \overline{d}_j|C_{\max}]$. For this, we use the problem instances from two different sized datasets and solve them with all three solution approaches. The first dataset S contains the problem instances with $n = 20$ jobs and $m \in \{4, 6, 8\}$ processors while the second dataset L consists of larger instances considering $n = 80$ jobs and $m = \{15, 20, 25\}$ machines. Further, we observe the impact of the time window tightness on the performance of our approaches by varying parameter $k \in \{2, 3, 4\}$. For each parameter constellation, we have 10 random instances, so that in total 180 new problem instances have been generated.

Table E.4 summarizes the aggregated computational results for the newly generated problem instances. The first two columns describe the instance characteristics: the number of jobs n and machines m and the value of parameter k. For each approach, in columns "Gap (%)" and "Time (s)" we report the average relative gap (in %) of the best found objective value to the best known objective value of that instance, calculated as described in Section E.5.2, and the average computational runtime (in CPU seconds), averaged per 10 instances of the same size. Furthermore, for each approach we report the share of the instances whose corresponding (M)ILP models are solved to optimality within the given time limit. Note that in case of our GSPP heuristic, solving the corresponding ILP does not necessarily mean that the found solution is optimal. Therefore, for GSPP, we additionally report the share of instances whose objective value matches the best known objective value (column "Best"). Moreover, the average number of generated cuts for B&BC and columns for GSPP are reported in columns "Cuts" and "Columns", respectively. For the large instances from dataset L the solver runs out of memory even before generating the corresponding MIP-OP model. Hence, for those instances, we only report the computational performance of the B&BC and GSPP approaches.

The first remarkable result is that B&BC outperforms its competitors, solving in total 147 out of 180 instances to optimality within 300 seconds. Only for 5 large instances, the solution obtained by B&BC is improved upon by the GSPP heuristic, which found the best known solution for 108 out of 180 instances. It is noteworthy that the heuristic column selection approach tends to be more successful for problem instances with tight time windows (i.e., low k) and fewer processors, leading to fewer columns (i.e., potential starting times of jobs). The default solver, on the other hand, struggles with solving

Instance		B&BC				GSPP					MIP-OP		
Size	k	Opt	Gap (%)	Time (s)	Cuts	Best	Opt	Gap (%)	Time (s)	Columns	Opt	Gap (%)	Time (s)
Dataset S													
20x4	2	1	0	0.2	314.9	1.0	1.0	0.0	0.3	1265.3	0.3	0.8	267.9
20x4	3	1	0	0.1	33.6	0.9	1.0	0.0	5.4	2465.2	0.3	0.2	226.2
20x4	4	1	0	9.3	851.6	0.6	1.0	0.3	13.7	3689.6	0.3	0.6	225.3
20x6	2	0.9	0	30.3	256.4	0.8	1.0	0.2	0.6	1882.7	0.2	0.9	276.3
20x6	3	0.9	0	30.5	246.6	0.8	1.0	0.1	9.9	3691.8	0.1	0.1	277.4
20x6	4	0.9	0	30.1	918.7	0.7	1.0	0.2	7.8	5489.2	0.4	0.3	215.9
20x8	2	1	0	0.7	13.6	0.8	1.0	0.2	2.9	2524.8	0.4	0.4	232.6
20x8	3	1	0	2.1	14.6	0.9	1.0	0.2	10.3	4925.7	0.4	0.0	203.6
20x8	4	1	0	3.6	53.1	1.0	1.0	0.0	9.7	7319.2	0.4	0.2	209.5
mean (S)		*0.97*	*0.00*	*11.88*	*300.34*	*0.83*	*1.00*	*0.15*	*6.73*	*3694.83*	*0.31*	*0.39*	*237.18*
Dataset L													
80x15	2	0.5	0.6	184.9	6620.4	0.8	1.0	0.2	61.6	5122.2	-	-	-
80x15	3	0.8	0	76.7	31.4	0.2	0.3	8.5	269.5	9642.5	-	-	-
80x15	4	0.6	0	121.0	38.8	0.0	0.0	21.0	300.6	14188.2	-	-	-
80x20	2	0.6	0	122.4	316.3	0.8	1.0	0.2	111.1	6820.6	-	-	-
80x20	3	0.8	0	89.0	74.0	0.3	0.4	8.8	262.0	12877.1	-	-	-
80x20	4	0.8	0	71.3	33.8	0.2	0.2	22.3	270.7	18926.9	-	-	-
80x25	2	0.7	0.7	121.1	303.2	0.5	0.5	7.4	219.6	8524.9	-	-	-
80x25	3	0.6	0.2	121.6	39.1	0.6	0.6	5.3	227.8	16082.4	-	-	-
80x25	4	0.6	0	123.2	43.0	0.0	0.0	36.0	300.8	23684.4	-	-	-
mean (L)		*0.67*	*0.17*	*114.58*	*833.33*	*0.38*	*0.44*	*12.18*	*224.86*	*12874.36*	-	-	-

Each entry contains averaged results over 10 instances with the same parameter constellation

Table E.4.: Comparison between B&BC, GSPP and MIP-OP on newly generated instances.

instances to proven optimality within the time limit using model [MIP-OP]. Only 28 out of 90 small instances from dataset S can be solved to optimality within a time limit and CPLEX runs out of memory even before generating the corresponding MIP-OP models for large instances from dataset L.

E.5.4. Benchmark tests on berth allocation problem instances from the literature

To observe the performance of our approaches on berth allocation problem instances, we use datasets $I2$ and $I3$ originally proposed by Cordeau et al. (2005), containing in total 80 realistic BAP instances.

Note that Cordeau et al. (2005), apart from the time window restrictions for each vessel, additionally consider the availability times of berths (i.e., start s_i and end e_i of availability for each berth $i \in P$). Thus, to solve these instances with our approaches, we modify them to include the machine availability time restrictions as described in Section E.4.1. Moreover, since minimizing the weighted flow time is a common objective in the berth allocation literature, apart from our baseline objective (i.e., makespan), we solve the instances with regard to maximum weighted flow time. Finally, we solve the instances with regard to maximum weighted lateness, which is considered an appropriate objective in many berth allocation scenarios (e.g., Liu et al., 2006, Chen et al., 2012).

Tables E.5, E.6 and E.7 summarize the computational results of our comparison on the BAP instances with regard to objective $C_{\max}$, $wF_{\max}$ and $wL_{\max}$, respectively. Each entry contains averaged results over all instances of the same size (i.e., for dataset $I2$, each entry contains results averaged over 10 instances and for dataset $I3$, over 30 instances). The columns of Tables E.5, E.6 and E.7 have the same meaning as in Table E.4 from Section E.5.3. Furthermore, we report the objective values of the best found solutions, averaged per parameters constellation (see columns "$C^*_{\max}$", "$wF^*_{\max}$" and "$wL^*_{\max}$" of Tables E.5, E.6 and E.7, respectively).

The results indicate that B&BC remains a successful solution approach for BAP with regard to all considered performance criteria. It solves in total 77, 78 and 78 out of 80 instances to proven optimality within 300 seconds for objectives $C_{\max}$, $wF_{\max}$ and $wL_{\max}$, respectively. Also, the GSPP heuristic performs quite well, providing optimal solutions for 70, 64 and 66 out of 80 instances for $C_{\max}$, $wF_{\max}$ and $wL_{\max}$, while the gaps for non-optimal solutions are almost always negligible. The default solver, on the

		B&BC				GSPP					MIP-OP		
Instance size	$C^*_{\max}$	Opt	Gap (%)	Time (s)	Cuts	Best	Opt	Gap (%)	Time (s)	Columns	Opt	Gap (%)	Time (s)
25x5	173.5	0.8	0.0	61.2	60.3	0.8	1.0	0.2	32.5	8,118.5	0.2	1.0	250.9
25x7	163.7	1.0	0.0	0.0	22.2	1.0	1.0	0.0	20.5	11,323.7	0.6	0.1	188.2
25x10	171.5	1.0	0.0	0.1	23.6	1.0	1.0	0.0	23.5	14,973.5	0.3	0.1	255.8
35x7	175.6	0.7	0.0	95.4	57.4	0.4	0.8	0.7	152.6	11,858.2	0.0	5.2	300.7
35x10	173.7	1.0	0.0	0.2	47.8	0.8	0.9	4.2	95.6	15,908.2	0.0	2.6	301.0
60x13	169.5	1.0	0.0	0.1	33.6	1.0	1.0	0.0	47.8	14,940.9	0.0	9.4	310.9

Each entry contains results averaged over all instances of the same size

Table E.5.: Comparison between B&BC, GSPP and MIP-OP on BAP minimizing makespan.

		B&BC				GSPP					MIP-OP		
Instance size	$wF^*_{\max}$	Opt	Gap (%)	Time (s)	Cuts	Best	Opt	Gap (%)	Time (s)	Columns	Opt	Gap (%)	Time (s)
25x5	58.2	1.0	0.0	5.6	47.3	0.2	1.0	2.5	23.1	8,118.5	0.0	4.5	300.2
25x7	48.9	1.0	0.0	0.3	24.1	1.0	1.0	0.0	24.0	11,323.7	0.0	4.3	300.2
25x10	51.3	1.0	0.0	0.3	20.3	1.0	1.0	0.0	16.3	14,973.5	0.0	1.0	300.3
35x7	57.1	0.9	0.0	55.5	50.5	0.4	1.0	1.4	61.7	11,858.2	0.0	14.8	300.7
35x10	61.1	0.9	0.0	31.5	53.7	0.8	1.0	0.4	72.2	15,908.2	0.0	19.6	301.0
60x13	43.1	1.0	0.0	0.5	11.6	1.0	1.0	0.0	40.7	14,940.9	0.0	219.5	310.7

Each entry contains results averaged over all instances of the same size

Table E.6.: Comparison between B&BC, GSPP and MIP-OP on BAP minimizing maximum weighted flow time.

		B&BC				GSPP					MIP-OP		
Instance size	$wL^*_{\max}$	Opt	Gap (%)	Time (s)	Cuts	Best	Opt	Gap (%)	Time (s)	Columns	Opt	Gap (%)	Time (s)
25x5	-241.8	1.0	0.0	6.0	44.1	0.2	1.0	-0.6	27.7	8,118.5	0.0	-1.0	300.0
25x7	-251.1	1.0	0.0	0.9	26.9	1.0	1.0	0.0	14.7	11,323.7	0.0	-0.1	300.0
25x10	-248.7	1.0	0.0	1.0	81.1	1.0	1.0	0.0	15.8	14,973.5	0.1	-0.2	271.4
35x7	-242.9	0.9	0.0	60.9	46.3	0.4	1.0	-0.3	75.1	11,858.2	0.0	-2.1	300.1
35x10	-238.9	0.9	0.0	32.6	52.1	1.0	1.0	0.0	37.3	15,908.2	0.0	-2.3	300.1
60x13	-256.9	1.0	0.0	0.9	11.0	1.0	1.0	0.0	23.2	14,940.9	0.0	-9.2	300.6

Each entry contains results averaged over all instances of the same size

Table E.7.: Comparison between B&BC, GSPP and MIP-OP on BAP minimizing maximum weighted lateness.

other hand, struggles with solving instances to proven optimality within the time limit using the MIP-OP formulation. CPLEX manages to solve only 11 out of 80 instances within the time limit while minimizing $C_{\max}$. The effort of MIP-OP rises for the alternative objectives (i.e., minimization $wF_{\max}$ and $wL_{\max}$). CPLEX only solves one out of 80 instances within the time limit when $wL_{\max}$ is minimized, and for $wL_{\max}$, none of 80 instances are solved within 300 CPU seconds.

E.5.5. Benchmark tests on truck scheduling problem instances from the literature

Our benchmark set of problem instances from the truck scheduling context is originally proposed by Tadumadze et al. (2019) for what they call the "integrated truck and workforce scheduling problem" (ITWS). The set contains 30 very large problem instances considering 100 trucks and 25 doors. These instances are characterized by tighter time windows than our newly generated instances, which are controlled by the parameter $\Omega_{\max}$. More specifically, the expected relative width of the trucks' time windows for an ITWS instance with $\Omega_{\max} = 1$ is comparable with the expected relative width of the time windows of an problem instance generated by our instance generator using $k = 0.5$. However, the instances of Tadumadze et al. (2019) are generated in such way that each problem instance has at least one feasible solution with regard to the trucks' time windows. Note that the formal definition of the ITWS for the distribution center context (ITWS-DC) from Tadumadze et al. (2019) differs from our problem as apart from truck scheduling, it simultaneously solves the workforce scheduling problem. The problem of scheduling jobs with time windows on unrelated machines is equivalent to the case when ITWS-DC is solved for a given dock-specific workforce assignment as described in the original paper.

Tables E.8 and E.9 present the computational results of our approaches on truck scheduling instances. These instances are too large to be solved by a default solver. MIP-OP runs out of the memory even before generating the corresponding MILP model. Thus, we only solve each instance with B&BC and GSPP heuristics. Moreover, not every instance could be solved to feasibility, which is why we additionally report the share of instances that could be solved to feasibility within the time limit (see column "Feas").

As can be seen from the results, B&BC tends to be less successful at solving the large truck scheduling instances with tighter time windows from Tadumadze et al. (2019).

Instance			B&BC					GSPP					
Size	$\Omega_{\max}$	$C^*_{\max}$	Feas	Opt	Gap (%)	Time (s)	Cuts	Feas	Best	Opt	Gap (%)	Time (s)	Columns
100x25	1	72.7	0.7	0.7	0.0	114.8	15897.0	1.0	1.0	1.0	0.0	7.8	5677.3
100x25	2	69.2	0.8	0.3	0.6	217.5	7806.5	1.0	0.6	0.8	2.5	131.5	8889.0
100x25	3	69.3	0.8	0.3	1.0	215.8	21499.4	1.0	0.8	0.8	4.3	113.9	10350.7
Each entry contains averaged results over all 10 instances with the same parameter constellation													

Table E.8.: Comparison between B&BC and GSPP on ITWS minimizing makespan.

Instance			B&BC					GSPP					
Size	$\Omega_{\max}$	$F^*_{\max}$	Feas	Opt	Gap (%)	Time (s)	Cuts	Feas	Best	Opt	Gap (%)	Time (s)	Columns
100x25	1	31.0	0.2	0.0	27.8	301.0	292.2	1.0	1.0	1.0	0.0	24.7	5677.3
100x25	2	37.4	0.8	0.0	20.3	300.9	131.8	1.0	0.7	0.6	8.7	167.8	8889.0
100x25	3	41.6	0.6	0.0	14.7	300.9	139.4	1.0	0.9	0.5	2.3	222.4	10350.7
Each entry contains averaged results over 10 instances with the same parameter constellation													

Table E.9.: Comparison between B&BC and GSPP on ITWS minimizing maximum flow time.

The effort of B&BC is especially high for problem instances with tight time windows (i.e., low $\Omega_{\max}$). Recall that [Master] is a relaxed version of the problem which ignores the time window-related constraints. If the time windows of the jobs are too tight, the relaxation is bound to be less tight. As a result, the number of cuts added to [Master] rises, which negatively affects the performance of B&BC. While minimizing the makespan, B&BC obtains a feasible solution for 23 out of 30 instances out of which 13 are optimal. The effort is greater when the maximum weighted flow time is minimized. Here, B&BC manages to find a feasible solution for 16 out of 30 instances, none of which is proven to be optimal. On the other hand, GSPP exhibits exactly the opposite behavior. This is due to there being fewer columns (i.e., fewer potential starting times of jobs) in problems with tighter time windows. GSPP manages to find a feasible solution for every instance, which in most cases matches or improves the solutions found by B&BC. Specifically, while minimizing the makespan, GSPP manages to match or improve solutions obtained by B&BC for 24 out of 30 instances in shorter computational times. For the alternative objective (i.e., $F_{\max}$), GSPP shows even better performance matching or improving the B&BC solutions for 26 out of 30 instances. This indicates that B&BC and GSPP complement each other.

E.6. Conclusion

In this paper, we propose a novel logic-based branch & Benders cut algorithm for a family of unrelated parallel machines scheduling problems with time windows and minmax objective. We also devise a heuristic column selection approach based on a generalized set partitioning problem formulation. We show that the proposed approaches can be easily modified for relevant generalizations. We compare our methods to an adaptation of an existing mixed integer linear program solved by a commercial solver. We test all algorithms on newly generated problem instances with varying time window width and realistic benchmark instances from the literature.

The exact B&BC algorithm performs quite well, solving most instances, both from the literature and generated ones, to optimality within a few minutes, clearly outperforming a default solver solving the non-decomposed MILP model. The only cases where B&BC exhibits non-negligible optimality gaps are very large problem instances with tight time windows. For those cases, our heuristic column selection method is faster and frequently delivers better results.

Future research could aim to adjust the proposed solution method for alternative performance criteria, like total weighted flow time and other minsum objectives. Moreover, alternative heuristic approaches can also be tested and compared to our GSPP approach.

Appendix

E.7. MILP formulations for $[R|r_j, \overline{d}_j|\cdot]$ and extensions

J	Set of jobs (index j)
P	Set of processors (index i)
K	Set of service positions (index k, $K = \{1, \ldots, n\}$)
$\mathcal{M}$	Big integer
p_{ij}	Processing time of job j on processor i
r_j	Release date of job j
$\overline{d}_j/d_j$	Deadline/due date of job j
x_{ijk}	Binary variable: 1, if job j is the kth job to be processed on processor i; 0, otherwise
X	set of x Variables: $X = \{x_{ijk} \mid i \in P, j \in J, k \in K\}$
u_{ik}	Continuous variable: time interval that processor i stays idle before executing the kth job and after executing the $(k-1)$th job
U	Set of u variables: $U = \{u_{ik} \mid i \in P, k \in K\}$

Table E.10.: Notation for the MILP model.

This appendix contains a mixed-integer linear programming model for $[R|r_j, \overline{d}_j|C_{\max}]$. The model is based on the original discrete berth allocation model proposed by Imai et al. (2001) and Monaco and Sammarra (2007), which is modified to account for minmax objectives. Table E.10 summarizes the notation.

$$\text{[MIP-OP] Minimize } C(X,U) = \max_{i \in P} \left\{ \sum_{k \in K} \left(u_{ik} + \sum_{j \in J} p_{ij} \cdot x_{ijk} \right) \right\} \tag{E.14}$$

subject to

$$\sum_{i \in P} \sum_{k \in K} x_{ijk} = 1 \qquad \forall j \in J \tag{E.15}$$

$$\sum_{j \in J} x_{ijk} \leq 1 \qquad \forall i \in P, k \in K \tag{E.16}$$

$$\sum_{j \in J} r_j \cdot x_{ijk} - \sum_{\substack{l \in K: \\ l < k}} \left(u_{il} + \sum_{j \in J} p_{ij} \cdot x_{ijl} \right) \leq u_{ik} \qquad \forall i \in P, k \in K \tag{E.17}$$

$$\sum_{\substack{l \in K: \\ l \leq k}} \left(u_{il} + \sum_{j' \in J} p_{ij'} \cdot x_{ij'l} \right) \leq \overline{d}_j + \mathcal{M} \cdot (1 - x_{ijk}) \quad \forall i \in P, j \in J, k \in K \tag{E.18}$$

$$u_{ik} \geq 0 \qquad \forall i \in P, k \in K \tag{E.19}$$

$$x_{ijk} \in \{0; 1\} \qquad \forall j \in J, i \in P, k \in K \tag{E.20}$$

Objective function (E.14) minimizes the makespan of the machine whose completion time after processing all jobs is highest. The makespan for each machine is derived by adding up the total processing time of the jobs and the total idle time between jobs. Constraints (E.15) ensure that every job is assigned to exactly one processor at exactly one service position. Constraints (E.16) make it impossible for multiple jobs to be assigned the same service position on the same machine. Inequalities (E.17) define the idle times between jobs: idle time u_{ik} (i.e., the amount of time that machine i stays idle before executing the kth job) must be no less than the duration between the completion time of the $(k-1)$th job on machine i and the release date of the kth job on the same machine (i.e., the earliest possible processing start time of its successive job). Note that the summation on the left-hand side of constraints (E.17) and (E.18) indicates the completion time of the $(k-1)$th job on machine i. Constraints (E.18) make sure that the deadlines are not violated. Finally, (E.19) and (E.20) define the domain of the variables.

To extend MIP-OP to account for machine availability restrictions, constraints (E.21)

and (E.22) are added to the model.

$$u_{i1} \geq s_i \qquad \forall i \in P \tag{E.21}$$

$$\sum_{k \in K} \left(u_{ik} + \sum_{j \in J} p_{ij} \cdot x_{ijk} \right) \leq e_i \qquad \forall i \in P \tag{E.22}$$

Constraints (E.21) guarantee that on each machine $i \in P$, processing of its first job cannot be started before its availability time s_i. Constraints (E.22) take care that on each machine i none of the jobs are completed after its availability time e_i.

Finally, to consider objectives maximum weighted flow time and lateness, we consider the following models.

To minimize the maximum weighted flow time,

$$\text{minimize } F(X, U, F^{\text{flow}}) = F^{\text{flow}},$$

subject to (E.15)-(E.20) and

$$F^{\text{flow}} \geq w_j \left(\sum_{\substack{k' \in K: \\ k' \leq k}} \left(u_{ik'} + \sum_{j' \in J} p_{ij'} \cdot x_{ij'k'} \right) - r_j \right) - \mathcal{M} \cdot (1 - x_{ijk}), \forall i \in P, j \in J, k \in K.$$

To minimize the maximum weighted lateness,

$$\text{minimize } L(X, U, F^{\text{late}}) = F^{\text{late}},$$

subject to (E.15)-(E.17), (E.19), (E.20), and

$$F^{\text{late}} \geq w_j \left(\sum_{\substack{k' \in K: \\ k' \leq k}} \left(u_{ik'} + \sum_{j' \in J} p_{ij'} \cdot x_{ij'k'} \right) - d_j \right) - \mathcal{M} \cdot (1 - x_{ijk}), \forall i \in P, j \in J, k \in K$$

Chapter F.

Conclusion

The aim of this thesis was to design novel models and algorithms for optimization problems related to trucks' loading and unloading operations while taking personnel aspects into account. Despite the increasing trend toward automation at many modern cargo facilities, the trucks' unloading and loading operations still require the humans' inevitable involvement. That is due to their inherent flexibility and unique ability to respond to real-time changes, unlike automated systems. The workforce's involvement is required at both sides of the dock door. *Outside the terminal building*, a large number of trailers have to be coordinated within the yard. The semitrailers of incoming and outgoing trucks have to be transported from the parking lot to the terminal building and docked by backward maneuvering toward the dedicated dock door. Many terminals apply specialized yard tractors (also denoted as spotters) for the intra-yard trailer transportation. This way, road truckers do not have to wait to complete the trailer's (un-)loading and can directly leave the facility while semitrailers are shunted within the yard by terminal employees. *Inside the terminal building*, unloading and loading of a trailer consist of several manual tasks that can be executed only by humans. There is a relatively small body of literature that is concerned with integrating workforce scheduling aspects into daily operations at trailer terminals. This thesis intends to close this research gap by designing models and algorithms for novel truck scheduling and related problems, considering terminal workforce scheduling aspects.

On the inbound side, logistics workers unload cargo from the trailers of incoming trucks. The duration of the unloading step of a particular trailer strongly depends on the number of logistics workers who simultaneously unload the truck. Thus, terminal managers can affect the truck's unloading duration by adapting the right amount of workers. To the best of the author's knowledge, this aspect has found little attention in the

scientific literature so far. The majority of existing models presume fixed processing times for each truck. Chapter B of this thesis has proposed a novel *integrated truck and workforce scheduling* problem, which we denote as ITWS. We have observed ITWS for two representative settings: for conventional distribution centers and cross-docking terminals. The former considers that the unloaded freight is stored in the terminal at least for some time. In contrast, in the latter case, the unloaded shipments are directly transshipped to the outbound gates, without considering any intermediate storage inside the terminal.

We have formalized the resulting optimization problems and developed appropriate mixed-integer linear programming models. Moreover, we have reformulated them as interval scheduling problems and, based on this, designed different efficient heuristic approaches. One of the proposed techniques has shown to perform very well, providing optimal (or near to optimal) solutions for large-scale ITWS problem instances in short computational times. Moreover, we have compared the novel ITWS approach with the existing alternative successive methods. Here, firstly, we allocate a fixed amount of workers at each gate, who operate there for the whole planning horizon; subsequently, we solve the remaining truck scheduling problem considering those given workforce assignments. Our results have shown that the novel integrated planning approach is superior to the successive planning approach. Compared to the successive equal workforce approach (i.e., when the personnel is equally distributed among the terminal gates), the ITWS promises significant benefits. For the one thing, the integrated planning approach can enable more efficient utilization of the given workforce, obtaining better solutions in terms of the terminal's performance measure (i.e., smaller trucks' service times / less delayed freight). For the other thing, the optimized workforce allocation can lead to significant workforce savings (by up to 36% or 42% in DC and CD settings).

On the outbound side, the truck scheduling problem may consist of an additional decision, namely how to load shipments into the trucks such that the limited carrying capacities the trucks are not exceeded. This decision problem is relevant for cargo facilities in a many-to-few-configuration, where the consolidation takes place on the outbound side. Such a setting often arises at dispatch warehouses of part suppliers (e.g., in the automotive industry), which ship the ordered goods to an original equipment manufacturer. The existing studies in this field have only focused on either the truck loading problem or the truck scheduling problem. In Chapter C of this thesis, we have proposed a novel *outbound truck loading and scheduling problem* (dubbed as OTSLP), which solves both problems simultaneously. More precisely, the OTSLP consists of the following de-

cisions: on the one hand, items are to be loaded into the trucks considering their limited capacity (i.e., truck loading); on the other hand, each truck is to be scheduled within the given time window considering the limited terminal resources required to load the trucks (i.e., truck scheduling).

After formalizing the resulting optimization problem, we provided two mixed-integer linear programming models that can be applied to solve the considered optimization problem using a commercial standard solver. Moreover, we designed an exact branch-and-price algorithm that has shown to perform very well, optimally solving the most large-scale OTSLP instances within a few minutes. Finally, we explored the impact of the time window flexibility and workforce size on delivery punctuality. As one would expect, more restrictive time windows and a scarce workforce lead to worse delivery punctuality. These findings will be of interest to terminal managers when facing the trade-off between the time window flexibility for trucks and the workforce size vs. the terminal's service level (i.e., delivery punctuality). Specifically, the insights gained from this study may be helpful for terminal managers at the tactical planning level to design a proper time window management for trucks and define an adequate workforce size to achieve the desired service level.

Outside the terminal building, transportation of the trailers needs to be coordinated within the terminal yard. Many large trailer terminals apply specialized terminal tractors (also called spotters) and own employees for the intra-yard trailer shunting. In Chapter D of this thesis, we proposed a novel *robust spotter scheduling problem* (RSSP). RSSP assigns a set of transportation requests to a fleet of spotters such that the trailers can be (un-)loaded at the dock doors as scheduled in the previous planning step. Thereby, our goal is to find such robust spotter schedules that reduce the propagation of small unforeseen delays in the yard. We inserted buffer times between any pair of successive jobs. Based on this, we designed two robust objectives, first, maximizing the sum weighted buffer time of all jobs, and, second, maximizing the minimum weighted buffer time.

We developed efficient polynomial-time algorithms for both RSSP versions, which have shown remarkable performance in the numerical experiments. In particular, our approaches can find optimal solutions within few minutes for large-scale problem instances (considering 2000 transportation jobs and 400 spotters). Furthermore, we evaluated the robustness of the proposed techniques in a simulation study. We simulated some internal and external delays during the execution of original schedules and have assessed

the found solutions' robustness concerning different performance measures. Our findings reveal that using the right robustness objective can significantly reduce the delay propagation. Applying the proper robustness objective can lead to a reduction of the average waiting time per truck up to one hour and the average idle time per dock door up to five hours.

It is worth mentioning that the truck scheduling problems proposed in Chapters B and C consider the workforce-related aspects via ensuring that the amount of required workforce never exceeds the available limits. Neither of these models explicitly derives the exact plans for each worker, determining the worker's particular tasks and the time schedule. However, note that such schedules can be easily derived using the techniques proposed in Chapter D. Although we motivate the RSSP for the terminal staff scheduling outside the terminal building, we can also apply the proposed approaches for the intra-terminal workforce scheduling problems.

Finally, this thesis has also looked into a more general class of scheduling problems, namely scheduling a set of jobs with time windows on unrelated parallel machines. Such widespread scheduling problems have a broad range of applications for many practical planning problems in logistics and production, including the truck scheduling problem. In this context, we can interpret a truck with the given arrival and departure time as a job with the release date and deadline. Moreover, a terminal dock door corresponds to a machine that can process at most one job at a time. By observing the most general case of unrelated parallel machines, we address such a truck scheduling problem where each truck's processing time depends on different dock-specific factors.

Chapter E of this thesis has looked into such unrelated parallel machine scheduling problems and designed novel exact and heuristic approaches. Specifically, our exact algorithm relies on a logic-based Benders decomposition scheme, which we denote as *branch & Benders cut*. The heuristic is based on the problem's reformulation as a general set partitioning problem, which we refer to as *GSPP heuristic*. It uses the heuristic column selection techniques, developed in Chapter B. Both solution approaches are suited for three different minmax objectives and can be easily adapted to several model extensions. We have tested the algorithmic performance of the proposed solution methods in extensive computational experiments. More precisely, we have used newly generated problem instances and existing benchmark instance sets from the literature. Our results have shown that the branch & Benders Cut algorithm performs very well, optimally solving the majority of problem instances within five minutes. The GSPP

heuristic, too, has demonstrated successful performance. It can obtain optimal or near-optimal solutions in short computational times, especially for the challenging problem instances with tight time windows. Thus, depending on the type of the problem instance, one has to carefully choose an appropriate approach to obtain the desired solution within the given time limit.

Bibliography

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., and Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1):86–110.

Baptiste, P. and Sadykov, R. (2009). On scheduling a single machine to minimize a piecewise linear objective function: A compact mip formulation. *Naval Research Logistics (NRL)*, 56(6):487–502.

Bartholdi, J. J. and Gue, K. R. (2000). Reducing labor costs in an LTL crossdocking terminal. *Operations Research*, 48(6):823–832.

Bartholdi, J. J. and Gue, K. R. (2004). The best shape for a crossdock. *Transportation Science*, 38(2):235–244.

Bartusch, M., Möhring, R. H., and Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240.

Batista Abikarram, J., McConky, K., and Proano, R. (2019). Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing. *Journal of Cleaner Production*, 208:232–242.

Battini, D., Boysen, N., and Emde, S. (2013). Just-in-time supermarkets for part supply in the automobile industry. *Journal of Management Control*, 24(2):209–217.

Ben-Tal, A. and Nemirovski, A. (2002). Robust optimization–methodology and applications. *Mathematical Programming*, 92(3):453–480.

Ben-Tal, A. and Nemirovski, A. (2008). Selected topics in robust convex optimization. *Mathematical Programming*, 112(1):125–158.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Berghman, L. and Leus, R. (2015). Practical solutions for a dock assignment problem with trailer transportation. *European Journal of Operational Research*, 246(3):787–799.

Berghman, L., Leus, R., and Spieksma, F. C. (2014). Optimal solutions for a dock assignment problem with trailer transportation. *Annals of Operations Research*, 213(1):3–25.

Bierwirth, C. and Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627.

Bierwirth, C. and Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 244(3):675–689.

Blanquart, C. and Burmeister, A. (2009). Evaluating the performance of freight transport: a service approach. *European Transport Research Review*, 1(3):135–145.

Blazewicz, J., Cheng, T. E., Machowiak, M., and Oguz, C. (2011). Berth and quay crane allocation: a moldable task scheduling model. *Journal of the Operational Research Society*, 62(7):1189–1197.

Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. (2007). *Handbook on Scheduling*. Springer, New York. Chapter 12.

Bodnar, P., de Koster, R., and Azadeh, K. (2017). Scheduling trucks in a cross-dock with mixed service mode dock doors. *Transportation Science*, 51(1):112–131.

Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research*, 37(1):32–41.

Boysen, N., Briskorn, D., and Emde, S. (2016). Just-in-time vehicle scheduling with capacity constraints. *IIE Transactions*, 48(2):134–145.

Boysen, N., Briskorn, D., Fedtke, S., and Schmickerath, M. (2019). Automated sortation conveyors: A survey from an operational research perspective. *European Journal of Operational Research*, 276(3):796–815.

Boysen, N., Briskorn, D., and Tschöke, M. (2013). Truck scheduling in cross docking terminals with fixed outbound departures. *OR Spectrum*, 35(2):479–504.

Boysen, N., Emde, S., Hoeck, M., and Kauderer, M. (2015). Part logistics in the automotive industry: Decision problems, literature review and research agenda. *European Journal of Operational Research*, 242(1):107–120.

Boysen, N., Fedtke, S., and Weidinger, F. (2017). Truck scheduling in the postal service industry. *Transportation Science*, 51(2):723–736.

Boysen, N. and Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413–422.

Boysen, N., Fliedner, M., and Scholl, A. (2009). Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192(2):349–373.

Boysen, N., Fliedner, M., and Scholl, A. (2010). Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum*, 32(1):135–161.

Brandeau, M. L. and Chiu, S. S. (1989). An overview of representative problems in location research. *Management Science*, 35(6):645–674.

Briskorn, D., Leung, J., and Pinedo, M. (2011). Robust scheduling on a single machine using time buffers. *IIE Transactions*, 43(6):383–398.

Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J., and Lusby, R. (2011). Models for the discrete berth allocation problem: A computational comparison. *Transportation Research Part E: Logistics and Transportation Review*, 47(4):461–473.

Buijs, P., Danhof, H. W., and Wortmann, J. C. (2016). Just-in-time retail distribution: A systems perspective on cross-docking. *Journal of Business Logistics*, 37(3):213–230.

Buijs, P., Vis, I. F., and Carlo, H. J. (2014). Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 239(3):593–608.

Burdzik, R., Cieśla, M., and Sładkowski, A. (2014). Cargo loading and unloading efficiency analysis in multimodal transport. *PROMET-Traffic&Transportation*, 26(4):323–331.

Burkard, R., Dell'Amico, M., and Martello, S. (2009). Assignment problems society for industrial and applied mathematics. *Philadelphia, PA, USA*.

Campbell, J. F. and O'Kelly, M. E. (2012). Twenty-five years of hub location research. *Transportation Science*, 46(2):153–169.

Carlier, J. (1982). The one-machine sequencing problem. *European Journal of Operational Research*, 11(1):42–47.

Carlo, H. J. and Bozer, Y. A. (2011). Analysis of optimum shape and door assignment problems in rectangular unit-load crossdocks. *International Journal of Logistics Research and Applications*, 14(3):149–163.

Che, A., Zhang, S., and Wu, X. (2017). Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, 156:688–697.

Chen, B., Potts, C. N., and Woeginger, G. J. (1998). A review of machine scheduling: Complexity, algorithms and approximability. In *Handbook of combinatorial optimization*, pages 1493–1641. Springer.

Chen, J. H., Lee, D.-H., and Cao, J. X. (2012). A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):266–275.

Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: review and extensions. *Operations Research*, 58(1):130–148.

Cheng, T. and Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292.

Codato, G. and Fischetti, M. (2006). Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

Cohen, Y. and Keren, B. (2009). Trailer to door assignment in a synchronous cross-dock operation. *International Journal of Logistics Systems and Management*, 5(5):574–590.

Cordeau, J.-F., Laporte, G., Legato, P., and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation Science*, 39(4):526–538.

Demange, M., Ekim, T., Ries, B., and Tanasescu, C. (2015). On some applications of the selective graph coloring problem. *European Journal of Operational Research*, 240(2):307–314.

Demange, M., Monnot, J., Pop, P., and Ries, B. (2014). On the complexity of the selective graph coloring problem in some special classes of graphs. *Theoretical Computer Science*, 540–541:89–102.

Descartes (2019). Descartes dock appointment scheduling software. `https://www.descartes.com/documents/descartes-dock-appointment-scheduling`. (last access: August 2019).

Domschke, W., Drexl, A., Mayer, G., and Tadumadze, G. (2018). Betriebliche standortplanung. In *Planung logistischer Systeme*, pages 1–27. Springer. (in German).

Ebenlendr, T., Krčál, M., and Sgall, J. (2014). Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80.

Edis, E. B., Oguz, C., and Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, 230(3):449–463.

Emde, S. (2011). *Feeding parts to mixed-model assembly lines in the automotive industry with tow trains*. PhD thesis, Friedrich-Schiller-Universität, Jena.

Emde, S. (2017). Optimally scheduling interfering and non-interfering cranes. *Naval Research Logistics (NRL)*, 64(6):476–489.

Emde, S., Boysen, N., and Briskorn, D. (2014). The berth allocation problem with mobile quay walls: problem definition, solution procedures, and extensions. *Journal of Scheduling*, 17(3):289–303.

Emde, S. and Zehtabian, S. (2019). Scheduling direct deliveries with time windows to minimise truck fleet size and customer waiting times. *International Journal of Production Research*, 57(5):1315–1330.

Erengüç, Ş. S., Simpson, N. C., and Vakharia, A. J. (1999). Integrated production/distribution planning in supply chains: An invited review. *European Journal of Operational Research*, 115(2):219–236.

Eurostat (2020). Freight transport statistics - modal split. `https://ec.europa.eu/eurostat/statistics-explained/index.php/Freight_transport_statistics_-_modal_split`. (last access: April 2020).

Fanjul-Peyro, L. and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207(1):55 – 69.

Fanjul-Peyro, L. and Ruiz, R. (2011). Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research*, 38(1):301 – 309.

Fedtke, S. and Boysen, N. (2017). Layout planning of sortation conveyors in parcel distribution centers. *Transportation Science*, 51(1):3–18.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424.

Fliedner, M., Briskorn, D., and Boysen, N. (2016). Vehicle scheduling under the warehouse-on-wheels policy. *Discrete Applied Mathematics*, 205:52–61.

Friesen, D. K. and Langston, M. A. (1986). Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222–230.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman.

Gedik, R., Rainwater, C., H., N., and Pohl, E. A. (2016). Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. *European Journal of Operational Research*, 251(2):640 – 650.

Gharbi, A. and Haouari, M. (2002). Minimizing makespan on parallel machines subject to release dates and delivery times. *Journal of Scheduling*, 5(4):329–355.

Ghirardi, M. and Potts, C. (2005). Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *European Journal of Operational Research*, 165(2):457 – 467.

Goodloading (2019). How to plan the arrangement of pallets on a truck? 10 september 2019. `https://www.goodloading.com/en/blog/loading/how-to-plan-the-arrangement-of-pallets-on-a-truck/`, (last access: February 2020).

Graham, R., Lawler, E., Lenstra, J., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Gschwind, T., Irnich, S., Tilk, C., and Emde, S. (2020). Branch-cut-and-price for scheduling deliveries with time windows in a direct shipping network. *Journal of Scheduling*, 23:363–377.

Günther, M. and Nissen, V. (2010). Sub-daily staff scheduling for a logistics service provider. *KI-Künstliche Intelligenz*, 24(2):105–113.

Günther, M. and Nissen, V. (2014). A comparison of three heuristics on a practical case of sub-daily staff scheduling. *Annals of Operations Research*, 218(1):201–219.

Hall, N. G. and Posner, M. E. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14.

Held, M. and Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210.

Hermel, D., Hasheminia, H., Adler, N., and Fry, M. J. (2016). A solution framework for the multi-mode resource-constrained cross-dock scheduling problem. *Omega*, 59:157–170.

Herroelen, W. and Leus, R. (2004). Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42(8):1599–1620.

Hooker, J. (2011). *Logic-based methods for optimization: combining optimization and constraint satisfaction*, volume 2. John Wiley & Sons, Hoboken, New York.

Hooker, J. N. (2007). Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55(3):588–602.

Hosseini, S. D., Shirazi, M. A., and Karimi, B. (2014). Cross-docking and milk run logistics in a consolidation network: A hybrid of harmony search and simulated annealing approach. *Journal of Manufacturing Systems*, 33(4):567–577.

Imai, A., Nishimura, E., and Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port. *Transportation Research Part B: Methodological*, 35(4):401–417.

INFORM (2019). Yard management. `https://www.inform-software.com/logistics/yard-management`. last access: March 2019.

Jain, V. and Grossmann, I. E. (2001). Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on Computing*, 13(4):258–276.

Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Kaneko, J. and Nojiri, W. (2008). The logistics of just-in-time between parts suppliers and car assemblers in japan. *Journal of Transport Geography*, 16(3):155–173.

Kang, M.-H., Lee, S., Chu, Y.-G., Choi, S.-H., Won, S.-H., Cho, S.-W., and Kim, W.-S. (2017). Research on the design and economic analysis for the operation of cargo batch loading and unloading systems. *Journal of Korea Port Economic Association*, 33(1):1–10.

Karaenke, P., Bichler, M., and Minner, S. (2019). Coordination is hard: Electronic auction mechanisms for increased efficiency in transportation logistics. *Management Science*, 65(12):5884–5900.

Kayvanfar, V., Komaki, G. M., Aalaei, A., and Zandieh, M. (2014). Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times. *Computers & Operations Research*, 41:31–43.

Klose, A. and Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29.

Knop, D. and Koutecky, M. (2017). Scheduling meets n-fold integer programming. In *Proceedings of the 13th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP)*.

Kolen, A. W., Lenstra, J. K., Papadimitriou, C. H., and Spieksma, F. C. (2007). Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543.

Konur, D. and Golias, M. M. (2017). Loading time flexibility in cross-docking systems. *Procedia Computer Science*, 114:491–498.

Kouvelis, P. and Yu, G. (2013). *Robust discrete optimization and its applications*, volume 14. Kluwer Academic Publishers, Dordrecht.

Ladier, A.-L. and Alpan, G. (2015). Integrating truck scheduling and employee rostering in a cross-docking platform-an iterative approach. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 676–685. IEEE.

Ladier, A.-L. and Alpan, G. (2016). Cross-docking operations: Current research versus industry practice. *Omega*, 62:145–162.

Ladier, A.-L., Alpan, G., and Penz, B. (2014). Joint employee weekly timetabling and daily rostering: A decision-support tool for a logistics platform. *European Journal of Operational Research*, 234(1):278–291.

Lalla-Ruiz, E., Expósito-Izquierdo, C., Melián-Batista, B., and Moreno-Vega, J. M. (2016). A set-partitioning-based model for the berth allocation problem under time-dependent limitations. *European Journal of Operational Research*, 250(3):1001–1012.

Lancia, G. (2000). Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan. *European Journal of Operational Research*, 120(2):277–288.

Larbi, R., Alpan, G., Baptiste, P., and Penz, B. (2011). Scheduling cross docking operations under full, partial and no information on inbound arrivals. *Computers & Operations Research*, 38(6):889–900.

Lawler, E. L. (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5):544–546.

Lee, D.-H. and Qiu Wang, H. (2010). Integrated discrete berth allocation and quay crane scheduling in port container terminals. *Engineering Optimization*, 42(8):747–761.

Lee, S., Chang, T.-W., Won, J.-U., and Kim, Y.-J. (2014). Systematic development of cargo batch loading and unloading systems. *International Journal of Advanced Logistics*, 3(1-2):75–85.

Lenstra, J., Rinnooy Kan, A., and Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362.

Lenstra, J. K., Shmoys, D. B., and Tardos, É. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1-3):259–271.

Leung, J. Y. (2004). *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, Boca Raton, FL.

Leung, J. Y.-T. and Li, C.-L. (2008). Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116(2):251–262.

Leus, R. and Herroelen, W. (2007). Scheduling for stability in single-machine production systems. *Journal of Scheduling*, 10(3):223–235.

Li, C.-l., Cai, X., and Lee, C.-y. (1998). Scheduling with multiple-job-on-one-processor pattern. *IIE Transactions*, 30(5):433–445.

Li, L., Fonseca, D. J., and Chen, D.-S. (2006). Earliness–tardiness production planning for just-in-time manufacturing: A unifying approach by goal programming. *European Journal of Operational Research*, 175(1):508–515.

Li, Y., Lim, A., and Rodrigues, B. (2004). Crossdocking—JIT scheduling with time windows. *Journal of the Operational Research Society*, 55(12):1342–1351.

Lin, Y.-K., Pfund, M. E., and Fowler, J. W. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research*, 38(6):901–916.

Liu, J., Wan, Y.-w., and Wang, L. (2006). Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics (NRL)*, 53(1):60–74.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424.

McWilliams, D. L., Stanfield, P. M., and Geiger, C. D. (2005). The parcel hub scheduling problem: A simulation-based solution approach. *Computers & Industrial Engineering*, 49(3):393–412.

Miao, Z., Lim, A., and Ma, H. (2009). Truck dock assignment problem with operational time constraint within crossdocks. *European Journal of Operational Research*, 192(1):105–115.

Mokotoff, E. and Chrétienne, P. (2002). A cutting plane algorithm for the unrelated parallel machine scheduling problem. *European Journal of Operational Research*, 141(3):515–525.

Monaco, M. F. and Sammarra, M. (2007). The berth allocation problem: a strong formulation solved by a lagrangean approach. *Transportation Science*, 41(2):265–280.

NTC (2018). Load restraint guide 2018. `https://www.ntc.gov.au/sites/default/files/assets/files/Load-Restraint-Guide-2018.pdf`. (last access: September 2020).

NTI (2013). Dock planning standards. `https://www.novalocks.com/wp-content/uploads/Dock-Planning-Standards-Guide.pdf`. (last access: August 2019).

Oh, Y., Hwang, H., Cha, C. N., and Lee, S. (2006). A dock-door assignment problem for the korean mail distribution center. *Computers & Industrial Engineering*, 51(2):288–296.

Ou, J., Hsu, V. N., and Li, C.-L. (2010). Scheduling truck arrivals at an air cargo terminal. *Production and Operations Management*, 19(1):83–97.

Pfund, M., Fowler, J. W., and Gupta, J. N. (2004). A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers*, 21(3):230–241.

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, Berlin, fifth edition edition.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817.

Rijal, A., Bijvank, M., de Koster, R., et al. (2019). Integrated scheduling and assignment of trucks at unit-load cross-dock terminals with mixed service mode dock doors. *European Journal of Operational Research*, 278(3):752–771.

Sadjadi, S., Jafari, M., and Amini, T. (2009). A new mathematical modeling and a genetic algorithm search for milk run problem (an auto industry supply chain case study). *The International Journal of Advanced Manufacturing Technology*, 44(1-2):194–200.

Sanlaville, E. and Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795–811.

Schwerdfeger, S., Boysen, N., and Briskorn, D. (2018). Just-in-time logistics for far-distant suppliers: scheduling truck departures from an intermediate cross-docking terminal. *OR Spectrum*, 40(1):1–21.

Sels, V., Coelho, J., Dias, A. M., and Vanhoucke, M. (2015). Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem. *Computers & Operations Research*, 53:107–117.

Shabtay, D. and Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666.

Shakeri, M., Low, M. Y. H., Turner, S. J., and Lee, E. W. (2012). A robust two-phase heuristic algorithm for the truck scheduling problem in a resource-constrained crossdock. *Computers & Operations Research*, 39(11):2564–2577.

Shchepin, E. V. and Vakhania, N. (2005). An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33(2):127 – 133.

Shuib, A. and Fatthi, W. N. A. W. A. (2012). A review on quantitative approaches for dock door assignment in cross-docking. *International Journal on Advanced Science, Engineering and Information Technology*, 2(5):370–374.

Statista (2018). Transportleistung des straßengüterverkehrs in der europäischen union von 1995 bis 2016 (in milliarden tonnenkilometer). `https://de.statista.com/statistik/daten/studie/282301/umfrage/transportleistung-des-strassengueterverkehrs-in-der-eu/`. (in German) last access: August 2019.

Statista (2020). Anteil der lkw an der transportleistung im güterverkehr in deutschland in den jahren von 2013 bis 2023. `https://de.statista.com/statistik/daten/studie/12195/umfrage/anteil-der-lkw-am-gueterverkehr-in-deutschland/`. (in German), (last access: September 2020).

Steenken, D., Voß, S., and Stahlbock, R. (2004). Container terminal operation and operations research-a classification and literature review. *OR Spectrum*, 26(1):3–49.

Stephan, K. and Boysen, N. (2011a). Cross-docking. *Journal of Management Control*, 22:129–137.

Stephan, K. and Boysen, N. (2011b). Vis-à-vis vs. mixed dock door assignment: A comparison of different cross dock layouts. *Operations Management Research*, 4(3-4):150.

Tadumadze, G., Boysen, N., and Emde, S. (2020a). Robust spotter scheduling in trailer yards. *OR Spectrum*, 42(4):995–1021.

Tadumadze, G., Boysen, N., Emde, S., and Weidinger, F. (2019). Integrated truck and workforce scheduling to accelerate the unloading of trucks. *European Journal of Operational Research*, 278(1):343–362.

Tadumadze, G. and Emde, S. (2020). Loading and scheduling outbound trucks at a dispatch warehouse. *Working Paper TU Darmstadt*.

Tadumadze, G., Emde, S., and Diefenbach, H. (2020b). Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. *OR Spectrum*, 42:461–497.

Tong, C. J., Lau, H. C., and Lim, A. (1999). Ant colony optimization for the ship berthing problem. In *Annual Asian Computing Science Conference*, pages 359–370. Springer.

Tran, T. T., Araujo, A., and Beck, J. C. (2016). Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing*, 28(1):83–95.

Tsui, L. Y. and Chang, C.-H. (1990). A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering*, 19(1):309–312.

Tsui, L. Y. and Chang, C.-H. (1992). An optimal solution to a dock door assignment problem. *Computers & Industrial Engineering*, 23(1-4):283–286.

UNCTAD (2016). Review of maritime transport 2016. Technical report, United Nations Conference on Trade and Development.

Vacca, I., Bierlaire, M., and Salani, M. (2007). Optimization at container terminals: status, trends and perspectives. In *7th Swiss Transport Research Conference*.

Van Belle, J., Valckenaers, P., and Cattrysse, D. (2012). Cross-docking: State of the art. *Omega*, 40(6):827–846.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., and Leus, R. (2005). The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, 97(2):227–240.

Vis, I. F. and Roodbergen, K. J. (2011). Layout and control policies for cross docking operations. *Computers & Industrial Engineering*, 61(4):911–919.

VRS (2011). Problemzone rampe. *Verkehrsrundschau*, 24/2011:24. (in German).

VRS (2012). Nadelöhr laderampe. *Verkehrsrundschau*, 17/2012:20–23. (in German).

Wang, Y. and Chen, F. (2019). Packed parts delivery problem of automotive inbound logistics with a supplier park. *Computers & Operations Research*, 101:116–129.

Wu, X. and Che, A. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega*, 82:155–165.

Xu, D., Li, C.-L., and Leung, J. Y.-T. (2012). Berth allocation with time-dependent physical limitations on vessels. *European Journal of Operational Research*, 216(1):47–56.

Yano, C. A., Bozer, Y., and Kamoun, M. (1998). Optimizing dock configuration and staffing in decentralized receiving. *IIE Transactions*, 30(7):657–668.

Yu, V. F., Sharma, D., and Murty, K. G. (2008). Door allocations to origins and destinations at less-than-truckload trucking terminals. *Journal of Industrial and Systems Engineering*, 2(1):1–15.

Yu, W. and Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184(1):377–396.

Zaerpour, N., Yu, Y., and de Koster, R. B. (2015). Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Production and Operations Management*, 24(8):1266–1284.

Zenker, M. and Boysen, N. (2018). Dock sharing in cross-docking facilities of the postal service industry. *Journal of the operational research society*, 69(7):1061–1076.