New solution approaches for optimization problems with combinatorial aspects in logistics management

Heiko Diefenbach

Dissertation
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. Christoph H. Glock
Second assessor: Prof. Dr. Simon Emde

Darmstadt, 2023

Diefenbach, Heiko
New solution approaches for optimization problems with combinatorial aspects in logistics management
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2023
Tag der mündlichen Prüfung: 26.04.2023

Danksagung

Diese Dissertation markiert das Ende eines langen, aber spannenden Weges, den ich in den letzten fünf Jahren am Fachgebiet Produktion und Supply Chain Management der TU Darmstadt beschritten habe. Während dieser Zeit habe ich vielfache Unterstützung erfahren, für die ich mich nachfolgend bedanken möchte.

Mein besonderer Dank gilt meinem Doktorvater, Prof. Dr. Christoph Glock, für die hervorragende Betreuung und fortwährende Unterstützung. Prof. Glock hat mich stets gefördert und mir dabei etliche Möglichkeiten eröffnet, mich weiterzuentwickeln und neue Erfahrungen zu sammeln – sei es in der Lehre, bei Forschungsprojekten oder bei meinem Auslandsaufenthalt an der Toronto Metropolitan University. Ich konnte viel von Prof. Glock lernen und bin dankbar für seine wertvollen Ratschläge und wertschätzende Art. Ein großer Dank gilt auch meinem Korreferenten, Prof. Dr. Simon Emde, der mich zu meiner Promotion ermuntert und währenddessen vielfach mit seinem großartigen Fachwissen und seinem entscheidenden Mitwirken in etlichen gemeinsamen Publikationen unterstützt hat. Auch Jun.-Prof. Dr. Eric Grosse und Dr. Giorgi Tadumadze möchte ich für ihre zentralen Beiträge bei unseren gemeinsamen Publikationen und ihre freundschaftliche Unterstützung während meiner Promotionszeit herzlich danken.

Darüber hinaus danke ich den Mitgliedern meiner Prüfungskommission, Prof. Dr. Ralf Elbert, Prof. Dr. Felix Weidinger und Prof. Ph.D. Frank Pisch, sowie dem Protokollanten, Julian Best, für ihre Zeit und aufschlussreichen Kommentare im Rahmen meiner Disputation.

Meinen Kollegen an der TU Darmstadt gilt ebenfalls meine tiefe Dankbarkeit für die gemeinsame Arbeit, ihre herzliche Unterstützung und die vielen anregenden Gespräche zu akademischen und nicht-akademischen Themen, die mich vielfach bereichert haben. Ein besonderer Dank gebührt in diesem Rahmen Dr. Fabian Beck, nicht nur weil er als erfahrenster Mitarbeiter am Fachgebiet immer guten Rat weiß und jederzeit hilfsbereit ist, sondern auch für unseren außerfachlichen Austausch, aus dem eine wichtige Freundschaft erwachsen ist, die mir insbesondere während der Covid-Pandemie viel Durchhaltevermögen gegeben hat. Mein Dank gilt auch meinem ehemaligen Bürokollegen Marc Füchtenhans, mit dem ich etwa zeitgleich die Arbeit am Fach-

gebiet aufgenommen und viele gemeinsame Erfahrungen gesammelt habe; ich danke ihm für die immer gute Zusammenarbeit, unsere vielen interessanten Gespräche und die Freundschaft, die sich zwischen uns entwickelt hat. Bei Kerstin Spiehl möchte ich mich für ihre Hilfe bei allen administrativen Angelegenheiten bedanken. Für die inspirierende und lehrreiche Zusammenarbeit in verschiedenen Forschungsprojekten bedanke ich mich bei Prof. Dr. Patrick Neumann, Prof. Dr. Kai-Oliver Schocke, Dr.-Ing. Tim Steinebach, Dr.-Ing. Jurij Wakula, Alexander Lunin und Nathalie Erlemann.

Zu guter Letzt danke ich meiner Familie und meinen Freunden, die immer an meiner Seite standen, in schweren Zeiten für mich da waren und mir auf meinem Weg stets viel Kraft gespendet haben.

Zusammenfassung

Diese Dissertation umfasst fünf Artikel, die zwischen 2019 und 2022 in wissenschaftlichen Fachzeitschriften veröffentlicht wurden. Die Artikel befassen sich mit logistischen Optimierungsproblemen aus drei verschiedenen Themenfeldern mit dem Schwerpunkt Intralogistik. Alle betrachteten Optimierungsprobleme weisen starke kombinatorische Aspekte auf. Zur Lösung der betrachteten Probleme werden verschiedene Lösungsansätze einschließlich unterschiedlicher Dekompositionstechniken verwendet.

Artikel 1 beschäftigt sich mit der Optimierung des Layouts und der Lagerplatzvergabe in Lagern mit U-förmigen Kommissionierzonen. Dabei werden zwei Optimierungsziele betrachtet, die Minimierung der zurückgelegten Wegstrecke und der körperlichen Belastung des Kommissionierers während der Kommissionierung. Zur Lösung des Problems wird ein semantischer Dekompositionsansatz vorgestellt, der das Problem in polynomialer Zeit löst. In einer Rechenstudie zeigt sich, dass beide betrachteten Ziele weitgehend komplementär sind. Darüber hinaus werden Empfehlungen für eine vorteilhafte Layoutgestaltung und Lagerplatzvergabe abgeleitet.

Artikel 2 betrachtet die Fragestellung, wie Routenzüge mit Transportbehältern beladen werden sollten, um die physische Belastung des eingesetzten Personals beim Be- und Entladen zu minimieren. Es wird gezeigt, dass das Problem NP-schwer ist und semantisch dekompositioniert werden kann. Unter Ausnutzung der Dekomposition wird das Problem exakt mittels dynamischer Programmierung und heuristisch mittels einer Greedy Randomized Adaptive Search Procedure gelöst. Eine anschließende Rechenstudie zeigt, dass beide Verfahren leistungsfähig sind. Ferner untersucht sie den Einfluss des Designs der Routenzugwagen auf das Optimierungsziel.

Artikel 3 befasst sich mit dem Problem der Belegungsplanung unabhängiger paralleler Aggregate bei Aufträgen mit Zeitfenstern, einem NP-schweren Optimierungsproblem, das unter anderem bei der Zuweisung von Schiffen zu Liegeplätzen und der Abfertigungsplanung an Lkw-Docks Anwendung findet. Der Artikel stellt ein exaktes logikbasiertes Benders Dekompositionsverfahren und einen heuristischen Lösungsansatz vor, der auf einer Mengenpartitionierungsformulierung des Problems beruht. Darüber hinaus werden drei verschiedene Ziele, nämlich die Mini-

mierung der maximal benötigten Zeitspanne, der maximalen Durchlaufzeit und der maximalen Verspätung, berücksichtigt. Beide Verfahren weisen in der abschließenden Rechenstudie gute Leistungen auf.

Artikel 4 befasst sich mit dem Problem der Routenplanung bei der Kommissionierung in einer U-förmigen Kommissionierzone mit dem Ziel, die zurückgelegte Wegstrecke zu minimieren. Das Problem wird als NP-schwer klassifiziert. Es werden ein exaktes logikbasiertes Benders Dekompositionsverfahren sowie ein heuristischer Ansatz basierend auf dynamischer Programmierung entwickelt, deren Leistungsfähigkeiten mittels Rechenstudie demonstriert werden. Darüber hinaus werden verschiedene Lagerplatzvergabestrategien erörtert und in einer numerischen Studie miteinander verglichen.

Artikel 5 untersucht die Einsatzplanung von elektrisch betriebenen Routenzügen in der innerbetrieblichen Produktionslogistik. Das Problem wird als Electric Vehicle Scheduling Problem aufgefasst, bei dem die Routenzüge vorterminierten Fahrten zugewiesen werden müssen. Da die Reichweite der Routenzüge begrenzt ist, müssen zwischen den Fahrten Ladepausen eingeplant werden, die zusätzliche Fahrstrecke und Zeit erfordern. Das Ziel besteht darin, die erforderliche Anzahl an Routenzügen zu minimieren. Das Problem wird als NP-schwer klassifiziert. Zur Lösung wird ein Branch-and-Check-Ansatz vorgeschlagen, der für verschiedene Ladetechnologien anwendbar ist, einschließlich Batterietausch und kabelgebundenem Laden mit nichtlinearem Ladungsanstieg. In einer Rechenstudie wird die Eignung des Lösungsansatzes für den praktischen Einsatz demonstriert. Außerdem wird untersucht, welche Einflüsse die maximale Kapazität der Batterien und die verwendete Ladetechnologie haben.

Abstract

This dissertation comprises five papers, which have been published in scientific journals between 2019 and 2022. The papers consider logistic optimization problems from three different subjects with a focus on intra-logistics. All considered optimization problems have strong combinatorial aspects. To solve the considered problems, various solution approaches including different decomposition techniques are employed.

Paper 1 investigates the optimization of the layout and storage assignment in warehouses with U-shaped order picking zones. The paper considers two objectives, namely minimizing the order picker's walking distance and physical strain during order picking. To solve the problem, a semantic decomposition approach is proposed, which solves the problem in polynomial time. In a computational study, both considered objectives are found to be mostly complementary. Moreover, suggestions for advantageous layout designs and storage assignments are derived.

Paper 2 considers the problem of how to stow bins on tow trains in order to minimize the handling personnel's physical strain for loading and unloading. The problem is shown to be NP-hard and decomposed semantically. Utilising the decomposition, the problem is solved exactly with dynamic programming and heuristically with a greedy randomized adaptive search procedure. A consecutive computational study shows that both procedures perform well. Beyond that, it investigates the influence of the tow train wagons' design on the considered objective.

Paper 3 is concerned with the problem of scheduling jobs with time windows on unrelated parallel machines, which is a NP-hard optimization problem that has applications, i.a., in berth allocation and truck dock scheduling. The paper presents an exact logic-based Benders decomposition procedure and a heuristic solution approach based on a set partitioning formulation of the problem. Moreover, three distinct objectives, namely minimizing the makespan, the maximum flow time, and the maximum lateness are considered. Both procedures exhibit good performances in the concluding computational study.

Paper 4 addresses the problem of order picker routing in a U-shaped order picking zone with the objective of minimizing the covered walking distance. The problem is proven to be NP-hard. An

exact logic-based Benders decomposition procedure as well as a heuristic dynamic programming approach are developed and shown to perform well in computational tests. Beyond that, the paper discusses different storage assignment policies and compares them in a numeric study.

Paper 5 studies scheduling electrically powered tow trains in in-plant production logistics. The problem is regarded as an Electric Vehicle Scheduling Problem, where tow trains must be assigned to timetabled service trips. Since the tow trains' range is limited, charging breaks need to be scheduled in-between trips, which require detours and time. The objective consists in minimizing the required fleet size. The problem is shown to be NP-hard. To solve the problem, Paper 5 proposes a branch-and-check approach that is applicable for various charging technologies, including battery swapping and plug-in charging with nonlinear charge increase. In a computational study, the approach's practical applicability is demonstrated. Moreover, influences of the batteries' maximum capacity and employed charging technology are investigated.

Contents

List of Figures

List of Tables

List of Abbreviations

BAP	Berth allocation problem
BCH	Branch-and-check
B&BC	Branch and Benders cut
BD	Benders decomposition
BDP	Bounded dynamic programming
BTPAP	Bin to position assignment problem
CBD	Combinatorial Benders decomposition
CC-CV	Constant current-constant voltage
cf.	Confer (compare)
cm	Centimeter
CP	Constraint programming
CP-CV	Constant power-constant voltage
CPU	Central processing unit
CSCMP	Council of Supply Chain Management Professionals
DP	Dynamic program
EAWS	Ergonomic Assessment Work Sheet
ec-USLPP	Economic U-shaped storage layout planning problem
EDD	Earliest due date
ERD	Earliest release date
er-USLPP	Ergonomic U-shaped storage layout planning problem
e.g.	Exempli gratia (for example)
EPA	European Pallet Association e.V.
ETTLP	Ergonomic tow train loading problem
EU	European Union
EVSP	Electric Vehicle Scheduling Problem
EVSP-MD-FS	Multi-depot EVSP with the objective of minimizing the fleet size
GAP	Generalized assignment problem
GB	Gigabyte
GDP	Gross domestic product
GHz	Gigahertz
GRASP	Greedy randomized adaptive search procedure
GSPP	Generalized Set Partitioning Problem

h	Hours
i.a.	Inter alia (among other things)
i.e.	Id est (that is)
IP	Integer program
ITWS	Integrated truck and workforce scheduling problem
ITWS-DC	ITWS for the distribution center context
kcal	Kilocalories
kg	Kilogram
LMRIS	Liberty Mutual Research Institute for Safety
m	Meters
MILP	Mixed-integer linear program
MIP	Mixed-integer program
MIP-OP	Original mixed integer program
MMH	Manual materials handling
MP	Master problem
MSD	Muscular-skeletal disorders
NCR	Institute of Medicine and National Research Council
NIOSH	National Institute for Occupational Safety and Health
no.	Number
OCRA	Occupational Risk Assessment
OR	Operations research
PC	Pallet cage
PRP-UA	Picker routing problem in a U-shaped pick area
RAM	Random-access memory
RBTWP	Relaxed bin to wagon problem
s	Seconds
SA	Sweep algorithm
SOC	State of charge
TSP	Traveling Salesman Problem
UNCTAD	United Nations Conference on Trade and Development
USLAP	U-shaped layout assignment problem
USLPP	U-shaped storage layout planning problem
VDA	Verband der Automobilindustrie
VSP	Vehicle Scheduling Problem

I. Introduction

The Council of Supply Chain Management Professionals defines logistics management as the "part of supply chain management that plans, implements, and controls the efficient, effective forward and reverses flow and storage of goods, services and related information between the point of origin and the point of consumption in order to meet customers' requirements" (CSCMP, 2018). The management of logistic processes plays a crucial role in the success of businesses. It enables the production of goods and provision of services by coordinating inputs, production factors, outputs, supply, and demand in a spacial and timely manner (Yalaoui et al., 2012, Murphy Jr and Knemeyer, 2018, Blanchard, 2021). Without logistics, almost no business could operate.

The importance of logistics becomes further eminent on an economic scale. Globally, logistic operations are estimated to account for 13.7% of the world's GDP (Bowersox et al., 2003). In the European Union, the logistic sector (which does not include logistic operations within companies) has a share of close to 14% on the total GDP (Satta et al., 2011). The European Commission estimates that in the European Union, on average, logistic expenses contribute to roughly one eighths of products' final values; about half of these expenses can potentially be saved by better logistics management (European Commission, 2022).

Logistics can be divided into external logistics and internal logistics with the latter often being referred to as intra-logistics (Gudehus and Kotzab, 2012). The former is concerned with logistic operations in-between geographically separated locations, e.g., the shipments of goods between different facilities or different agents in a supply chain. The latter considers logistic operations within a single facility, e.g., a production plant or a logistics center. Important intra-logistic operations include warehousing and in-plant transport of goods and materials.

Recent trends such as the mass customization of products (Da Silveira et al., 2001, Boysen et al., 2015), the shortening of product life cycles (Tyulin et al., 2020), or the demand for highly responsive supply chains (Negri et al., 2017) present various challenges for companies, especially in a competitive global market. To face these, many manufacturing companies have been re-

designing their production systems, e.g., by employing flexible production methods (Da Silveira et al., 2001) or by adopting lean production concepts (Jasti and Kodali, 2015). While this offers a lot of potential, it also increases the systems' complexity and requires expanded management effort (Da Silveira et al., 2001). It is therefore imminent to plan logistics efficiently. This is especially important for intra-logistics, as is presents an integral part of any production system (Gudehus and Kotzab, 2012, Boysen et al., 2015).

Various planning aspects that occur in (intra-)logistics management can be translated into mathematical optimization problems, where the aim is to select a solution out of a domain of possibilities (i.e., the solution space) to minimize or maximize a certain objective function (Griffis et al., 2012, Deroussi, 2016). A lot of optimization problems in (intra-)logistics are combinatorial (or exhibit strong combinatorial characteristics), which means that the solution space presents a (mostly) discrete set of possibilities (Papadimitriou and Steiglitz, 1998). Classical examples include the Traveling Salesman Problem (cf., Applegate et al., 2011), Bin Packing (cf., Yao, 1980), or Matching Problems (cf., Lovász and Plummer, 2009).

Many combinatorial optimization problems – especially those classified as NP-hard (for a formal definition, see Garey and Johnson, 1979) – are notorious for being computationally challenging to solve (Korte et al., 2011). This has sparked intensive research in the development and refinement of algorithms and procedures that are capable of finding good or, ideally, optimal solutions to combinatorial optimization problems. Some successful approaches include methods of (mixed-)integer programming (cf., e.g., Wolsey, 2007), problem decomposition techniques, e.g., Benders decomposition (BD, cf., Rahmaniani et al., 2017), dynamic programming (DP, Bellman, 1954), and the development of heuristics (cf., Zanakis and Evans, 1981). The suitability and performance of different approaches depend on the specific problem, however. Moreover, the approaches need to be deliberately adapted to a problem's characteristics.

Motivated by the importance of logistics management, this cumulative dissertation comprises five papers considering different logistic planning aspects with a focus on intra-logistics. The papers discuss the respective planning problems, present formal optimization models, and provide suitable solution procedures. All considered problems have strong combinatorial aspects. The developed solution procedures utilize various optimization methods with an emphasis on problem decomposition. Numerical studies are used in all five papers to test and validate the performance of the proposed solution procedures and to attain general insights into the considered logistic problems. The contribution of this dissertation is therefore threefold: First, it provides algorithms and procedures to better manage important logistic planning problems. Second, it contributes to the theory, development, and expansion of solution approaches for (combinato-

rial) optimization problems. Third, it provides helpful insights for practitioners and researchers alike.

All papers included in this cumulative dissertation have been published in peer-reviewed scientific journals as shown in Table I.1. Although the papers consider different planning problems and varying solution approaches, they are linked by multiple aspects: the subjects, the objectives, and the employed optimization methodology. Table I.2 provides an overview. In the following, these aspects are discussed. Afterwards, each paper is presented in a separate section.

no.	year	authors	title	journal
1	2019	H. Diefenbach, C. H. Glock	Ergonomic and economic optimization of layout and item assignment of a U-shaped order picking zone	Computers & Industrial Engineering
2	2019	H. Diefenbach, S. Emde, C. H. Glock	Loading tow trains ergonomically for just-in-time part supply	European Journal of Operational Research
3	2020	G. Tadumadze, S. Emde, H. Diefenbach	Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines	OR Spectrum
4	2021	H. Diefenbach, S. Emde, C. H. Glock, E. H. Grosse	New solution procedures for the order picker routing problem in U-shaped pick areas with a movable depot	OR Spectrum
5	2022	H. Diefenbach, S. Emde, C. H. Glock	Multi-depot electric vehicle scheduling in in-plant production logistics considering non-linear charging models	European Journal of Operational Research

Table I.1.: Overview of papers included in this dissertation.

paper		subject			objective		optimization methodology				
no.	titel	warehousing: U-shaped order picking zones	intralogistic transport: tow trains	parallel machine scheduling	efficiency	ergonomics	(mixed-) integer programming	problem decomposition	logic-based BD/ branch-and-check	dynamic programming	heuristics
1	Ergonomic and economic optimization of layout and item assignment of a U-shaped order picking zone	✓			✓	✓	✓	✓			
2	Loading tow trains ergonomically for just-in-time part supply		✓			✓	✓	✓		✓	✓
3	Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines			✓	✓		✓	✓	✓	✓	✓
4	New solution procedures for the order picker routing problem in U-shaped pick areas with a movable depot	✓			✓		✓	✓	✓	✓	✓
5	Multi-depot electric vehicle scheduling in in-plant production logistics considering non-linear charging models		✓		✓		✓	✓	✓		

abbreviations: no. = number; BD = Benders decomposition

Table I.2.: Overview of the contents and methodology of papers included in this dissertation.

I.1. Subjects considered in this dissertation

The papers that make up this cumulative dissertation consider three different subjects: the design and operation of U-shaped order picking zones, the operation of tow train systems, and parallel machine scheduling, which, i.a., has applications in truck scheduling and berth allocation.

Papers 1 and 4 are concerned with the design and operation of U-shaped order picking zones. Order picking is the process of retrieving items from storage based on (customer) orders. It is estimated that more than half of a warehouse's operational costs can be attributed to order picking (Coyle et al., 1996, De Koster et al., 2007, Tompkins et al., 2010, Richards, 2017), where, on average, operational warehousing costs account for roughly one fifth of a company's total logistics costs (De Koster et al., 2007). Hence, there is a strong incentive to plan and operate order picking systems efficiently. The majority of order picking systems are so-called picker-to-parts systems, where human order pickers move through the storage area to collect required items from stationary storage locations, e.g., racks (De Koster et al., 2007). The main advantages of these systems are the flexibility of human workers and the lower investment costs compared to automated parts-to-picker systems (Grosse et al., 2015, Tompkins et al., 2010).

Traditional layout designs for picker-to-parts warehouses consist of racks arranged in parallel aisles that might be separated by one or multiple perpendicular cross aisles. More recently proposed layouts, such as the flying-V, fishbone or chevron layouts, extend this concept by rotating areas of parallel aisles by some angle and/or employing diagonal cross aisles (cf., e.g., Pohl et al., 2009, Öztürkoğlu et al., 2012). An alternative to parallel aisle layouts are warehouses with multiple U-shaped picking zones, which were first described by Glock and Grosse (2012). Here, the warehouse is separated into multiple zones each containing racks or stacked stillages (also called pallet cages) arranged in a U-shape with a depot (also called base) in the middle, where picked items are dropped off. Compared to parallel aisle warehouses, warehouses with U-shaped order picking zones better facilitate storing items in groups, as each group can be stored in a separate U-shaped zone. Following this idea, a prominent application for warehouses with U-shaped order picking zones is part kitting, where parts for different assembly stations are pre-packed at the warehouse close to the assembly line. Here, the portfolio of parts required at each assembly station presents a group that is stored in a separate U-shaped zone (Glock and Grosse, 2012).

Glock and Grosse (2012) identified certain planning problems that need to be solved for warehouses with U-shaped order picking zones, which includes determining the layout (i.e., width, length, and depot location) of a zone's U-shape, the assignment of items to storage locations (i.e, stillages) within a zone, and the routing of the picker during order picking. Glock and Grosse (2012) address the first problem by simply comparing a few layout alternatives. For the second problem, they suggest and compare five different storage assignment policies, i.e., simple rules that assign items to stillages based on their demand rate. The third problem is solved via

a simple sweep algorithm. Exact solution procedures are not presented, resulting in a lack of insight into the solution quality of the proposed approaches. Papers 1 and 4 aim to address these shortcomings.

Deviating from Glock and Grosse (2012) where order pickers pick multiple items per tour, Paper 1 considers a situation where only one items is picked at a time, which eliminates complex picker routing. Otherwise, the assumptions are in line with Glock and Grosse (2012). For the remaining problems of finding an optimal layout and storage location assignment, the paper proposes a mixed-integer program (MIP) and an exact polynomial-time solution procedure based on a decomposition of the problem. The paper considers two different objectives, namely minimizing the order picker's expected walking distance and expected physical strain to process an order. In a subsequent computational study, it provides insights into the procedure's runtime, properties of good layouts and storage assignments, and the trade-off between both considered objectives.

Paper 4 considers the same situation as Glock and Grosse (2012). It addresses the problem of simultaneously finding optimal locations for the depot and optimal routes for the order picker with the objective of minimizing the walking distance during order picking. The problem is proven to be NP-hard. To solve the problem, the paper develops an exact procedure based on logic-based BD. Since this procedure struggles to solve problems of practically relevant size in subsequently conducted computational experiments, the paper also presents a heuristic DP approach that extends the concept of the sweep algorithm of Glock and Grosse (2012). The newly developed heuristic is theoretically guaranteed and numerically demonstrated to outperform the sweep algorithm in a computational study. Besides comparing the performance of the proposed algorithms, the computational study draws some insights on the position and properties of the depot, and investigates the influence of differently skewed demand distributions of the stored items. Beyond that, a new storage assignment policy is proposed and shown to compare favorable to the ones considered by Glock and Grosse (2012).

The second subject considered in this cumulative dissertation is intra-logistic transport using tow train systems; Papers 2 and 5 are concerned with this topic. As mentioned earlier, companies have been diversifying their product portfolios and adopting lean production strategies to face current trends and stay competitive in a global market. One principle of lean production is to minimize stock at the assembly line to reduce search effort and increase productivity (Jasti and Kodali, 2015). Combined with the need for a vast variety of parts to produce enlarged product portfolios, this has led to the practice of part kitting (cf., Limère et al., 2012), where required

parts are pre-sorted and pre-packed in bins at warehouses and delivered to assembly stations in a cyclic just-in-time manner (Emde et al., 2012). To facilitate the resulting frequent, small-lot deliveries, tow train systems are often employed.

A tow train (also called tugger train) consists of an electrically powered towing vehicle with a handful of wagons attached. The wagons are usually equipped with shelves to store and transport the kitted parts. Typical tow train systems comprise a couple to multiple dozen tow trains, where each tow train supplies multiple stations with parts. In the majority of tow train systems, human drivers are responsible for operating the tow trains, which does not only include driving but also loading and unloading the tow trains (Lieb et al., 2017).

The management of tow train systems requires careful planning in various respects. On the long-term end, this includes the planning of in-plant warehouse locations; on the operational level, it includes the routing, scheduling, and loading of tow trains (Emde et al., 2012). Most of these aspects have been considered by previous research either as individual problems or in holistic approaches. However, previous planning approaches had only focused on technical aspects while human factors were mostly ignored. Loading and unloading tow trains manually requires repeated lifting, lowering and carrying of bins weighing up to multiple kilogram. High physical strains for tow train drivers are the result. Not only does this entail discomfort for the drivers, but it also increases their likelihood to develop muscular-skeletal disorders (MSD).

Paper 2 addresses this issue by determining how bins with different parts and therefore different weights should be assigned to the different wagons, shelves and shelf levels of a tow train to minimize the driver's physical strain for loading and unloading the bins. The paper acknowledges the usual planning hierarchy of tow train systems and assumes loads, tours and routes are already fixed when the tow train is loaded. Furthermore, constraints are applied to ensure the tow train is loaded in a way that makes unloading at the assembly stations time-efficient to facilitate the just-in-time delivery concept. The paper presents a MIP model for the considered problem and proves NP-hardness. It proposes an exact and a heuristic solution procedure both based on a hierarchical decomposition of the problem. While the exact procedure uses DP to handle the hierarchically superordinate problem, the heuristic applies a greedy randomized adaptive search procedure (GRASP). In a computational study, both procedures are shown to perform well. Moreover, the computational study investigates influences of the relative filling of the tow train as well as various design aspects of the wagons and shelves.

Tow trains are electrically powered vehicles, as combustion engines are generally not legal on a shop floor. With current battery technology, they are usually unable to operate for a whole shift without recharging. Hence, charging breaks need to be scheduled in-between service trips (i.e.,

trips, where a tow train supplies assembly stations). Previous planning approaches have mostly ignored this aspect, which results in incomplete or infeasible plans, where trips are timetabled but not assigned to tow trains in a manner that acknowledges the need for charging breaks. Paper 5 aims to resolve this issue by modeling the problem as an Electric Vehicle Scheduling Problem with the objective of minimizing the number of required tow trains, which presents the main cost driver. The paper assumes that tow trains must visit distinct charging stations to recharge their batteries. It considers different practically relevant charging technologies, including battery swapping and plug-in charging with realistic non-linear increase of battery charge. An integer program is formulated and the problem is shown to be NP-hard. To solve the problem, the paper proposes an exact branch-and-check solution procedure. It is shown to perform well in a computational study, which also draws some insights on the influence of the tow trains' maximum battery capacity and the used charging technology.

Paper 3 considers the problem of scheduling jobs with time windows on unrelated parallel machines. Unlike the previous two subjects that focus on specific intra-logistic aspects, this subject presents a more generalized optimization problem that has various applications in logistics. Among them are berth allocation and truck scheduling at truck docks, which are discussed briefly in the following and in more detail in Paper 3.

In machine scheduling, a job denotes a task that should be processed, which requires a certain processing time. If jobs have time windows, processing each job should start and finish within a pre-specified time frame. A machine presents an entity – not necessarily a machine in the narrow sense – that is used to process jobs. The notion that machines are unrelated means that jobs can have different processing times on different machines. Parallel machines mean that all machines are on the same hierarchical level and that there is no requirement for jobs to be processed on a sequence of machines (Graham et al., 1979).

Berth allocation is a problem from maritime logistics, which deals with the question of how to assign arriving vessels to quays at a harbor to load and unload cargo (Bierwirth and Meisel, 2010, 2015). In the notion of machine scheduling, quays (which might have different equipment, e.g., cranes, and therefore different processing speeds) can be considered as parallel unrelated machines. The jobs are given by the vessels that need to be (un-)loaded. As vessels do not necessarily need to be processed immediately upon arrival but still have certain due dates, the jobs have time windows.

Truck scheduling is concerned with the question of how to assign arriving trucks to truck docks to (un-)load them at a facility, e.g., a distribution center or production plant (Boysen and Flied-

ner, 2010). Hence, it presents a logistics problem that connects external with internal logistics. From a machine scheduling perspective, truck docks can be considered as parallel machines. As the equipment, personnel, and hence processing speed can vary between truck docks, they present unrelated machines (Tadumadze et al., 2019). Similar to the vessels that need to be (un-)loaded in berth allocation, the jobs consist in (un-)loading the arriving trucks. It is very common that trucks have certain time windows, in which they must be processed upon arrival (Tadumadze et al., 2019).

Paper 3 considers three distinct objectives for the problem of scheduling jobs with time windows on unrelated parallel machines. The first is to minimize the makespan, i.e., the time when the last job is completed. The second is to minimize the maximum flow time, i.e., the time between a job becoming available and being completed. The third is to minimize the maximum lateness, i.e., the time by which a job's time window is exceeded if timely completion is not possible. Depending on the application of the problem, all three objectives can be relevant. Paper 3 presents two novel solution approaches for the problem – an exact logic-based BD procedure and a heuristic approach based on a generalized set partitioning formulation of the problem. In a computational study, both approaches are shown to perform well.

I.2. Objectives considered in this dissertation

As described in Section I.1, the papers in this dissertation consider different logistic optimization problems with differing objectives. Nevertheless, the objectives can be grouped into two general categories: (economic) efficiency and ergonomics.

Efficiency is often regarded as the ability to attain a desired outcome with a minimum of invested input, e.g., time, resources, or cost (Coelli et al., 2005). Following the Council of Supply Chain Management Professionals' definition, the strive for efficient plans is an integral part of logistics management (CSCMP, 2018). Efficiency is therefore the primary objective for the majority of logistic optimization problems. However, efficiency presents an abstract concept. To serve as the objective of an optimization problem, it needs to be quantifiable. A usual approach is therefore to quantify efficiency as time needed to execute a considered logistic task – the lower the time, the higher the efficiency. Papers 1, 3 and 4 follow that idea.

Paper 1 considers two objectives, where one presents minimizing the order pickers' walking distance to process orders. Paper 4 takes an identical approach. Hence, strictly speaking, both

papers do not consider order processing times directly. However, assuming the pickers' walking speed to be constant, the objectives of minimizing walking distances and processing times relate by a constant factor and can be used interchangeably. Paper 3 considers three different measures of time, namely the makespan, the flow time, and the lateness. Depending on the situation and application, either can be used to quantify efficiency. Beyond that, Paper 3 allows to weight the flow time or lateness of different jobs to account for priority differences.

Paper 5 assumes a divergent notion of efficiency as it aims to minimize the number of required tow trains. The rational behind this is that tow trains and especially their drivers' labor costs are the main cost factors in the operation of tow train systems. Hence, by minimizing the number of required tow trains, the costs are minimized. Considering the objectives of Papers 1, 3, and 4 in a more abstract way, minimizing the processing time can also be interpreted as minimizing costs, as the reduced processing times can be translated into a reduction in labor time and hence labor costs. The relation is less strict, however.

While efficiency is considered frequently in logistic optimization problems, other objectives have been paid less attention. Especially in intra-logistics, many processes require a large proportion of human labor and manual materials handling (MMH), which has repeatedly been shown to increase the workers' likelihood to develop MSD (cf., e.g., Punnett and Wegman, 2004, Larsson et al., 2007), such as lower back pain. Not only do these put workers at discomfort, they also increase error rates, reduce work capacities, and cause lost-time injuries, which result in direct and indirect costs for companies and the public. For example, MSD are assumed to make up 56% of work-related illnesses in the European Union causing costs of 2% of the GDP (Schneider and Irastorza, 2010). Researchers have therefore advocated to incorporate ergonomic factors in intra-logistic planning and optimization models and identified the lack thereof as a major research gap (Grosse et al., 2015, Boysen et al., 2015).

Motivated by this assessment, Papers 1 and 2 aim to minimize physical strains during MMH in two intra-logistic applications, i.e., order picking and (un-)loading tow trains. In both applications, materials (items, parts or bins) need to be manually loaded and/or unloaded from racks, which physically strains workers. The amount of physical strain depends on various factors, including the weight and access frequency of handled materials as well as the height of the shelves they are stored in. As materials have different weights and demand rates, the total strain for handling the materials can be reduced by assigning them cleverly to different shelves and shelf levels.

Other than, e.g., processing times, physical strains are less easily quantified. Researchers have

therefore defined different quantification methods, focusing, i.a., on biomechanical, physiologiocal, and/or psychophysical aspects (Dempsey, 1998). One physiological approach that has been used repeatedly to quantify ergonomic strains in intra-logistic optimization models (cf., e.g., Battini et al., 2016, Calzavara et al., 2018, Otto et al., 2017) is the energy expenditure prediction model by Garg et al. (1982). Its rational is to quantify the physical strain of a task as the physiological energy expenditure during task execution. Based on a regression of experimental data, Garg et al. (1982) provide equations to calculate the energy expenditure for a given task, e.g., extracting materials from a shelf, depending on its characteristics, e.g., the handled weight and shelf height. Due to its widespread use, suitability and usability, the model is also applied in Papers 1 and 2.

I.3. Optimization methodology applied in this dissertation

The papers in this dissertation consider optimization problems with strong combinatorial characteristics, which means that large parts of their solution spaces are discrete. Paper 2 considers a purely combinatorial problem, while the remaining papers also include real-valued decision variables. All papers present novel solution procedures for their respective optimization problem. Even though all solution procedures are individually tailored to the respective optimization problem, they share common concepts and methods, which are discussed in the following.

(Mixed-)integer programming is a frequently applied method to formalize optimization problems (Wolsey, 2007). Besides providing exact formal definitions for problems, it enables the use of generalized solution procedures such as branch-and-bound (Lawler and Wood, 1966) or branch-and-cut (Mitchell, 2002). These procedures have been refined continuously and are readily available through sophisticated off-the-shelf software (Anand et al., 2017). Especially for problems with a small number of variables and constraints, off-the-shelf software performs well.

Given the benefits of (mixed-)integer programming, all five papers in this dissertation apply this methodology. Paper 1 presents a MIP for the purpose of defining the considered problem formally; it is however not used in the solution process. Papers 2, 3, and 5 develop MIPs to compare the performance of off-the-shelf software to the performance of individually developed algorithms, whereat the latter compare favorably. Beyond that, Papers 2, 3, 4, and 5 present (mixed-)integer programs that are used to solve certain parts of the considered problems with off-the-shelf software.

Even though modeling optimization problems as (mixed-)integer programs and solving them with off-the-shelf software presents a versatile solution approach, it has limitations. First, due to its general nature, the approach often falls short in using special problem characteristics to its advantage. Second, some problems cannot be formulated as (mixed-)integer programs in a compact way, especially if non-linear elements are present. Third, for problems containing a large number of constraints and/or variables, the performance of off-the-shelf software drops significantly.

One option to alleviate the shortcomings of (mixed-)integer programming is through problem decomposition. Problem decomposition denotes the general concept of splitting an optimization problem into smaller problems that can be solved individually (Boyd et al., 2007, Chaieb et al., 2015) – and, ideally, much more computationally efficient. Various decomposition methods exist. An often encountered approach is hierarchical decomposition. Here, the original problem is split into a superordinate master problem and subordinate sub-problems. The master problem presents a relaxed version of the original problem. It contains the objective of the original problem (possibly in a relaxed formulation) and its final solution corresponds to the optimal solution to the original problem. However, as the master problem does not contain all information of the original problem, sub-problems need to be solved repeatedly during the master problem's solution process to re-integrate required information.

All papers in this dissertation use problem decomposition in their exact solution approaches. Papers 1 and 2 employ a hierarchical decomposition approach that has been labeled semantic decomposition by Chaieb et al. (2015). The idea is to split the original problem into semantically linked problems that are well-studied and/or easy to solve on their own. Both papers utilize the fact that parts of their considered problems present Linear Assignment Problems, which are solvable in polynomial time (e.g., using the Hungarian method of Kuhn, 1955). Each paper separates all decisions that are not part of the Linear Assignment Problems into an overarching master problem. The master problem is then solved repeatedly calling upon emerging linear assignment sub-problems during the solution process. In the case of Paper 1, the master problem is solved by complete enumeration; in the case of Paper 2, DP is used.

Papers 3 and 4 employ logic-based (also called combinatorial) BD (cf., Hooker and Ottosson, 2003, Codato and Fischetti, 2006), while Paper 5 develops a branch-and-check procedure (cf., Thorsteinsson, 2001, Beck, 2010). Both procedures are closely related and build upon the original ideas of BD. Classical BD presents a formalized approach to decompose and solve MIPs

with a special block structure (for a detailed explanation, see Rahmaniani et al., 2017). It is based on the notion that by fixing a problem's integer variables, the remaining problem only contains continuous variables and becomes efficiently solvable. Therefore, the discrete variables are separated from the original problem into a master problem, which is solved using common mixed-integer programming solution approaches, e.g., branch-and-bound. While solving the master problem, the solution procedure repeatedly fixes the discrete variables temporarily. Whenever the discrete variables are temporarily fixed, they are used to formulate and solve a purely continuous sub-problem. The master problem is then updated with the information from a sub-problem's solution through the separation of cuts from the solution space, i.e., the addition of constraints. Afterwards the solution process continues solving the updated master problem.

Logic-based BD and branch-and-check procedures extend the idea of classical BD by deviating form the latter's formalism. Instead of strictly separating a problem based on its types of variables, problems are split where it seems computationally appropriate – following a similar idea as semantic decomposition. The general concept of temporarily fixing (some of) the integer variables in a mixed-integer master problem, solving the emerging sub-problems and adding the attained information back to the master model via cuts remains, however. One major benefit of logic-based BD and branch-and-check procedures is that they are not limited by a strict formalism. Moreover, they offer greater flexibility for formulating the sub-problems, which do not need to be in the form of linear programs.

The differences between logic-based BD and branch-and-check procedures are subtle and not consistently defined in the literature. According to Beck (2010), in the former, the master problem only forwards an integer solution to a sub-problem when they are considered optimal by the current master problem (which does generally not imply optimality for the original problem, as the master problem might lack critical information that has not yet been added). The latter forwards solutions at every encountered integer solution. Moreover, in the former, the solution process of the master problem is restarted after every added cut, while in the latter the search is continued without a restart (Beck, 2010). Contrarily, Hooker (2011) considers both alternatives to be logic-based BD approaches referring to the latter as dynamic implementation. In application, researchers have also proposed approaches that fall in-between the two options and used the terminology interchangeably to some degree (cf., Beck, 2010, Hooker, 2007, Thorsteinsson, 2001). This is also the case for the papers in this dissertation.

In Papers 3, 4, and 5, the respective master models are solved by of-the-shelf software, namely CPLEX. Sub-problems are solved and, consequently, cuts are separated from the master problem at every integer solution the software considers viable (IBM CPLEX, 2021), which is neither at

every encountered integer solution nor only at optimal solutions. After cuts are separated, the procedure continues to solve the master problem without restarting. In Paper 3, the sub-problem is solved using a DP approach. Paper 4 proposes a novel solution approach employing a gradient descent method to solve the respective non-linear sub-problem. Finally, Paper 5 divides the sub-problem further into two hierarchical connected problems. One is solved through an iterative procedure, while the other is formulated as a (small) integer program that is solved by of-the-shelf software.

Another optimization approach, which is employed by Papers 2, 3, and 4 in this dissertation, is DP. DP describes a general algorithmic concept originally proposed by Bellman (1954). It is suitable to solve optimization problems, where the solution depends on a sequence of linked decisions. DP can be considered as a sequential decomposition approach (Sniedovich, 1991), which decomposes a problem into sub-problems along the linked decisions. Sub-problems are then solved one after the other, where the solutions of preceding sub-problems are built upon to determine the solutions of succeeding ones. As sub-problems only need to be solved once and existing solutions are reused, DP alleviates the computational effort for repeated computations.

Paper 2 uses DP to assign sets of bins to two train wagons, which is related to Bin Packing. The sub-problems consist in finding combinations of sets of bins that are stored on the same wagon. Even though the problem is NP-hard and the DP procedure has an exponential worst-case runtime, it performs well in computational tests on practically relevant instances.

Paper 3 employs DP to solve the sub-problem within the proposed logic-based BD. It consists in the NP-hard problem of scheduling a set of given jobs with time windows on a single machine. Its worst-case runtime is exponential. Nevertheless, it shows a good performance in the conducted computational tests.

While Papers 2 and 3 use DP as exact procedures, Paper 4 proposes a heuristic DP approach. The procedure assumes that the order picker's walking path never crosses itself while processing an order. Though this holds true in many optimal solutions, it is not guaranteed to be optimal. Based on this assumption, the heuristic DP consists in constructing the order picker's route from non-intersecting sub-tours. The procedure has a polynomial runtime and finds close-to-optimal solutions in computational tests.

While all papers in this dissertation develop exact solution approaches, Papers 2, 3, and 4 propose heuristics in addition. Heuristics present rule-based algorithms that aim to find good but not

necessarily optimal solutions to optimization problems. Compared to exact procedures, which guarantee optimal solutions, they usually have much shorter runtimes and/or lower memory requirements (Zanakis and Evans, 1981). There can be many motivations to prefer heuristics to exact procedures (cf., Zanakis and Evans, 1981). One of the primary reasons in logistics is that the runtime of exact procedures exceeds available planning lead times in practical application (Griffis et al., 2012). This is also the rational employed in Papers 2, 3, and 4, which consider NP-hard problems in an operational short-term setting.

Paper 2 proposes a semantic decomposition approach, where the problem is separated into Linear Assignment Problems (which are solvable in polynomial time) and an NP-hard problem related to Bin Packing. In the exact solution approach, the latter is solved through DP. The proposed heuristic replaces the DP procedure by a GRASP. In the computational study, the heuristic is shown to have low runtimes and small optimality gaps.

Paper 3 employs a heuristic column selection procedure based on a set partitioning formulation of the considered parallel machine scheduling problem. The set partitioning formulation assumes all parameters to be integer and generates columns of a generalized set partitioning problem by considering all possibilities to assign jobs to machines at all feasible points in time. The problem is then formulated as a MIP and solved by off-the-shelf software. By restricting the considered columns to a subset of all possibilities, the problem is transformed into a heuristic formulation, which is easier to solve due to a reduced problem size.

As mentioned above, Paper 4 employs a heuristic DP approach based on an assumed property of the optimal solution that does not always hold. Other than the heuristics in Papers 2 and 4, this directly forecloses the possibility to find the optimal solution in some problem instances. Nevertheless, the computational tests demonstrate that this is only rarely the case and optimality gaps are generally small.

I.B. Bibliography

Anand, R., Aggarwal, D., and Kumar, V. (2017). A comparative analysis of optimization solvers. *Journal of Statistics and Management Systems*, 20(4):623–635.

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2011). The traveling salesman problem. In *The Traveling Salesman Problem*. Princeton university press.

Battini, D., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2016). Human energy expenditure in order picking storage assignment: A bi-objective method. *Computers & Industrial Engineering*, 94:147–157.

Beck, J. C. (2010). Checking-up on branch-and-check. In *International Conference on Principles and Practice of Constraint Programming*, pages 84–98. Springer.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

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.

Blanchard, D. (2021). *Supply chain management best practices*. John Wiley & Sons.

Bowersox, D. J., Calantone, R. J., and Rodrigues, A. M. (2003). Estimation of global logistics expenditures using neural networks. *Journal of Business Logistics*, 24(2):21–36.

Boyd, S., Xiao, L., Mutapcic, A., and Mattingley, J. (2007). Notes on decomposition methods. *Notes for EE364B, Stanford University*, 635:1–36.

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. and Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413–422.

Calzavara, M., Glock, C. H., Grosse, E. H., and Sgarbossa, F. (2018). An integrated storage assignment method for manual order picking warehouses considering cost, workload and posture. *International Journal of Production Research*, pages 1–17.

Chaieb, M., Jemai, J., and Mellouli, K. (2015). A hierarchical decomposition framework for modeling combinatorial optimization problems. *Procedia Computer Science*, 60:478–487.

Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

Coelli, T. J., Rao, D. S. P., O'Donnell, C. J., and Battese, G. E. (2005). *An introduction to efficiency and productivity analysis*. Springer Science & Business Media.

Coyle, J. J., Bardi, E. J., Langley, C. J., et al. (1996). *The management of business logistics*, volume 6. West publishing company St Paul, MN.

Da Silveira, G., Borenstein, D., and Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International journal of production economics*, 72(1):1–13.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

Dempsey, P. G. (1998). A critical review of biomechanical, epidemiological, physiological and psychophysical criteria for designing manual materials handling tasks. *Ergonomics*, 41(1):73–88.

Deroussi, L. (2016). *Metaheuristics for logistics*. John Wiley & Sons.

Emde, S., Fliedner, M., and Boysen, N. (2012). Optimally loading tow trains for just-in-time supply of mixed-model assembly lines. *IIE Transactions*, 44(2):121–135.

European Commission. (2022). Logistics. Online: `https://transport.ec.europa.eu/transport-themes/logistics-and-multimodal-transport/logistics_en` [2022-11-15].

Garey, M. and Johnson, D. (1979). *Computers and Intractability – A guide to NP-completeness*. WH Freeman and Company, San Francisco.

Garg, A., Chaffin, D. B., and Freivalds, A. (1982). Biomechanical stresses from manual load lifting: a static vs dynamic evaluation. *IIE Transactions*, 14(4):272–281.

Glock, C. H. and Grosse, E. H. (2012). Storage policies and order picking strategies in U-shaped order-picking systems with a movable base. *International Journal of Production Research*, 50(16):4344–4357.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier.

Griffis, S. E., Bell, J. E., and Closs, D. J. (2012). Metaheuristics in logistics and supply chain management. *Journal of Business Logistics*, 33(2):90–106.

Grosse, E. H., Glock, C. H., Jaber, M. Y., and Neumann, W. P. (2015). Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, 53(3):695–717.

Gudehus, T. and Kotzab, H. (2012). *Comprehensive logistics*. Springer Science & Business Media.

Hooker, J. (2011). *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley & Sons.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55(3):588–602.

Hooker, J. N. and Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60.

IBM CPLEX Optimization Studio. (2021). Cut callback. Online: `https://www.ibm.com/docs/en/icos/12.9.0?topic=legacy-cut-callback` [2022-11-15].

Jasti, N. V. K. and Kodali, R. (2015). Lean production: literature review and trends. *International Journal of Production Research*, 53(3):867–885.

Korte, B. H., Vygen, J., Korte, B., and Vygen, J. (2011). *Combinatorial optimization*, volume 1. Springer.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Larsson, B., Søgaard, K., and Rosendal, L. (2007). Work related neck-shoulder pain: a review on magnitude, risk factors, biochemical characteristics, clinical picture and preventive interventions. *Best Practice & Research Clinical Rheumatology*, 21(3):447–463.

Lawler, E. L. and Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719.

Lieb, C., Klenk, E., Galka, S., and Keuntje, C. (2017). Einsatz von Routenzugsystemen zur Produktionsversorgung – Studie zu Planung, Steuerung und Betrieb. Technical report, Technische Universität München, Garching.

Limère, V., Landeghem, H. V., Goetschalckx, M., Aghezzaf, E.-H., and McGinnis, L. F. (2012). Optimising part feeding in the automotive assembly industry: deciding between kitting and line stocking. *International Journal of Production Research*, 50(15):4046–4060.

Lovász, L. and Plummer, M. D. (2009). *Matching theory*, volume 367. American Mathematical Soc.

Mitchell, J. E. (2002). Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, 1(1):65–77.

Murphy Jr, P. R. and Knemeyer, A. M. (2018). *Contemporary logistics*. Pearson.

Negri, E., Perotti, S., Fumagalli, L., Marchet, G., and Garetti, M. (2017). Modelling internal logistics systems through ontologies. *Computers in Industry*, 88:19–34.

Otto, A., Boysen, N., Scholl, A., and Walter, R. (2017). Ergonomic workplace design in the fast pick area. *OR Spectrum*, 39(4):945–975.

Öztürkoğlu, Ö., Gue, K. R., and Meller, R. D. (2012). Optimal unit-load warehouse designs for single-command operations. *IIE Transactions*, 44(6):459–475.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Pohl, L. M., Meller, R. D., and Gue, K. R. (2009). Optimizing fishbone aisles for dual-command operations in a warehouse. *Naval Research Logistics (NRL)*, 56(5):389–403.

Punnett, L. and Wegman, D. H. (2004). Work-related musculoskeletal disorders: the epidemiologic evidence and the debate. *Journal of Electromyography and Kinesiology*, 14(1):13–23.

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.

Richards, G. (2017). *Warehouse management: a complete guide to improving efficiency and minimizing costs in the modern warehouse*. Kogan Page Publishers.

Satta, G., Parola, F., and Lee, S.-w. (2011). The EU-27 logistics industry: structure and trends of major subsectors and national markets. *KMI International Journal Of Maritime Affairs and Fisheries*, 3(1):1–32.

Schneider, E. and Irastorza, X. (2010). Work-related musculoskeletal disorders in the EU – facts and figures. Technical report, European Agency for Safety and Health at Work, Bilbao, Spain.

Sniedovich, M. (1991). *Dynamic programming*, volume 297. CRC press.

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.

The Council of Supply Chain Management Professionals (2022). Definitions of supply chain management. Online: `https://cscmp.org/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms.aspx` [2022-11-15].

Thorsteinsson, E. S. (2001). Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In *International conference on principles and practice of constraint programming*, pages 16–30. Springer.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons.

Tyulin, A., Chursin, A., et al. (2020). The new economy of the product life cycle. *Springer Books*.

Wolsey, L. A. (2007). Mixed integer programming. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–10.

Yalaoui, A., Chehade, H., Yalaoui, F., and Amodeo, L. (2012). *Optimization of logistics*. John Wiley & Sons.

Yao, A. C.-C. (1980). New algorithms for bin packing. *Journal of the ACM (JACM)*, 27(2):207–227.

Zanakis, S. H. and Evans, J. R. (1981). Heuristic "optimization": Why, when, and how to use it. *Interfaces*, 11(5):84–91.

Paper 1: Ergonomic and economic optimization of layout and item assignment of a U-shaped order picking zone[1]

Abstract: Order picking is considered one of the most labor- and cost-intensive warehouse operating processes. Regularly, order pickers are exposed to severe physical demands, which increase the likelihood for developing muscular-skeletal disorders, such as lower back pain. Muscular-skeletal disorders cause significant compensation, recovery and deficiency costs and are expected to gain further importance due to an aging workforce. Both companies and workers may therefore benefit from decision support models that explicitly take ergonomics aspects into account. The work at hand investigates a warehouses where the storage area is divided into zones with shelves in each zone arranged in the shape of a U. For this warehouse, we determine an optimal configuration of the U-zone's layout as well as an optimal assignment of products to storage locations. We depart from prior research by considering both the minimization of the total travel distance as well as the minimization of the total ergonomic strain workers are exposed to. Both optimization problems are formalized as mixed-integer programs. An exact polynomial-runtime solution procedure, suitable for both objectives, is developed. Using this solution procedure, we illustrate how the relevant ergonomic strains can be quantified to apply them to our optimization model. Computational studies illustrate the efficacy of our proposed solution procedure. Optimal layouts and storage assignments significantly reduce the walking distance and ergonomic strain during order picking. Additionally, both objectives are only marginally conflicting, such that, mutually, an optimal solution for one objective is also a close-to-optimal solution for the other. We finally derive insights on the optimal layout and storage assignment for future research and practical application.

Keywords: Ergonomics; Human factors; Order picking; Storage assignment; Warehouse layout design; Warehouse optimization

[1]This chapter has been published as: Diefenbach, H., and Glock, C. H. (2019). Ergonomic and economic optimization of layout and item assignment of a U-shaped order picking zone. *Computers & Industrial Engineering*, 138, 106094. DOI: `https://doi.org/10.1016/j.cie.2019.106094`
Compared to the original publication, minor correctional changes have been made; they are marked with footnotes.

1.1. Introduction

Order picking, i.e. the process of retrieving items from storage, is one of the most labor-, time- and, hence, cost-intensive warehouse operating processes. It is estimated that order picking accounts for up to $50\% - 75\%$ of warehouse operating costs (Coyle et al., 1996, De Koster et al., 2007, Tompkins et al., 2010, Richards, 2017).

Even though in many areas of logistics, human work is increasingly substituted by fully or partially automated processes, order picking is still performed manually for the biggest part. De Koster et al. (2007), for example, estimate that about 80% of order picking systems are operated by human workers employing the picker-to-parts principle. In such systems, pickers travel through the storage area – either walking or driving small vehicles – to collect items from shelves, pallets or boxes. The main advantage of employing human pickers is their high flexibility, especially their ability to pick arbitrary shaped items, which will most likely not be achieved by automated machines in the foreseeable future (De Koster, 2018).

Due to the cost-intensity of order picking, a lot of research has focused on increasing its efficiency – mainly by means of reducing the pickers' average and total traveling time/distances to retrieve items. Most of this research, which we will discuss in more detail in Section 1.2, focuses on traditional storage area layouts, i.e. straight, parallel shelves, which may be intersected by one or multiple cross aisles. In their literature review on *picker routing*, Masae et al. (2020)* found, for example, that 83% (45 of 54) of the papers included in their core sample consider these kinds of layouts. Besides *picker routing*, commonly regarded decision problems faced in the context of order picking include item *storage assignment* and *layout design* (De Koster et al., 2007), where the latter is primarily concerned with determining the number, length, width, orientation etc. of shelves and aisles.

Even though the warehouse layout mentioned earlier has attracted the most attention in the literature so far, in practice, other layouts can be beneficial as well. Motivated from a practical case described by Glock and Grosse (2012), we consider a storage area that is used to store parts for small-lot delivery to nearby production stations. Stations only demand items from certain mutually exclusive sets of parts. The storage area is therefore separated into zones, where each zone contains the complete set of items demanded by one or (if the sets are small) multiple stations. Within each zone, items are stored in shelves arranged in the shape of a U. Shelves themselves are built from pallet cages (PCs), i.e. boxes generally based on the ground area of

*In the original version, this referred to an unpublished result by Masae et. al (2019). The reference has been updated in this version, as it is now publicly available.

a europallet with a height of about one meter, which can be stacked on top of each other and accessed via a frontal opening. Each type of item is stored in a separate PC. Once all items stowed in a PC have been picked, it is interchanged with a fresh PC by a forklift truck. Lastly, the peculiarity that PCs can be stacked without the need for shelves offers the possibility to place item-containing PCs on top of empty PCs (if the latter are available). This elevates the former to an easier accessible height, reducing the time and physical effort required for extracting items (cf., Petersen et al., 2005). While picker *routing* is only of minor concern in U-shaped storage areas (cf. Section 1.3, Glock et al., 2019), picking efficiency is strongly dependent on *storage assignment* and *layout design*. The latter is, in contrast to a traditional *layout design* problem, concerned with the length and width of a U-zone restricted by the spacial limits of the available area.

In practice, PCs are often used in industrial settings to store heavy, bulky items. As order picking is a highly manual and labor-intensive task, pickers are frequently exposed to significant physical demands, especially if items are heavy. Physical demands in the course of manual materials handling (MMH) – such as the lifting, carrying and lowering of items encountered in order picking – have been linked to an increase of workers' risks of developing muscular-skeletal disorders (MSD) (cf. Section 1.2). Despite their severity, the scientific warehousing literature started to consider ergonomic aspects only in very recent years (cf. Section 1.2). Furthermore, those concerns are expected to further gain in importance in the future due to an aging workforce (e.g., Otto and Scholl, 2011, Aiyar et al., 2017, European Commission, 2017).

Among environmental and worker-specific factors, physical strains during MMH activities are strongly dependent on task characteristics (cf. Mital, 2017). For example, the strain for extracting an item from storage is, i.a., influenced by the height at which the item is stored. The strain for carrying a certain item is, i.a., influenced by the carrying distance (cf. Section 1.3.4). Both, carrying distances and storage heights are themselves dependent on the *layout design* of the storage area and the items' *storage assignment*. The scope of this paper is therefore threefold. First, we model and solve an optimization problem regarding the layout (i.e., the width and length) of a U-shaped storage zone with the (traditional) objective of minimizing total walking distances for picking items stowed in the zone. Secondly, we solve the same optimization problem by minimizing the total ergonomic strain for picking. Thirdly, we compare the results of both optimization objectives and derive some recommendations for practical application and further research.

1.2. Literature review

The planning and operation of warehouses, especially in the context of order picking, has been intensely studied in the scientific literature. Ranging from strategic to operational, five planing and decision problems are commonly distinguished: *layout design*, *storage assignment*, *zoning* (i.e., the division of the warehouse or storage area into distinctive zones), *batching* (i.e., the accumulation of small orders into larger batches), and, finally, *routing* of the picker through the storage area. With regard to the problem studied in this paper, we focus on research on *layout design* and *storage assignment* in the following. For further readings on *zoning*, *batching* and *routing* problems, we refer to the comprehensive reviews of Rouwenhorst et al. (2000), De Koster et al. (2007) and Gu et al. (2007, 2010).

Layout design problems can focus on varying levels of aggregation. The perspective can range from planning warehouse locations in facility layouts (cf., Kusiak and Heragu, 1987), through deciding on the layout of departments within warehouses (e.g., Heragu et al., 2005), to the determination of the number, arrangement, spacial orientation etc. of shelves within the storage area. The latter type of problems are obviously strongly interdependent with tactical and operational order picking problems, and here in particular *storage assignment*, *zoning* and order picker *routing*. Most research on *layout design* focuses on automated warehouses (De Koster et al., 2007). Fewer authors are concerned with the layout optimization in manually operated systems. Caron et al. (2000), for example, analyze the pickers' expected traveling distances depending on the number, length and spacial arrangement of shelves for different common routing policies. The authors further assume items to be randomly assigned to storage positions. Roodbergen et al. (2008) consider a similar problem, for which they develop an optimization approach that also includes the determination of the optimal position of a base, where items are dropped off. Shelf heights, i.e. the layout in the vertical direction, are rarely considered. An exception is the paper of Calzavara et al. (2017) that evaluates different shelf heights for the storage of pallets from an economic as well as an ergonomic perspective.

Most research on the *layout design* of warehouses assumes a conventional aisle arrangement consisting of parallel picking aisles that are possibly intersected by perpendicular cross aisles. A few exceptions are discussed in the following. Dukic and Opetuk (2008), Pohl et al. (2009) and Cardona et al. (2012), for example, studied the optimization of traveling distances in a warehouse with a so-called fishbone layout, where aisles intersect parallel or perpendicularly arranged shelves in a straight or bend V-shape, resulting in the aisles resembling a fishbone-like

shape. Henn et al. (2013)[2] analyze traveling distances in a warehouse consisting of shelves arranged in narrow aisles that are intersected by broad aisles forming an H that spans the whole length of the warehouse. While these layouts are still based on zones of mostly parallel aisles, Glock and Grosse (2012) consider warehouse layouts consisting of multiple zones, each containing shelves arranged in the shape of a U. To the best knowledge of the authors, Glock and Grosse (2012) are the first to study this kind of layout. Their objective was to analyze total picking times for different U-layouts (width and length) in combination with different *storage assignment* policies via numerical simulations. However, the layout of the U-zones was not optimized in their work. Similar U-shaped layouts are considered by Grosse and Glock (2013), who investigate performance improvements due to picker learning, and Glock et al. (2019), who studied the option of rotating half-empty pallets to ease item extraction. The latter work is described in more detail later in this section.

Storage assignment is concerned with assigning items to storage locations within the storage area. In the literature, different policies – i.e., general concepts – for storage assignment are considered. Among the most common are: random storage, dedicated storage and class-based storage (De Koster et al., 2007, Gu et al., 2007, 2010). If a random storage policy is used, items are assigned randomly to (empty) storage locations in the warehouse without taking their characteristics, such as pick frequencies, weight or pick correlations with other items, into account. Furthermore, locations are subject to change over time when items are replenished. While, generally, random storage policies lead to higher average travel times and distances (Hausman et al., 1976), their major advantage is a better spacial utilization compared to other policies (De Koster et al., 2007). Closely related to random assignment is the policy of closest open location assignment, where items are not assigned to random positions, but instead to an empty one closest to the depot (e.g., Hausman et al., 1976). From a human factors point of view, random and closest open location assignment policies have a disadvantage as they can confuse workers and hinder learning effects (cf., Grosse et al., 2013, Grosse and Glock, 2015).

Dedicated assignment policies assign items to fixed positions in the storage area, usually based on item characteristics. Most commonly, item pick frequencies are used to determine storage locations; the higher the pick frequency, the closer the items should be placed to the depot to minimize travel distances (e.g. Rao and Adil, 2013). Alternatively, the cube-per-order index (i.e., an item's pick frequency divided by its spacial requirement) is often used to determine its storage location (e.g., Kallina and Lynn, 1976). Other approaches additionally account for pick correlations of items and emphasize placing those items frequently picked together in close

[2]The authors denote the layout considered in their paper as a "U-shaped layout". However, to avoid confusion, we point out that their definition of a U-shaped layout differs from the one made in this paper.

proximity (e.g., Chuang et al., 2012, Glock and Grosse, 2012). While research traditionally focused on assigning items in the planar dimension with the objective of minimizing traveling distances or times, recent works also consider the assignment of items to different shelf levels as a decision variable. Petersen et al. (2005), for example, study the influence of assigning items to shelves of different height on item retrieval (pick) times. They find that picking is most efficient – and presumably less physically exhausting – if items are stored in a "golden zone" between the picker's hip and shoulder height. Despite the abundance of research on dedicated item assignment policies, the objective is almost exclusively the minimization of travel distances and item retrieval times; ergonomic aspects have only very infrequently been discussed. A few notable exceptions are discussed in more detail in the following (cf., Grosse et al., 2015, Reyes et al., 2019).

Battini et al. (2016b) analyze the optimal allocation of items to storage locations in a single elongated shelf with the objective to minimize both total picking times and ergonomic strains. The latter are quantified via an energy expenditure prediction model proposed by Garg et al. (1978) (cf. Section 1.3.4). A similar bi-objective problem is considered by Larco et al. (2017), who optimize *storage assignments* by minimizing pick cycle times and picker discomfort in a real-word case study. For the evaluation of (ergonomic) discomfort, the authors utilize the CR-10 scale of Borg (1982). Pickers' ergonomic strains when extracting items from pallets assigned to racks of different height are analyzed by Calzavara et al. (2018) using the approach proposed by Garg et al. (1978). Otto et al. (2017) consider a combined *storage assignment* and *zoning* problem in a fast pick area with the objective of minimizing the pickers' ergonomic strains, which they solve via a tabu search heuristic. The authors quantify ergonomic strain using two different approaches, namely the NIOSH lifting equation (Waters et al., 1993) as well as a predetermined motion energy system (cf. Battini et al., 2016a) that is based on the energy expenditure prediction model by Garg et al. (1978). Pan et al. (2015) consider and heuristically solve a *storage assignment* problem in a pick-and-pass system with the objective of balancing the workload between pickers. For their model, the authors only consider pick frequencies, but neglect item weights. Hence, they do not quantify ergonomic strains explicitly. Fontana and Nepomuceno (2017) develop a multi-criteria methodology for *storage assignment* that, among other objectives, also considers ergonomics aspects in a rudimentary manner. Their methodology thus assigns items closer to the ground the heavier they are. Finally, Glock et al. (2019) consider the problem of picking items stored on pallets in a U-shaped storage zone with the option to rotate pallets by 180° once the items on the frontal half of the pallet have been picked. Rotating pallets then avoids that items have to be picked from the pallet's back side, which reduces load on the order picker during item retrievals. The authors optimize the regular pick and the pallet rota-

tion process of this system with respect to both economic and ergonomic indicators and analyze the influence of different storage assignment strategies via computational studies. Ergonomic strains are quantified by calculating the compressive forces acting on the L4/L5 intervertebral joint during the lifting activity, which are computed by modelling the workers' body postures in the biomechanical modelling software 4DWATBAK (Neumann et al., 1999).

Lastly, class-based storage policies aim to combine the advantage of random and dedicated storage assignment policies by grouping items into classes – for example, according to pick frequency or by article similarity – and by assigning each group to a dedicated zone in the warehouse. Within each zone dedicated to a certain class, items of the respective class are assigned randomly to the available locations. Class-based storage policies are, for example, recently considered by Qin et al. (2015), Sharma and Shah (2015) and Ene et al. (2016). For further readings on *storage assignment*, we refer to the recent survey of Reyes et al. (2019).

While this brief review shows that in recent years, researchers have started to integrate ergonomic aspects into decision support models for order picking activities, there are still various research gaps in this area (cf., Grosse et al., 2015, 2017, Neumann and Medbo, 2010). In particular, optimization approaches that take account of (different types of) ergonomic objectives are still missing for different warehouse layouts and order picking decision problems. This is in sharp contrast to the literature on the impact of MMH activities on worker health and safety that have frequently been studied in the human factors engineering literature. MMH activities, such as the lifting, lowering and carrying of items encountered in most order picking systems (cf., De Koster et al., 2007), have repeatedly been shown to be one of the worker's primary risk factors for developing MSD (Larsson et al., 2007, Punnett and Wegman, 2004, Roquelaure et al., 2009). Lower back pain, for example, which has one of the highest MSD incident rates, is estimated to be caused by MMH activities in $50\% - 75\%$ of all cases (Bigos et al., 1986, Snook, 1989, Spengler et al., 1986).

Besides the apparent disadvantages for workers affected by MSD, MSD cause immense direct and indirect costs for the private as well as for the public sector. It is estimated that in the USA, for example, the annual direct cost resulting from MSD amount to $\$14.2$ billion, while the total annual compensatory costs are estimated at $\$58.5$ billion (LMRIS, 2018, NRC, 2011). Schneider and Irastorza (2010) estimate that in the EU, MSD account for 58% of work-related illnesses, causing costs that reach 2% of the European Union's gross national product. Additionally, various studies confirm the general health risks associated with MMH activities to be also particularly relevant in the order picking context (Braam et al., 1996, Gardner et al., 1999,

Garg, 2000, Marras et al., 2010, Lavender et al., 2012). This further amplifies the importance of integrating ergonomic aspects into order picking decision support models, which is the purpose of this paper.

In this context, this paper makes the following two contributions. Firstly, we extend the previous research of Glock and Grosse (2012) on the *layout design* and *storage assignment* of U-zones. While Glock and Grosse (2012) only consider different heuristic policies, this paper provides an exact solution procedure that yields guaranteed optimal solutions in polynomial runtime. As a numerical experiment in Section 1.5.2 shows, optimal solutions can yield significant improvements over heuristic solutions. Secondly, besides the classical objective of increasing efficiency, this paper also addresses the objective of minimizing the picker's total ergonomic strain. Our literature review emphasizes the importance of considering ergonomic aspects in the design and operation of order picking systems. Nevertheless, research on this topic is still scarce in the general context of order picking, and there is no research at all so far that investigates the ergonomic design of U-shaped order picking zones. This paper intends to contribute to closing this research gap.

1.3. Problem description

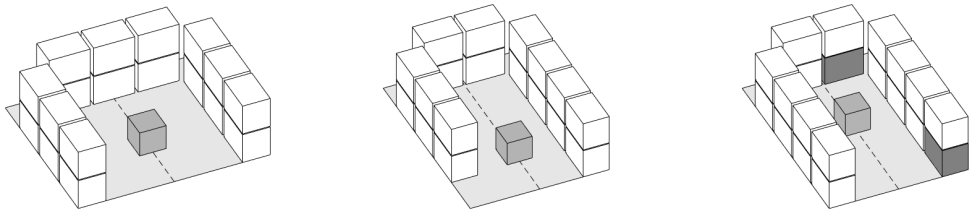
Storage areas employing U-shaped layouts usually consist of multiple zones, each with shelves arranged in the shape of a U (cf., Glock and Grosse, 2012). In such systems, planners have to decide on the number and sizes of zones, the set of items that should be stowed in each zone, the layout of each zone, and, finally, the assignment of items to the storage locations available in each zone. Clearly, these decisions are interdependent. A holistic model that considers all decision problems simultaneously, however, is prone to not being optimally solvable due to its complexity. (Meta-)heuristics are a commonly used alternative in such situations. They do, however, not guarantee good quality solutions, especially if the properties of the problem and its optimal solutions are not well understood. We, therefore, employ a different approach in this paper by assuming that the sets of items that should be stored in a certain zone have already been determined, which is reasonable for many practical cases, such as the one that motivated this paper. We further allow for the zone size (i.e., its planar dimension) to be either predetermined, restricting the width and length of possible U-layouts, or to be subject to optimization. We are then able to optimize both the U-layout and the assignment of items to storage locations for each zone. In future research, the results on optimal U-layouts and storage assignments obtained in

this paper could then be used to develop well-suited heuristics.

The problem considered in this paper can be described as follows: The U-shaped storage layout planning problem (USLPP) consists of determining the layout of a U-shaped storage zone, i.e., its length and width, and the assignment of a given set of items to stowage locations (i.e., PCs). Further, we also determine the position of the depot (or base; depicted in medium grey in Figure 1.1), which, for example, may represent a picker cart where retrieved items are dropped off. In the scenario considered in this paper, the picker is assumed to travel between the stationary base and the items' stowage locations to complete orders, while picking only one item at a time due to the items' weights or dimensions. Therefore, routing the order picker to the storage locations is independent of the pick sequence, and consequently it is not necessary to solve a routing problem. Further, we acknowledge that between orders, the base is replaced by a new (empty) base and consequently moved to ship retrieved items to their destination, e.g. using a forklift truck. However, while processing an order, we assume that the base is kept stationary.

The storage area is built from equal-sized PCs labeled with indices $i \in I$, each containing exactly one kind of item of equal index. Each PC can either be placed on the floor or on top of another PC, where the maximum stack height is limited to two PCs due to the maximum height a human picker is able to reach and to respect safety regulations. From a top-down view, the PCs must be arranged in the shape of a U of arbitrary dimension (i.e., length and width) but minimal sufficient size, such that all PCs $i \in I$ still fit in. We define a U-zone to be of minimal sufficient size if it contains the maximum amount of stacks (i.e., the minimal amount of non-stacked PCs, which is 0 if $|I|$ is even and 1 if $|I|$ is odd), and if it does not contain any gaps within its U-structure where an additional stack of PCs would fit in. However, we do allow for unsymmetrical U-layouts with a gap of a single PC's size at the U-shape's opening (cf. example (b) in Figure 1.1). We further allow empty PCs (depicted in dark grey in Figure 1.1) – which are abundantly available in a lot of practical cases – to be placed on the ground to support item-containing PCs on their top, as long as this does not necessitate an increase in the area of the rectangular shape spanned by a U-zone of minimal sufficient size (depicted in light grey in Figure 1.1). The underlying idea is that it can be more efficient and less stressful to pick items from an upper than from a lower PC (cf., Petersen et al., 2005). Additionally, spacial restrictions may be added (but do not necessarily have to), limiting the maximal length and/or width of a zone. In-between two adjacent PCs, we assume that a small gap d^s exists to facilitate handling the PCs by a forklift truck. The replenishment of empty PCs is not explicitly considered in the proposed model, as this is usually done using a forklift truck and does not significantly interfere with the order picking process. The base has to be placed on the floor within the U-zone. We assume that it is always placed centered on the vertical symmetry line of the U, but that it may

be placed closer to or further away from to the U-shape's opening. In addition, we demand that the base is at least a distance d^p away from the nearest PC, such that the picker is always able to circumvent the base to reach items from PCs close to the base. Three exemplary arrangement for $|I| = n = 18$ PCs, which are feasible according to the requirements formulated above, are schematically depicted in Figure 1.1.



(a) (b) (c)

Figure 1.1.: Exemplary (feasible) arrangements of PCs in a U-zone for $n = 18$.

We investigate the described problem on an aggregated level, not considering individual orders, but accumulated demands over the course of a longer period. We assume that every type of item $i \in I$ has a known pick frequency f_i that describes how often an item of type i is picked on average. Such assumptions have frequently been made in the literature on storage assignment problems (e.g, Hausman et al., 1976, Jarvis and McDowell, 1991, Petersen et al., 2005). We acknowledge that in practice, pick frequencies may not be known in advance. However, often they can be calculated with sufficient precision from past data or using prediction models (cf., Kallina and Lynn, 1976, Kovács, 2011, Sadiq et al., 1996). Apart from this, we assume that the weight w_i of each type of item $i \in I$ is known.

Picking an item results in an ergonomic strain for the picker. For each item $i \in I$ that is picked, we differentiate between four distinct tasks causing ergonomic strains, which accumulate to the total strain to retrieve the respective item: Firstly, the picker walks from the base to the PC that contains the requested item. Secondly, the picker extracts the item from the PC and brings it into an easy to carry position. Thirdly, the picker carries the item back to the base. Fourthly, the picker deposits the item at the base. For additional clarification, the picking task is schematically illustrated in Figure 1.4 in the Appendix. The ergonomic strains resulting from the tasks depend on the distance between the base and the item's storage location (first and third task), the weight of the item (second, third and fourth task) as well as the storage of the item on the upper or lower PC (second task). Further, the strain has to be accounted for each time an item i is picked, and, therefore, further depends on the item's pick frequency f_i. Any other strains that may occur,

such as, for example, strains that result from moving the base with a forklift truck, are neglected due to their low relative impact on the total ergonomic strain. The calculation of ergonomic strains is described in more detail in Section 1.3.4.

Lastly, we assume that the picker travels from the base to the storage locations and back on a path that resembles a sigmoid curve, which is in-between a rectangular and an euclidean path. This has been observed to closely resemble actual human walking paths in warehouses (Goetschalckx and Ratliff, 1988). A detailed explanation of how traveling distances are calculated is given in Section 1.3.2.

A feasible arrangement of PCs into a U-shaped zone of minimal sufficient size already ensures an efficient utilization of space. However, it does not guarantee that items are picked either efficiently or ergonomically. Therefore, among all feasible arrangements, we seek the one that either minimizes the picker's total walking distance, or the one that minimizes the picker's total ergonomic strain for picking all stowed items with their respective pick frequencies. In the following, we refer to the former problem as the economic USLPP (ec-USLPP) and to the latter problem as the ergonomic USLPP (er-USLPP)

1.3.1. Notation for modeling USLPP

In formulating mathematical models and the solution procedure in Section 1.4, for both versions of USLPP, we use the notation introduced in Table 1.1.

We model USLPP in a discrete (p, q)-coordinate system, as illustrated in Figure 1.2 for case (a) of the exemplary arrangement in Figure 1.1. The (p, q)-plane represents a top-down view of the U-shaped zone, where PCs can be placed at each discrete (p, q)-position. To define the boundaries of the U-layout, we introduce two "dummy" PCs $j \in \{1, 2\}$. Both "dummy" PCs do not represent physical PCs, but are used for mathematical modelling only. We allow actual PCs only to be in the same p- or q-row as the "dummy" PCs – either on the ground or on top of another PC. Hence, it is only necessary to decide on the placement of the "dummy" PCs to determine the U's layout.

I	set of PCs/ types of items (index i); $\|I\| = n$
P	set of discrete horizontal positions (index p); $P' = P \setminus \{1, 2\}$
Q	set of discrete vertical positions (index q); $Q' = Q \setminus \{1\}$
D	set of all possible walking distances for a given U-shaped storage layout and base position
K	index set (index k) of all (p, q)-positions of a given U-shaped storage layout
S	solution to USLPP
$y_{p,q,j}$	binary variable: 1, if "dummy" PC $j \in \{1, 2\}$ is placed at location (p, q); 0 otherwise
$x^b_{p,q,i}$	binary variable: 1, if box i is placed at location (p, q) at the bottom; 0 otherwise
$x^t_{p,q,i}$	binary variable: 1, if box i is placed at location (p, q) at the top; 0 otherwise
r[†]	continuous variable denoting the base's position in q-direction
$d_{p,q}$	continuous variable: traveling distance from the base to location (p,q)
$d^p_{p,q}$	continuous variable: distance from the base to location (p,q) in p-direction
$d^q_{p,q}$	continuous variable: distance from the base to location (p,q) in q-direction
Δ_q	discrete length of a given U-zone in q-direction (i.e., the number of q-positions)
Δ_p	discrete length of a given U-zone in p-direction (i.e., the number of p-positions)
$w^{w,c}_i$	weighting factor (i.e., ergonomic strain) for walking without load per distance walked and, additionally, for carrying an item of type i per distance walked
w^t_i	weighting factor (i.e., ergonomic strain) for picking an item from PC i, if PC i is located in the top row
w^b_i	weighting factor (i.e., ergonomic strain) for picking an item from PC i, if PC i is located in the bottom row
w^d_i	weighting factor (i.e., ergonomic strain) for dropping off an item of type i at the base
f_i	item picking frequency of items of type i
l^f	frontal length of a PC
l^d	depth of a PC
l^h	height of a PC
b^p	length of the base in p-direction
b^q	length of the base in q-direction
b^h	height of the base
s	safety distance/ gap between two adjacent PCs
d^s	minimal allowed distance between the base and the nearest PC
a^p	maximal length of the U in p-direction
a^q	maximal length of the U in q-direction
δ_r	small increment of the base position

Table 1.1.: Notation.

[†] In the originally published version of this paper, occasionally, the continuous variable denoting the base's position in q-direction is mistakenly labeled n, which is however already used to denote the number of items. Hence, in this version, the variable is continuously labeled r for clearance.

The size of the index sets P and Q (and therefore the maximal extension of the U-zone) can be bounded by either the number of discrete positions a feasible U-shape (that is of minimal sufficient size) can at most occupy in the respective direction, or by the maximum available space in the respective direction; whichever is more restrictive. Since, per definition, U-shaped zones of minimal sufficient size are not allowed to contain any gaps, a feasible layout is maximally extended in q-direction if it is as narrow as possible in p-direction (and vice versa). In p-direction, any layout must at least occupy three discrete positions; otherwise, the zone would not be U-shaped. In such a layout, in total two PCs can be placed at the p-row (one at the bottom and one at the top position at $(p, q) = (2, |Q|)$). All other $n - 2$ PCs must be contained in one of the two q-rows, which, together, can contain four PCs per discrete q-position, except for the position at $q = |Q|$, which is occupied by the "dummy" bins. Hence, the maximal necessary extension of the U-layout in q-direction is $\left\lceil \frac{n-2}{4} \right\rceil + 1$ discrete positions. The extension of the U-zone may further be limited by the available space. In q-direction, each regular PC contained in a q-row takes up a distance of $l^f + s$ (i.e., the PC's frontal length and the safety distance between two PCs) and the PCs contained in the p-row take up a distance of l^d. Hence, if the space in q-direction is limited by a^q, the term $(|Q| - 1)(l^f + s) + l^d \leq l^d$ gives an upper bound for $|Q|$. A similar reasoning can be applied to attain upper bounds for $|P|$. In conclusion, $|P|$ and $|Q|$ are bounded by

$$|P| = \min \left\{ \left\lceil \frac{1}{2} n \right\rceil + 1, \left\lfloor \frac{a^q - 2l^d - s}{l^f + s} \right\rfloor + 2 \right\}$$

and

$$|Q| = \min \left\{ \left\lceil \frac{n-2}{4} \right\rceil + 1, \left\lfloor \frac{a^p - l^d}{l^f + s} \right\rfloor + 1 \right\}.$$

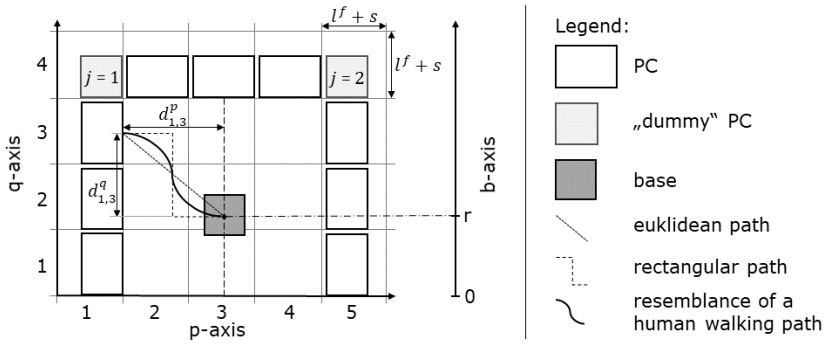


Figure 1.2.: Discrete (p, q)-coordinate system the mathematical model is based on.

1.3.2. Calculating traveling distances

Figure 1.2 shows an example of a human walking path between the base and a PC located at $(p, q) = (1, 3)$ as well as its euclidean and rectangular approximation. In accordance with Goetschalckx and Ratliff (1988), we assume that the picker's walking path is in-between the euclidean and the rectangular distance. We assume every path starts at the center of the base and ends centered in front of the targeted PC. We acknowledge that this is an approximation, since the actual path does not start at the center of the base, but in front of one of its sides. However, the approximation error is acceptably small and about equal for every targeted PC location, and should thus not significantly distort the fundamental results of our optimization. To calculate the traveling distance, we calculate the distance between the start and end points of the path in p- and in q-direction ($d^p_{p,q}$ and $d^q_{p,q}$). The traveling distance then follows as

$$d_{p,q} = \alpha \sqrt{\left(d^p_{p,q}\right)^2 + \left(d^q_{p,q}\right)^2} + (1-\alpha)\left(d^p_{p,q} + d^p_{p,q}\right), \tag{1.1}$$

where $0 \leq \alpha \leq 1$ is an adjustable parameter, with the left term representing the euclidean and the right term in brackets representing the rectangular distance. For the remainder of this paper, we set $\alpha = 0.5$. However, our model and solution procedure work for arbitrary values $0 \leq \alpha \leq 1$. Further, the term $d^p_{p,q}$ depends on the layout of the U-zone (defined by $y_{p,q,j}$, $j \in \{1, 2\}$ and $r^\dagger$), and it is given as

$$\begin{aligned} d^p_{p,q} = \Bigg| & \left(p\,(l^f + s) - \frac{1}{2}\, l^f\, y_{1,q,1} - \sum_{q' \in Q} l^f\, y_{p,q',2} \right) \\ & - \frac{1}{2}\left((l^f + s) \sum_{p' \in P} \sum_{q' \in Q} p'\, y_{p',q',2} + s \right) \Bigg|, \end{aligned} \tag{1.2}$$

with the first term denoting the position of the path's end and the second term denoting the base's center in p-direction. Obeying the same principle, the term $d^q_{p,q}$ is given as

$$d^q_{p,q} = \left| \left((l^f + s)\,(q-1) + \frac{1}{2}\, l^f\, (1 - y_{1,q,1}) \right) - r \right|. \tag{1.3}$$

1.3.3. MIP model for layout planning in a U-shaped warehouse zone

With the notation summarized in Table 1.1, we formalize both variants of the problem as the following MIP models:

$$\text{[ec-USLPP] Minimize} \sum_{p\in P}\sum_{q\in Q}\sum_{i\in I} f_i\, d_{p,q}\, \left(x^b_{p,q,i} + x^t_{p,q,i}\right) \tag{1.4}$$

or

$$\begin{aligned}\text{[er-USLPP] Minimize} \sum_{p\in P}\sum_{q\in Q}\sum_{i\in I} f_i \Big(& d_{p,q}\, w^{w,c}_i\, \left(x^b_{p,q,i} + x^t_{p,q,i}\right) \\ & + \left(w^t_i\, x^t_{p,q,i} + w^b_i\, x_{p,q,i}\right) + \frac{w^d_i}{|P|\,|Q|}\Big)^{\ddagger}\end{aligned} \tag{1.5}$$

subject to

$$\begin{aligned} d^p_{p,q} \geq & \left(p\,(l^f + s) - \frac{1}{2}\, l^f\, y_{1,q,1} - \sum_{q'\in Q} l^f\, y_{p,q',2}\right) \\ & - \frac{1}{2}\left((l^f + s)\sum_{p'\in P}\sum_{q'\in Q} p'\, y_{p',q',2} + s\right) \qquad \forall p\in P; q\in Q \end{aligned} \tag{1.6}$$

$$\begin{aligned} d^p_{p,q} \geq & -\left(p\,(l^f + s) - \frac{1}{2}\, l^f\, y_{1,q,1} - \sum_{q'\in Q} l^f\, y_{p,q',2}\right) \\ & + \frac{1}{2}\left((l^f + s)\sum_{p'\in P}\sum_{q'\in Q} p'\, y_{p',q',2} + s\right) \qquad \forall p\in P; q\in Q \end{aligned} \tag{1.7}$$

$$d^q_{p,q} \geq \left((l^f + s)\,(q-1) + \frac{1}{2}\, l^f\,(1 - y_{1,q,1})\right) - r \qquad \forall p\in P; q\in Q \tag{1.8}$$

$$d^q_{p,q} \geq -\left((l^f + s)\,(q-1) + \frac{1}{2}\, l^f\,(1 - y_{1,q,1})\right) + r \qquad \forall p\in P; q\in Q \tag{1.9}$$

$$\sum_{p\in P}\sum_{q\in Q} y_{p,q,j} = 1 \qquad \forall j\in\{1,2\} \tag{1.10}$$

$$\sum_{q\in Q'} y_{1,q,1} = 1 \tag{1.11}$$

[‡] In the originally published version of this paper, the final term of Equation 1.5 is incorrectly stated as w^d_i. It has been corrected to $\frac{w^d_i}{|P|\,|Q|}$ in this version.

$$\sum_{p\in P'} y_{p,q,2} = y_{1,q,1} \qquad \forall q \in Q \quad (1.12)$$

$$n + 3 \geq \sum_{p\in P}\sum_{q\in Q} (4(q-1) + 2(p-2))\, y_{p,q,2} \quad (1.13)$$

$$n \leq \sum_{p\in P}\sum_{q\in Q} (4(q-1) + 2(p-2))\, y_{p,q,2} \quad (1.14)$$

$$\sum_{p'\in P}\sum_{q'\in Q} p'\, y_{p',q',2} \geq \sum_{i\in I} p\, x^b_{p,q,i} \qquad \forall p \in P; q \in Q \quad (1.15)$$

$$\sum_{p'\in P}\sum_{q'\in Q} p'\, y_{p',q',2} \geq \sum_{i\in I} p\, x^t_{p,q,i} \qquad \forall p \in P; q \in Q \quad (1.16)$$

$$\sum_{p'\in P}\sum_{q'\in Q} q'\, y_{p',q',2} \geq \sum_{i\in I} q\, x^b_{p,q,i} \qquad \forall p \in P; q \in Q \quad (1.17)$$

$$\sum_{p'\in P}\sum_{q'\in Q} q'\, y_{p',q',2} \geq \sum_{i\in I} q\, x^t_{p,q,i} \qquad \forall p \in P; q \in Q \quad (1.18)$$

$$\sum_{i\in I} \left(x^b_{p,q,i} + x^t_{p,q,i}\right) \leq 2 \left(\sum_{p'\in P\setminus\{p\}} y_{p',q,2} + \sum_{q'\in Q\setminus\{q\}} y_{p,q',2} \right) \qquad \forall p \in P \setminus \{1\}; q \in Q \quad (1.19)$$

$$\sum_{j\in\{1,2\}} y_{p,q,j} + \sum_{i\in I} x^b_{p,q,i} \leq 1 \qquad \forall p \in P, q \in Q \quad (1.20)$$

$$\sum_{j\in\{1,2\}} y_{p,q,j} + \sum_{i\in I} x^t_{p,q,i} \leq 1 \qquad \forall p \in P, q \in Q \quad (1.21)$$

$$\sum_{p\in P}\sum_{q\in Q} x^b_{p,q,i} + x^t_{p,q,i} = 1 \qquad \forall i \in I \quad (1.22)$$

$$r + \frac{1}{2} b^q + d^s \leq \sum_{q\in Q} (l^f + s)(q-1)\, y_{1,q,1} \quad (1.23)$$

$$b^p + 2d^s \leq \sum_{p\in P}\sum_{q\in Q} (l^f + s)(p-2)\, y_{p,q,2} + s \quad (1.24)$$

$$x^b_{p,q,i},\, x^t_{p,q,i} \in \{0,1\} \qquad \forall p \in P; q \in Q; i \in I \quad (1.25)$$

$$y_{p,q,j} \in \{0,1\} \qquad \forall p \in P; q \in Q; j \in \{1,2\} \quad (1.26)$$

$$r \in \mathbb{R} : r \geq \frac{1}{2} d^q \quad (1.27)$$

$$d_{p,q}, d^p_{p,q}, d^q_{p,q} \in \mathbb{R}^+ \qquad \forall p \in P, q \in Q \quad (1.28)$$

as well as Equation (1.1), $\forall p \in P; q \in Q$.

Objective function (1.4) minimizes the sum of the total walking distances, whereas (1.5) minimizes the sum of the total ergonomic strain that results from picking all items $i \in I$ with their respective frequencies f_i. The total ergonomic strain to pick an item of type i is the sum of the

strain for walking to the item's stowage location and carrying it back to the base ($d_{p,q}\, w_i^{w,c}$), the strain for extracting the item from a PC placed at the top (w_i^t) or bottom (w_i^b) position and for dropping the item off at the base (w_i^b). From a mathematical viewpoint, ec-USLPP is a special case version of er-USLPP, where the ergonomic weighting factors are set $w_i^{w,c} = 1$ and $w_i^t = w_i^b = w_i^d = 0, \forall i \in I$.

Constraints (1.6) and (1.7) as well as (1.8) and (1.9) are linearized versions of Equations (1.2) and (1.3). In conjunction with Equation (1.1), $\forall p \in P; q \in Q$, they define the walking distance $d_{p,q}$, $\forall p \in P; q \in Q$. Constraints (1.10) enforce each "dummy" PC to be placed at exactly one position. Constraint (1.11) ensures that the first "dummy" PC is always placed in the first p-row, while Constraints (1.12) force both "dummy" PCs to be placed in the same q-row, such that they define a valid U-shape. Constraints (1.13) and (1.14) define the realm of allowed U-layouts, such that the U is big enough to contain all PCs (Constraint (1.14)), while still being of minimal sufficient size (Constraint (1.13)). The right side of Constraint (1.13) describes the number of discrete positions available in the U-layout. If "dummy" PC $j = 2$ is placed at the position (p', q'), it defines a U-layout with two q-rows of length $q' - 1$, where at each position of each row, two PCs can be placed; one at the bottom and one on top. This allows for the placement of in total $4(q'-1)$ PCs. Similarly, we can derive that in the p-row, $2(p'-2)$ PCs can be placed. The total number of PCs we want to place is n. If n is uneven, one (p, q)-position of the U will only have one PC; either at the bottom position or on top supported by an empty PC on the ground. This means that the minimal sufficient layout capacity is one PC larger than n. Further, since we allow unsymmetrical layouts with a gap of one PC right at the U's opening (see Figure 1.1 (b)), minimal sufficient layouts may provide an additional capacity for two PCs, which explains the left side of Constraint (1.13). Constraint (1.14) is derived in the same way.

Constraints (1.15) to (1.18) prohibit that PCs are placed at p- or q-positions larger than the position of "dummy" PC $j = 2$. Combined with Constraints (1.19) that enforce PCs to be placed in the same p- or q-row as either of the "dummy" PCs, this ensures that each PC is placed according to the U-layout as defined by the "dummy" PCs.

Further, Constraints (1.20) and (1.21) ensure that every (p, q)-position is occupied by no more than one PC at the bottom as well as one PC at the top position; if a "dummy" PC is present, this prohibits the use of the position completely. Constraints (1.22) assures that every PC $i \in I$ is placed exactly once into the storage area. Constraints (1.23) and (1.24) take care that the base and the nearest PC in both, p- and q-direction, are at least a distance of d^s apart. Hence, both constraints also limit the allowed narrowness of the U in p- and q-direction. Finally, Constraints (1.25) to (1.28) define the domains of the decision variables.

Due to Equation (1.1), USLPP is not linear.

1.3.4. Calculating ergonomic strains

In the problem description, in the MIP model as well as in the solution procedure (which we present in Section 1.4), ergonomic strains are treated in a generalized manner that allows for different ergonomic assessment methods to be used. However, for the remainder of this paper, we use a particular assessment method to attain consistent results in our computational study (see Section 1.5).

In general, there are four different approaches to assess workers' physical strains – and hence, to assess increases in risk for the development of MSDs – during MMH activities: expert evaluations, the study of epidemiological data (e.g., MSD incident rates of workers performing specific tasks or jobs), the determination of critical threshold values, and combinations of the aforementioned methods (Moore and Garg, 1995). Although expert evaluations are a valuable approach in practice, they are not suitable for the purpose of this paper due to their lack of objectiveness and reproducibility (Moore and Garg, 1995). In contrast, the study of epidemiological data – while being an objective assessment method –, is more concerned about finding and verifying general links between certain variables, tasks or jobs and the risk of developing MSDs (Dempsey, 1998). Furthermore, the collection and evaluation of epidemiological data is very time-consuming and, therefore, beyond the scope of this paper.

The determination of critical threshold values is based on the observation – e.g., the study of epidemiological data – that workers exceeding certain measurable or calculable threshold values while performing MMH activities are exposed to a higher risk of developing MSDs. Usually, three different approaches, each emphasizing different critical thresholds, are distinguished: the physiological, the biomechanical, and the psychophysical approach (Dempsey, 1998). Combinations of the aforementioned approaches have also been proposed.

The biomechanical approach focuses on the forces and torques certain joints, muscles or other body structures are exposed to during MMH activities; high forces on the lower back joints L_4/L_5 or L_5/S_1 have, for example, been linked to the development of lower back pain (Dempsey, 1998). The emphasis of the physiological approach is on a worker's cardiovascular effort while performing certain MMH tasks. The associated threshold value is most commonly expressed in oxygen consumption or energy consumption rates, which are directly correlated (Dempsey, 1998). Finally, psychophysical approaches consider a worker's perceived strain during MMH tasks to quantify how hazardous the task is to the worker (Dempsey, 1998).

While direct measurements of certain threshold values are possible – oxygen consumption rates, for example, can be measured via special masks (Levine, 2005) and intervertebrate pressures can be measured via surgically implanted sensors (Wilke et al., 1999) – most commonly models that have been developed to calculate respective values are used in practice (Dempsey, 1998). One such model, which has frequently been applied in recent years to quantify the ergonomic strain of MMH tasks in order picking (cf. Section 1.2), is the energy expenditure prediction model developed by Garg et al. (1978). This physiological model is based on the assumption that the energy expenditure for performing a complex task is the sum of the energy expenditures of its comprising basic tasks. An advantage of the model of Garg et al. (1978) is that it provides equations for calculating energy expenditure rates for a broad spectrum of basic tasks, depending on the task's properties and the performing worker's anthropometric characteristics. For example, the energy expenditure for lifting an item depends on the item's weight, the body posture adopted by the worker at the beginning and at the end of the movement, as well as the worker's sex and body weight. All equations were derived by the authors by applying regression analyses to data collected in laboratory experiments.

The evaluation of the relevant tasks, for which we need to quantify ergonomic strains in USLPP – i.e., walking without load, carrying items, extracting items from PCs, and dropping off items at the base –, necessitates a flexible ergonomic assessment method. Therefore, and because of its widespread use in practice, we employ the energy expenditure prediction model of Garg et al. (1978) to assess ergonomic strains in this paper.

We calculate the ergonomic strain of each considered task (i.e., $w_i^{w,c}\, d_{p,q}$, w_i^b, w_i^t and w_i^d, $\forall i \in I$) as a value in the physical dimension of absolute energy expenditure. This enables us to sum up these values to calculate the total energy needed for picking all items, as we did in the objective function of USLPP. The total energy needed can then be divided by the total time required to derive the (mean) energy expenditure rate the worker has to apply during picking. This approach is equal to considering the whole picking process as a single, complex task. For additional clarification, the Appendix demonstrates how energy expenditures are derived in this paper for an exemplary picking task.

We now take account of the fact that the picker has to walk the distance from the base to a PC containing a requested item twice, once empty and once carrying the item back to the base. To simplify calculations, we summarize the energy expenditures for both activities in a single parameter, $w_i^{w,c}$. Further, we calculate this parameter, $w_i^{w,c}$, in the physical dimension of energy

expenditure per distance walked[3], so that we can multiply it with the walking distance (which depends on the position of the base and the PC that contains the requested item) to attain the total energy expenditure for this basic task. We further assume that items are always carried at hip height.

For calculating energy expenditures for extracting items (w_i^b for extracting from a bottom PC; w_i^t for extracting from a top PC) and for dropping off items (w_i^d), we only consider movements in the sagittal body plane, since those account for the majority of the tasks' energy expenditures. Beyond that, we assume items are (on average) extracted from the center of a PC and dropped off at the center of the base.

Finally, we use the anthropometric measurements of the same, representative worker throughout all calculations to ease comparisons in our computational study (cf. Section 1.5), which is an approach that has frequently been used in earlier works in this area (cf., Battini et al., 2016, Diefenbach et al., 2020**, Glock et al., 2019). Diefenbach et al. (2020)** showed that implications derived from assuming a representative worker's anthropometrics are valid for a broad spectrum of workers with differing anthropometric measurements. Based on observations in practice, we assume the average worker to be male, weighing 75 kg, measuring 178 cm in height and having a walking speed of 1.4 $\frac{\text{m}}{\text{s}}$ (cf., Glock et al., 2019, Garg et al., 1978). However, these measurements can be freely adjusted to account for different individuals if desired.

Given a set of items I with their respective weights w_i, $\forall i \in I$, we are now able to derive all ergonomic strains needed for USLPP ($w_i^{w,c}$, w_i^b, w_i^t and w_i^d). We calculate all ergonomic strains in advance and forward them as fixed parameters to attain a respective instance of er-USLPP, which we can then solve utilizing the solution procedure proposed in Section 1.4.

1.4. Solution procedure

We solve both versions of USLPP optimally[4] by disaggregating the problem into two consecutive problems and solving them in hierarchical order. First, we determine a feasible U-layout and a base position. Secondly, as a sub-problem, we assign PCs to (p, q)-positions within the

[3] Garg et al. (1978) formulate the equation for calculating energy expenditures for walking and carrying depending on the duration of the respective activity. To derive energy expenditures per distance walked, we divide these equations by the worker's walking speed.

[**] In the original version, this referred to an unpublished result by Diefenbach et. al (2019). The reference has been updated in this version, as it is now publicly available.

[4] within numerical precision of δ_r for $r^\dagger$, the position of the base.

determined layout, minimizing the respective objective value for that layout. To obtain the optimal solution for USLPP, we repeat both steps for all possible layouts and save the solution yielding the minimal objective value. In the following, we explain the solution procedure in detail for er-USLPP. However, since ec-USLPP is a special case of er-USLPP (cf. Section 1.3.3), the procedure is also valid for the former.

1.4.1. Assigning PCs to feasible (p, q)-positions

This section examines the sub-problem that consists of assigning PCs to feasible (p, q)-positions, such that the total ergonomic strain for picking the items stowed in the respective PCs is minimized. We assume that both the U-layout (as defined by $y_{p,q,1}$ and $y_{p,q,2}$) and the position of the base $r^{\dagger}$ have already been determined (in a feasible manner) and that, consequently, the set D of distance $d_{p,q}$ to all feasible (p, q)-positions – which only depend on those variables (cf. Equation (1.1) - (1.3)) – have already been derived.

Additionally, we introduce a relabeling of indices by replacing the pair (p, q) by a single index $k \in K$ for all (p, q)-positions where PCs are allowed to be placed. We replace $(p, q) = (1, 1)$ with $k = 1$ and proceed in a clockwise manner, while incrementing k. For the example depicted in Figure 1.2, this results in $(1, 1) \rightarrow 1$, $(1, 2) \rightarrow 2$, $(1, 3) \rightarrow 3$, $(2, 4) \rightarrow 4$, ..., $(1, 5) \rightarrow 9$ and $|K| = |\{1, 2, ..., 9\}| = 9$. In addition, we define new decision variables by replacing the previous ones according to the mapping $x^b_{k,i} \rightarrow \bar{x}_{k,i}$; $x^t_{k,i} \rightarrow \bar{x}_{k+|K|,i}$, $\forall k \in K; i \in I$. If $|I| < 2|K|$, i.e., if the number of PCs is smaller than the number of available top and bottom positions, we further add "ghost" PCs $\bar{x}_{k,i}, \forall k \in K'; i \in K' \backslash I$, with $K' = K \bigcup\limits_{k \in K} \{k+|K|\}$ that do not contain any items and therefore do not contribute to the walking distance or ergonomic strain; i.e., they do not influence the objective function. Hence, "ghost" PCs correspond to empty PCs that we can use to support regular PCs on their top. This allows us to formulate the following MIP, which we call U-shaped layout assignment problem (USLAP):

$$\text{[USLAP] Minimize} \sum_{k \in K'} \sum_{i \in I'} c_{k,i}\, \bar{x}_{k,i} \tag{1.29}$$

subject to

$$\sum_{k \in K'} \bar{x}_{k,i} = 1 \qquad \forall i \in I' \tag{1.30}$$

$$\sum_{i \in I'} \bar{x}_{k,i} = 1 \qquad \forall k \in K' \tag{1.31}$$

$$\bar{x}_{k,i} \in \{0,1\} \qquad \forall k \in K'; i \in I' \quad (1.32)$$

with

$$c_{k,i} = \begin{cases} f_i \left(d_k \, w_i^{w,c} + w_i^b + w_i^d\right) & \forall k \in K; i \in I \\ f_i \left(d_k \, w_i^{w,c} + w_i^t + w_i^d\right) & \forall k \in K' \setminus K; i \in I \\ 0 & \forall k \in K'; i \in I' \setminus I \end{cases} . \quad (1.33)$$

In comparison to USLPP, in USLAP, most of the constraints become obsolete, since either the corresponding variables have already been determined (Constraints (1.1), to (1.5) - (1.14), (1.23), (1.24) and (1.26)), or the relevant variable space has been reduced by discarding (p,q)-positions not part of the U (Equations (1.15) - (1.19)). Constraints (1.30) corresponds to the former Constraints (1.22), while (1.31) corresponds to the Constraints (1.20) and (1.21). The resulting problem is in the form of a linear assignment problem. It can be solved in a polynomial runtime of $O(|K'|^3)$ using, for example, the formulation of Munkres (1957) of the Hungarian algorithm.

1.4.2. Determining feasible U-layouts and base positions

To simplify notation, we define Δ_p and Δ_q as discrete lengths (i.e., the number of p- and q-positions) of the U-zone in p- and q-direction, respectively. For example, for the case illustrated in Figure 1.2, it follows that $\Delta_p = 5$ and $\Delta_q = 4$. We define the following procedure. *Step (1):* Our procedure starts by setting the U's dimensions, beginning by setting $\Delta_p = 3 \Leftrightarrow y_{3,1,1} = 1$, which is the minimal feasible value for Δ_p. *Step (2):* We then – in accordance with Constraints (1.13) and (1.14) – derive the feasible value for Δ_q as

$$\Delta_q = \left\lceil \frac{n - 2(\Delta_p - 2))}{4} \right\rceil + 1 \quad \Leftrightarrow \quad y_{\Delta_p,\Delta_q,2} = 1, \quad (1.34)$$

which results in a minimal sufficient U-layout. *Step (3):* In the next step, we check if the U-zone is sufficiently wide in the p-direction to contain the base; i.e., if Constraint (1.23) is not violated. This may, for example, be the case if $\Delta_p = 3$, i.e., if the U's opening measures only one discrete p-position in width. If the current U-zone is feasible, we proceed by choosing a feasible position $r^\dagger$ for the base. Due to Theorem 1.1, we begin with setting $r = \frac{1}{2}\left((l^f + s)(\Delta_q - 1) - s\right)$, which is in the middle of the q-direction of PCs contained in the q-rows. Otherwise, if Constraint (1.23) is violated, we increment Δ_p by one and go to *Step (2)*. *Step (4):* After setting $r^\dagger$, we proceed with checking if the base position is feasible in q-direction; i.e., if Equation (1.22) is

not violated. If this is true, we calculate the set D of distances $d_{p,q}$ between the base and every (p,q)-position where PCs are allowed to be placed. Otherwise, we increment Δ_p by one and go to *Step (2)*. *Step (5):* We forward set D to solve the emerging instance of USLAP utilizing the Hungarian method. Afterwards, we increase $r^\dagger$ by a small amount δ_r and go to *Step (4)*. This procedure is continued until $\Delta_p \geq \frac{1}{2}n + 2 \Leftrightarrow \Delta_q < 2$ (cf. Equation (1.34)) is reached, after which no further feasible layouts can be found.

The number of feasible values for Δ_q (and, hence, the number of feasible U-layouts) grows according to $O\left(\frac{1}{4}n\right)$ (cf. Equation (1.34)). For a given value of Δ_q, there exists an order of $O\left(\frac{1}{2}\frac{\Delta_q\,(l^f+s)}{\delta_r}\right)$ possible base positions $r^\dagger$. Using the gaussian sum formula, this yields a number of possible combinations of U-layouts and base positions that grows according to $O\left(\frac{1}{16}\frac{l^f+s}{\delta_r}n^2\right)$. Hence, the asymptotic runtime of the entire solution procedure (that also solves USLAP with the Hungarian method) is polynomially bounded by $O\left((\frac{1}{16}\frac{l^f+s}{\delta_r}n^2)(n^3)\right) = O\left(\frac{1}{16}\frac{l^f+s}{\delta_r}n^5\right)$.

Theorem 1.1. *In the optimal solution S^* of USLPP, the base is always placed at $r^* \geq \frac{1}{2}\big((l^f+s)(\Delta_q-1)-s\big)$.*

Proof. We prove Theorem 1.1 by contradiction in a mostly visual way. Assume there exists an optimal solution S^*, where the base is placed at $r^* < \frac{1}{2}\left((l^f+s)(\Delta_q-1)-s\right)$, i.e., below the middle in q-direction of the PCs contained in the q-rows, like the one exemplarily depicted in Figure 1.2. We can create an alternative feasible solution S' by flipping the base as well as the PCs contained in the q-rows at an axis parallel to the p-direction at $r = \frac{1}{2}\left((l^f+s)(\Delta_q-1)-s\right)$ and keeping the PCs contained in the p-row in place. The distance between the PCs contained in the q-rows and the base has obviously not changed. However, the distance between the base and each PC contained in the p-row was reduced (since the base moved closer to the p-row in q-direction by a distance of $\frac{1}{2}\left((l^f+s)(\Delta_q-1)-s\right)-r^*$). This, in turn, means that there exists a mapping of all PCs of solution S^* to solution S', such that no PC's distance to the base increases, but at least one PC's distance decreases (since the p-row must at least contain one PC). Hence, the objective value of S' is smaller than the one of S^*, which contradicts that S^* is the optimal solution and that $r^* < \frac{1}{2}\left((l^f+s)(\Delta_q-1)-s\right)$ holds true in the optimal solution. □

1.5. Computational study

This section presents the results of a computational study to demonstrate the efficiency of our proposed solution procedure. Beyond that, we compare non-optimized U-shaped layouts to layouts optimized with respect to either the total walking distance or the total ergonomic strain for picking all stowed items with their respective pick frequencies.

1.5.1. Generating instances

To examine our solution procedure, we randomly generate test instances for USLPP with different zone sizes $n = \{30, 60, 100\}$, to which we refer as small, large and very large, respectively. While the small and large instance are comparable to real-world problem sizes (cf. Glock and Grosse (2012)), the purpose of the very large instances is to demonstrate the solution procedure's efficiency, since problems of comparable size are unlikely to occur in practice. For each instance, the items' pick frequencies and weights are randomly drawn from intervals according to a uniform distribution. We define one broad and one narrow interval for each problem parameter in question. For pick frequencies, we assume that the picker performs about 200 picks per hour (cf., Marras et al., 2010), from which we derive intervals of $\frac{20}{n}\,[8, 12]$ and $\frac{20}{n}\,[2, 18]$ picks per item type and per hour. For the weights, we use the intervals $\{[12, 18], [5, 25]\}$ kg, which are representative distributions in industry (e.g., Drury et al., 1982), and calculate the resulting ergonomic strains according to the approach described in Section 1.3.4. Note that the mean of the broad and narrow intervals are equal for both, pick frequencies and weights, respectively, which allows us to derive insights into the respective distribution's influence in our analysis in Section 1.5.2. For the large and small instances, we generate ten instances for each possible combination of broad and narrow intervals of pick frequencies and weights. For the very large instance, we only generate ten instances with frequencies and weights drawn from the broad intervals, since this is sufficient to examine the solution procedure's performance.

For all instances, the measurements of the PCs is set to $l^f = 120$ cm, $l^d = 80$ cm and $l^h = 97$ cm (cf., EPA, 2016). The measurements of the base are set to $b^p = b^q = b^h = 80$ cm. Further, we set $s = 10$ cm, $d^s = 50$ cm and do not restrict the size of the zone ($a^p = a^q = \infty$).

Instances are labeled in the following way. From left to right, each instance label starts with a capital "I" followed by four segments separated by hyphen. The number in the fist segment denotes the size (i.e., the number n of PCs) of the instance. The letter in the brackets of the second segment denotes if pick frequencies were drawn from the wide ("w") or narrow ("n")

interval. Likewise, the letter in the third segment's brackets denotes the respective interval from which item weights were drawn. The last segment is a two-digit continuous counting number to identify the instance. In total, we generate 90 instances, which are available from `https://doi.org/10.5281/zenodo.3364271`.

1.5.2. Computational results

We solve each instance with the solution procedure proposed in Section 1.4 twice; first, we minimize total walking distance and, secondly, we minimize total ergonomic strain. Beyond that, for every instance, we generate 1000 random solutions (i.e., all parameters are chosen randomly) and calculate their mean total walking distance and ergonomic strain. The primary purpose of the randomly generated solutions is to provide a reference value for every instance that allows us to examine the influences of the instances' sizes, pick frequencies and item weights on the optimization. A comparison of the optimal solutions to heuristic solutions that emulate U-zone configurations often employed in practice is provided later in this section.

The implementation was done in C#. Computational testing was performed on an Intel Core i7-3631QM CPU @ 2.20 GHz and with 8 GB of RAM. The precision for the optimal base position is set to $\delta_r = 1$ cm for all instances, which we regard as sufficiently accurate for practical applications. The results for all instances are summarized in Table 1.2. The following evaluations, however, focus primarily on the small and large instances, since their size is more representative for practical cases. In contrast, the purpose of the very large instances is to demonstrate the solution procedure's efficiency even for huge problem sizes. Detailed reports of all results are available from `https://doi.org/10.5281/zenodo.3364271`.

Our proposed solution procedure was able to solve every small instance in below two seconds to optimality, while every large instance was solved in under 47 s. Even the very large instances could be solved in below nine minutes. These results demonstrate that our solution procedure is sufficient to solve problems of even exceptionally large sizes in acceptable time. As a side note, we observe the runtimes are on average about 14% lower for optimizing walking distances compared to optimizing ergonomic strains. This can most likely be attributed to the former's objective function being easier to compute, given that most parameters have been set to one in this case.

The relative average reduction of total ergonomic strains achieved by the optimization ranges from 17.56% to 34.33%, where the reduction seems to depend mainly on two factors; the in-

instance			objective: minimizing total walking distance						objective: minimizing total ergonomic strain					
labeling	aver. dist. [$\frac{m}{h}$]	aver. ergo. [$\frac{kcal}{h}$]	opt. dist. [$\frac{m}{h}$]	rel. impr. dist. [%]	res. ergo. [$\frac{kcal}{h}$]	rel. gap ergo. [%]	(Δ_p^*, Δ_q^*, r^*) [-, -, m]	time [s]	opt. ergo. [$\frac{kcal}{h}$]	rel. impr. ergo. [%]	res. dist. [m]	rel. gap dist. [%]	(Δ_p^*, Δ_q^*, r^*) [-, -, m]	time [s]
I-n30-f[n]-w[n]-01	889.81	319.07	592.08	33.46	262.78	0.22	(4, 8, 5.80)	1.17	262.19	17.83	593.92	0.31	(4, 8, 5.80)	1.36
I-n30-f[n]-w[n]-02	932.93	339.81	623.46	33.17	280.95	0.11	(4, 8, 5.80)	1.25	280.65	17.41	624.89	0.23	(4, 8, 5.80)	1.44
I-n30-f[n]-w[n]-03	913.07	331.82	612.17	32.95	274.21	0.13	(4, 8, 5.80)	1.21	273.85	17.47	613.06	0.15	(4, 8, 5.80)	1.50
I-n30-f[n]-w[n]-04	923.85	333.59	619.11	32.99	275.71	0.09	(4, 8, 5.80)	1.20	275.45	17.43	620.29	0.19	(4, 8, 5.80)	1.44
I-n30-f[n]-w[n]-05	911.69	326.96	610.73	33.01	270.18	0.17	(4, 8, 5.80)	1.24	269.71	17.51	611.99	0.21	(4, 8, 5.80)	1.58
I-n30-f[n]-w[n]-06	911.66	331.92	609.15	33.18	274.03	0.14	(4, 8, 5.80)	1.18	273.64	17.56	609.82	0.11	(4, 8, 5.80)	1.47
I-n30-f[n]-w[n]-07	926.02	337.14	618.86	33.17	278.70	0.15	(4, 8, 5.80)	1.11	278.30	17.45	619.74	0.14	(4, 8, 5.80)	1.44
I-n30-f[n]-w[n]-08	848.92	305.28	567.63	33.14	252.08	0.16	(4, 8, 5.80)	1.15	251.69	17.55	568.14	0.09	(4, 8, 5.80)	1.39
I-n30-f[n]-w[n]-09	910.14	326.45	603.26	33.72	268.20	0.15	(4, 8, 5.80)	1.25	267.80	17.97	603.82	0.09	(4, 8, 5.80)	1.46
I-n30-f[n]-w[n]-10	903.19	323.87	605.05	33.01	267.65	0.12	(4, 8, 5.80)	1.22	267.33	17.46	606.97	0.32	(4, 8, 5.80)	1.42
mean	**907.13**	**327.59**	**606.15**	**33.18**	**270.45**	**0.14**		**1.20**	**270.06**	**17.56**	**607.27**	**0.18**		**1.45**
I-n30-f[b]-w[n]-01	992.12	361.12	577.79	41.76	282.06	0.29	(4, 8, 5.80)	1.31	281.24	22.12	578.93	0.20	(4, 8, 5.80)	1.57
I-n30-f[b]-w[n]-02	840.51	301.39	485.13	42.28	235.07	0.58	(4, 8, 5.80)	1.24	233.72	22.45	485.84	0.15	(4, 8, 5.80)	1.45
I-n30-f[b]-w[n]-03	845.69	307.77	489.52	42.12	240.13	0.24	(4, 8, 5.80)	1.23	239.54	22.17	489.81	0.06	(4, 8, 5.80)	1.44
I-n30-f[b]-w[n]-04	948.51	340.13	557.39	41.24	266.82	0.41	(4, 8, 5.80)	1.18	265.73	21.87	558.10	0.13	(4, 8, 5.80)	1.52
I-n30-f[b]-w[n]-05	807.40	291.83	452.17	44.00	224.92	0.56	(4, 8, 5.80)	1.32	223.66	23.36	452.52	0.08	(4, 8, 5.80)	1.54
I-n30-f[b]-w[n]-06	810.13	295.27	462.00	42.97	228.97	0.32	(4, 8, 5.80)	1.23	228.24	22.70	462.30	0.06	(4, 8, 5.80)	1.52
I-n30-f[b]-w[n]-07	866.95	312.97	491.03	43.36	242.23	0.50	(4, 8, 5.80)	1.13	241.03	22.99	492.08	0.21	(4, 8, 5.80)	1.42
I-n30-f[b]-w[n]-08	858.38	309.21	487.83	43.17	239.58	0.47	(4, 8, 5.80)	1.25	238.45	22.88	488.26	0.09	(4, 8, 5.80)	1.43
I-n30-f[b]-w[n]-09	970.70	350.99	564.35	41.86	274.02	0.37	(4, 8, 5.80)	1.21	273.01	22.22	565.01	0.12	(4, 8, 5.80)	1.46
I-n30-f[b]-w[n]-10	826.62	297.91	463.66	43.91	229.62	0.52	(4, 8, 5.80)	1.28	228.43	23.32	464.13	0.10	(4, 8, 5.80)	1.48
mean	**876.70**	**316.86**	**503.09**	**42.67**	**246.34**	**0.43**		**1.24**	**241.31**	**22.61**	**503.70**	**0.12**		**1.48**
I-n30-f[n]-w[b]-01	893.85	327.97	595.91	33.33	272.23	0.59	(4, 8, 5.80)	1.23	270.62	17.49	603.67	1.29	(4, 8, 5.80)	1.44
I-n30-f[n]-w[b]-02	950.37	355.49	639.04	32.76	294.78	0.40	(4, 8, 5.80)	1.25	293.60	17.41	645.06	0.93	(4, 8, 5.80)	1.41
I-n30-f[n]-w[b]-03	926.49	339.99	620.34	33.04	280.97	0.51	(4, 8, 5.80)	1.25	279.53	17.78	624.53	0.67	(4, 8, 5.80)	1.52
I-n30-f[n]-w[b]-04	907.50	339.45	602.15	33.65	280.30	0.42	(4, 8, 5.80)	1.18	279.14	17.77	606.94	0.79	(4, 8, 5.80)	1.43
I-n30-f[n]-w[b]-05	928.21	331.25	619.55	33.25	273.00	0.52	(4, 8, 5.80)	1.23	271.60	18.01	624.55	0.80	(4, 8, 5.80)	1.40
I-n30-f[n]-w[b]-06	976.55	349.11	648.05	33.64	286.90	0.63	(4, 8, 5.80)	1.29	285.10	18.34	655.27	1.10	(4, 8, 5.80)	1.42
I-n30-f[n]-w[b]-07	929.56	336.85	613.13	34.04	276.20	0.59	(4, 8, 5.80)	1.22	274.58	18.49	618.82	0.92	(4, 8, 5.80)	1.44
I-n30-f[n]-w[b]-08	907.54	322.35	610.81	32.70	266.79	0.51	(4, 8, 5.80)	1.28	265.44	17.66	616.15	0.87	(4, 8, 5.80)	1.47
I-n30-f[n]-w[b]-09	925.53	333.43	627.09	32.25	277.00	0.45	(4, 8, 5.80)	1.21	275.76	17.29	631.46	0.69	(4, 8, 5.80)	1.37
I-n30-f[n]-w[b]-10	902.59	323.43	596.71	33.89	264.22	0.30	(4, 8, 5.80)	1.21	263.43	18.55	599.98	0.54	(4, 8, 5.80)	1.41
mean	**924.82**	**335.93**	**617.28**	**33.25**	**277.24**	**0.49**		**1.23**	**275.88**	**17.88**	**622.64**	**0.86**		**1.43**
I-n30-f[b]-w[b]-01	894.04	318.93	514.04	42.50	248.02	0.61	(4, 8, 5.80)	1.28	246.50	22.71	520.99	1.33	(4, 8, 5.80)	1.37
I-n30-f[b]-w[b]-02	948.86	346.11	551.80	41.85	269.90	0.39	(4, 8, 5.80)	1.17	268.85	22.32	556.71	0.88	(4, 8, 5.80)	1.36
I-n30-f[b]-w[b]-03	800.17	300.09	450.61	43.69	232.27	0.32	(4, 8, 5.80)	1.23	231.53	22.85	452.55	0.43	(4, 8, 5.80)	1.45
I-n30-f[b]-w[b]-04	804.91	293.76	476.14	40.84	232.23	0.52	(4, 8, 5.80)	1.26	231.02	21.36	477.82	0.35	(4, 8, 5.80)	1.41
I-n30-f[b]-w[b]-05	909.16	325.47	522.69	42.51	251.44	0.37	(4, 8, 5.80)	1.27	250.51	23.03	525.35	0.51	(4, 8, 5.80)	1.42
I-n30-f[b]-w[b]-06	1103.63	415.32	692.70	37.23	334.54	0.33	(4, 8, 5.80)	1.21	333.45	19.71	694.53	0.26	(4, 8, 5.80)	1.36
I-n30-f[b]-w[b]-07	1017.32	362.68	584.52	42.54	281.39	0.43	(4, 8, 7.04)	1.28	280.19	22.74	590.67	1.04	(4, 8, 5.80)	1.39
I-n30-f[b]-w[b]-08	820.91	285.29	458.15	44.19	218.90	0.64	(4, 8, 7.14)	1.22	217.51	23.76	463.58	1.17	(4, 8, 7.10)	1.37
I-n30-f[b]-w[b]-09	930.60	338.91	543.96	41.55	265.04	0.41	(4, 8, 5.80)	1.36	263.95	22.12	549.14	0.94	(4, 8, 5.80)	1.48
I-n30-f[b]-w[b]-10	929.07	335.63	550.47	40.75	264.31	0.48	(4, 8, 5.80)	1.28	263.05	21.62	552.65	0.39	(4, 8, 5.80)	1.42
mean	**915.87**	**332.22**	**534.51**	**41.77**	**259.80**	**0.45**		**1.26**	**258.66**	**22.22**	**538.40**	**0.73**		**1.40**
I-n60-f[n]-w[n]-01	1807.59	501.30	1074.60	40.55	362.85	0.10	(4, 15, 9.70)	36.88	362.48	27.69	1076.63	0.19	(4, 15, 9.70)	40.48
I-n60-f[n]-w[n]-02	1774.13	491.92	1049.71	40.83	355.55	0.10	(4, 15, 9.70)	36.94	355.20	27.79	1051.51	0.17	(4, 15, 9.70)	42.96
I-n60-f[n]-w[n]-03	1806.39	499.86	1071.55	40.68	361.54	0.12	(4, 15, 9.70)	37.52	361.10	27.76	1073.80	0.21	(4, 15, 9.70)	43.00
I-n60-f[n]-w[n]-04	1816.73	501.70	1070.77	41.06	361.09	0.08	(4, 15, 9.70)	37.53	360.81	28.08	1072.27	0.14	(4, 15, 9.70)	41.21
I-n60-f[n]-w[n]-05	1809.76	500.85	1069.79	40.89	361.54	0.12	(4, 15, 9.70)	39.35	361.09	27.90	1071.90	0.20	(4, 15, 9.70)	42.04
I-n60-f[n]-w[n]-06	1785.00	495.32	1054.10	40.95	357.67	0.09	(4, 15, 9.70)	37.90	357.36	27.85	1055.23	0.11	(4, 15, 9.70)	41.10
I-n60-f[n]-w[n]-07	1746.82	485.97	1032.38	40.90	350.95	0.08	(4, 15, 9.70)	36.78	350.68	27.84	1034.20	0.18	(4, 15, 9.70)	41.53
I-n60-f[n]-w[n]-08	1808.82	499.98	1063.76	41.19	359.30	0.08	(4, 15, 9.70)	40.68	359.02	28.19	1065.09	0.12	(4, 15, 9.70)	41.92
I-n60-f[n]-w[n]-09	1805.85	494.62	1062.80	41.15	355.56	0.11	(4, 15, 9.70)	38.70	355.16	28.20	1065.05	0.21	(4, 15, 9.70)	43.64
I-n60-f[n]-w[n]-10	1827.60	507.38	1070.91	41.40	364.52	0.11	(4, 15, 9.70)	37.19	364.11	28.24	1072.42	0.14	(4, 15, 9.70)	41.54
mean	**1798.87**	**497.89**	**1062.04**	**40.96**	**359.06**	**0.10**		**37.95**	**358.70**	**27.95**	**1063.81**	**0.17**		**41.94**
I-n60-f[b]-w[n]-01	2066.28	576.49	1066.50	48.39	387.24	0.14	(4, 15, 9.70)	39.33	386.70	32.92	1067.37	0.08	(4, 15, 9.70)	45.66
I-n60-f[b]-w[n]-02	1802.86	497.47	881.29	51.12	324.15	0.18	(4, 15, 9.70)	39.33	323.58	34.95	882.36	0.12	(4, 15, 9.70)	45.53
I-n60-f[b]-w[n]-03	1584.90	436.00	767.13	51.60	282.48	0.21	(4, 15, 9.70)	38.91	281.88	35.35	768.07	0.12	(4, 15, 9.70)	44.79
I-n60-f[b]-w[n]-04	1866.70	522.13	960.95	48.52	350.73	0.16	(4, 15, 9.70)	38.29	350.17	32.93	961.83	0.09	(4, 15, 9.70)	44.54
I-n60-f[b]-w[n]-05	1859.15	512.88	946.86	49.07	341.51	0.17	(4, 15, 9.70)	38.67	340.93	33.53	947.56	0.07	(4, 15, 9.70)	43.98
I-n60-f[b]-w[n]-06	1968.55	541.35	986.94	49.86	357.43	0.17	(4, 15, 9.70)	37.82	356.81	34.09	988.14	0.12	(4, 15, 9.70)	43.45
I-n60-f[b]-w[n]-07	1764.90	492.67	868.09	50.81	322.30	0.18	(4, 15, 9.70)	37.84	321.74	34.70	869.17	0.12	(4, 15, 9.70)	43.55
I-n60-f[b]-w[n]-08	1863.70	513.19	950.83	48.98	341.87	0.14	(4, 15, 9.70)	39.39	341.39	33.48	952.20	0.14	(4, 15, 9.70)	44.33
I-n60-f[b]-w[n]-09	1699.05	465.28	846.27	50.19	305.61	0.15	(4, 15, 9.70)	38.53	305.15	34.42	847.00	0.09	(4, 15, 9.70)	43.86
I-n60-f[b]-w[n]-10	1809.49	505.15	869.80	51.93	326.55	0.12	(4, 15, 9.70)	40.12	326.15	35.43	870.11	0.04	(4, 15, 9.70)	46.03
mean	**1828.56**	**506.26**	**914.47**	**50.05**	**333.99**	**0.16**		**38.82**	**333.45**	**34.18**	**915.38**	**0.10**		**44.57**
I-n60-f[n]-w[b]-01	1808.96	499.92	1069.25	40.89	360.14	0.47	(4, 15, 9.70)	36.22	358.47	28.29	1076.09	0.64	(4, 15, 9.70)	39.73
I-n60-f[n]-w[b]-02	1783.90	473.63	1050.30	41.12	338.21	0.59	(4, 15, 9.70)	38.70	336.22	29.01	1060.04	0.92	(4, 15, 9.70)	42.94
I-n60-f[n]-w[b]-03	1834.48	504.47	1087.48	40.72	364.18	0.70	(4, 15, 9.70)	37.51	361.66	28.31	1097.08	0.88	(4, 15, 9.70)	42.07
I-n60-f[n]-w[b]-04	1783.20	501.42	1049.19	41.16	360.95	0.60	(4, 15, 9.70)	38.31	358.78	28.45	1059.36	0.96	(4, 15, 9.70)	42.87
I-n60-f[n]-w[b]-05	1779.07	505.43	1061.13	40.36	368.03	0.53	(4, 15, 9.70)	37.54	366.08	27.57	1070.92	0.91	(4, 15, 9.70)	41.96
I-n60-f[n]-w[b]-06	1823.76	494.02	1082.79	40.63	356.12	0.56	(4, 15, 9.70)	37.42	354.13	28.32	1091.28	0.78	(4, 15, 9.70)	42.31
I-n60-f[n]-w[b]-07	1793.08	494.01	1073.22	40.15	358.91	0.60	(4, 15, 9.70)	37.22	356.76	27.78	1083.02	0.90	(4, 15, 9.70)	41.60
I-n60-f[n]-w[b]-08	1765.68	474.55	1049.74	40.55	341.56	0.49	(4, 15, 9.70)	35.63	339.92	28.37	1056.82	0.67	(4, 15, 9.70)	43.22
I-n60-f[n]-w[b]-09	1821.09	493.52	1081.93	40.59	355.21	0.56	(4, 15, 9.70)	38.89	353.23	28.43	1091.97	0.92	(4, 15, 9.70)	41.89
I-n60-f[n]-w[b]-10	1804.46	498.50	1064.83	40.99	359.47	0.62	(4, 15, 9.70)	38.61	357.24	28.34	1075.44	0.99	(4, 15, 9.70)	41.90
mean	**1799.77**	**493.95**	**1066.99**	**40.72**	**356.28**	**0.57**		**37.60**	**354.25**	**28.29**	**1076.20**	**0.86**		**42.05**

explanations aver. = average value (of 1000 randomly generated solutions) ergo. = total ergonomic strain dist. = total walking distance
rel. impr. = relative improvement compared to the average value rel. gap = relative gap compared to the optimal value

Table 1.2.: Computational results.

instance			objective: minimizing total walking distance						objective: minimizing total ergonomic strain					
labeling	aver. dist. [$\frac{m}{h}$]	aver. ergo. [$\frac{kcal}{h}$]	opt. dist. [$\frac{m}{h}$]	rel. impr. dist. [%]	res. ergo. [$\frac{kcal}{h}$]	rel. gap ergo. [%]	(Δ_p^*, Δ_q^*, r^*) [-, -, m]	time [s]	opt. ergo. [$\frac{kcal}{h}$]	rel. impr. ergo. [%]	res. dist. [m]	rel. gap dist. [%]	(Δ_p^*, Δ_q^*, r^*) [-, -, m]	time [s]
I-n60-f[b]-w[b]-01	1705.93	464.89	840.77	50.71	304.81	0.61	(4, 15, 9.70)	39.15	302.96	34.83	848.03	0.86	(4, 15, 9.70)	40.99
I-n60-f[b]-w[b]-02	1728.07	485.25	832.30	51.84	314.60	0.37	(4, 15, 9.70)	37.94	313.45	35.40	834.83	0.30	(4, 15, 9.70)	41.73
I-n60-f[b]-w[b]-03	1935.06	540.21	983.70	49.16	360.56	0.37	(4, 15, 11.00)	37.82	359.25	33.50	989.79	0.62	(4, 15, 11.00)	43.47
I-n60-f[b]-w[b]-04	1865.73	518.88	941.52	49.54	345.81	0.44	(4, 15, 9.70)	38.84	344.30	33.65	948.69	0.76	(4, 15, 9.70)	42.92
I-n60-f[b]-w[b]-05	1766.78	488.15	884.82	49.92	323.69	0.48	(4, 15, 9.70)	39.17	322.15	34.01	888.29	0.39	(4, 15, 9.70)	42.30
I-n60-f[b]-w[b]-06	1753.27	484.60	864.42	50.70	317.98	0.46	(4, 15, 9.70)	40.59	316.51	34.69	868.65	0.49	(4, 15, 9.70)	44.01
I-n60-f[b]-w[b]-07	1779.40	490.40	875.52	50.80	321.18	0.34	(4, 15, 9.70)	38.68	320.09	34.73	879.16	0.41	(4, 15, 9.70)	42.99
I-n60-f[b]-w[b]-08	1647.34	463.93	822.59	50.07	306.80	0.33	(4, 15, 9.70)	39.10	305.80	34.08	827.50	0.59	(4, 15, 9.70)	42.94
I-n60-f[b]-w[b]-09	2061.54	571.82	1048.45	49.14	379.60	0.43	(4, 15, 9.70)	39.39	377.96	33.90	1055.28	0.65	(4, 15, 9.70)	44.19
I-n60-f[b]-w[b]-10	1638.87	461.69	806.90	50.76	303.12	0.29	(4, 15, 9.70)	36.00	302.25	34.53	811.93	0.62	(4, 15, 9.70)	40.75
mean	**1788.20**	**496.98**	**890.10**	**50.26**	**327.81**	**0.41**		**38.67**	**326.47**	**34.33**	**895.21**	**0.57**		**42.63**
I-n100-f[b]-w[b]-01	2751.83	665.02	1214.68	55.86	373.48	0.28	(4, 25, 16.20)	472.20	372.44	44.00	1221.72	0.58	(4, 25, 16.20)	516.58
I-n100-f[b]-w[b]-02	3061.87	749.82	1373.58	55.14	428.63	0.36	(4, 25, 16.20)	459.99	427.08	43.04	1380.81	0.52	(4, 25, 16.20)	515.68
I-n100-f[b]-w[b]-03	2925.23	716.16	1354.53	53.70	417.67	0.36	(4, 25, 16.20)	479.46	416.15	41.89	1361.63	0.52	(4, 25, 16.20)	520.50
I-n100-f[b]-w[b]-04	2958.29	719.68	1375.69	53.50	419.61	0.38	(4, 25, 16.20)	464.43	418.03	41.92	1383.69	0.58	(4, 25, 16.20)	499.66
I-n100-f[b]-w[b]-05	2870.11	680.06	1319.08	54.04	392.40	0.45	(4, 25, 16.20)	470.53	390.66	42.55	1326.46	0.56	(4, 25, 16.20)	517.04
I-n100-f[b]-w[b]-06	2755.59	663.78	1253.51	54.51	382.98	0.39	(4, 25, 16.20)	450.73	381.50	42.53	1260.83	0.58	(4, 25, 16.20)	504.46
I-n100-f[b]-w[b]-07	2964.26	695.12	1364.22	53.98	402.12	0.35	(4, 25, 16.20)	476.24	400.71	42.35	1372.28	0.59	(4, 25, 16.20)	511.89
I-n100-f[b]-w[b]-08	2867.43	679.71	1299.45	54.68	389.06	0.46	(4, 25, 16.20)	483.77	387.29	43.02	1305.68	0.48	(4, 25, 16.20)	513.14
I-n100-f[b]-w[b]-09	3130.26	773.13	1445.61	53.82	450.34	0.35	(4, 25, 16.20)	481.48	448.78	41.95	1454.31	0.60	(4, 25, 16.20)	519.75
I-n100-f[b]-w[b]-10	3092.19	744.23	1448.30	53.16	435.22	0.38	(4, 25, 16.20)	480.28	433.59	41.74	1455.69	0.51	(4, 25, 16.20)	500.92
mean	**2937.71**	**708.67**	**1344.87**	**54.24**	**409.15**	**0.38**		**471.91**	**407.62**	**42.50**	**1352.31**	**0.55**		**511.96**
explanations	aver. = average value (of 1000 randomly generated solutions) ergo. = total ergonomic strain dist. = total walking distance rel. impr. = relative improvement compared to the average value rel. gap = relative gap compared to the optimal value													

Table 1.2 (continued): Computational results.

stance size and the range of the pick frequencies. For all combinations of pick frequency and item weight ranges, optimizing the large instances yields an about 10 - 12 percentage points greater improvement as compared to the small ones. We attribute this to the large instance's greater flexibility, i.e., more possible layouts and assignments, which allows for greater improvement potential in the optimization. Consider, for example, the difference in the objective value of assigning a very heavy item type to the worst possible position in the worst possible layout compared to assigning it to the best position in the best possible layout. For larger instances, i.e., larger n, the difference in the objective value is (on average) greater. Hence, the difference between the optimal and an average randomly generated solution, which by chance contains some bad assignments, is greater for larger instances.

The range of the pick frequencies influences the achieved improvement in a similar way. Broader ranges result in an additional improvement of between 4 and 6 percentage points compared to smaller ranges. This is due to the fact that a broader range of pick frequencies results directly in a broader range of "costs" $c_{i,k}$ for assigning items to different positions (cf. Equation (1.33)), which – in a similar way as described above – increases the improvement potential between an average and the optimal solution.

In contrast to this, the influence of the item weight range is only marginal – even though one might have expected similar influences as observed for the pick frequency range. However, the range of "costs" $c_{i,k}$ is less influenced by item weights. This is due to two reasons. Firstly, the energy expenditure for walking and carrying items, $w_i^{w,c}$, is only weakly dependent on the

carried items' weights (cf., Garg et al., 1978). Instead, most of the worker's energy is spent simply on moving his/her body weight. We acknowledge that this weak dependence on the carried item's weight does not hold true for muscle fatigue of the arms, shoulders and back. However, the ergonomic assessment method of Garg et al. (1978) does not account for those, which is one of its shortcomings that consequently also extends to our evaluation. The second reason for the weak influence of item weight ranges on ergonomic strain is that the energy expenditures for picking an item either from a bottom (w_i^b) or top (w_i^t) position – while being stronger influenced by item weight – both increase by about the same amount with increasing weight. This explains why the difference between $c_{i,k}$, $\forall k \in K$ and $c_{i,k}$, $\forall k \in K' \setminus K$ (i.e., the importance of the decision to store items in a top or bottom position) also only weakly depends on item weights.

The optimization of walking distances yields average improvements between 33% and 50% compared to an average solution. As for the optimization of ergonomic strains, the amount of improvement strongly depends on instance size and the range of pick frequencies. The large instances' relative improvements are about 7 to 8 percentage points greater than the small instances' ones. Beyond that, improvements are about 8 to 10 percentage points greater for instances with large ranges of pick frequencies. The reasons for both observations are the same as for the optimization of ergonomic strains. Item weight ranges, since they do not influence walking distances, have no impact on the magnitude of achieved improvements.

Another observation regarding the objective values is that minimizing either ergonomic strains or walking distances also results in close-to-optimal values for the respective other objective with relative optimization gaps below 1% on average. This result may seem counterintuitive at first glance, but can be attributed to the small influence of item weights on the ergonomic strains (calculated according to the model of Garg et al. (1978)). If the ergonomic strain to carry each item is approximately equal (independent of the respective item's weight), the total ergonomic strain is minimized by minimizing the total amount of carrying, hence the total walking distance. We note that using alternative ergonomic assessment methods could, therefore, yield more conflicting objectives.

Comparing optimal solutions to randomly generated ones, as done above, may yield a biased estimate of improvements that can be realized in practice. Usually, when U-shaped zones are employed in practice, layouts and assignments are not determined randomly, but according to simple rules or heuristics. Glock and Grosse (2012) describe i.a. two such assignment heuristics, to which they refer as horizontal and vertical assignment. In both heuristics, the items are ranked

according to their pick frequency in decreasing order. Further, the storage locations are ranked according to, either, their increasing horizontal distance (i.e., in p-direction), or their increasing vertical distance (i.e., in q-direction) from the base. Each item is then assigned to the storage location with an equal rank. Layouts are usually determined by a rule of thumb. According to our observations in practice, the length of the p-row had often been chosen to be about half (for smaller U-shaped zones) to a third (for bigger U-shaped zones) of the length of one q-row. The base was usually placed in the middle of the q-rows.

To asses the improvement potential of optimal solutions over solutions resembling currently employed U-zones in practice, we solved all small and large instances according to the heuristic layout and storage assignment policies described above. We assumed $\Delta_p = 3$ and $\Delta_q = 6$ for the small as well as $\Delta_p = 4$ and $\Delta_q = 13$ for the large instances, which corresponds to a relation of one half as well as roughly one third between the lengths of the p- and q-row, respectively. Table 1.3 presents a comparison of the heuristic and optimal solutions. Detailed reports of the results are available from `https://doi.org/10.5281/zenodo.3364271`.

instance	objective: minimizing total walking distance					objective: minimizing total ergonomic strain				
labeling	opt. [m]	hor. assign. [m]	rel. impr. from hor. assign. [%]	vert. assign. [m]	rel. impr. from vert. assign. [%]	opt. [kcal]	hor. assign. [kcal]	rel. impr. from hor. assign. [%]	vert. assign. [kcal]	rel. impr. from vert. assign. [%]
I-n30-f[n]-w[n]-##	606.15	713.86	15.09	695.16	12.80	270.06	291.16	7.25	287.16	5.95
I-n30-f[b]-w[n]-##	503.09	689.11	27.00	611.02	17.66	241.31	281.45	14.26	265.48	9.10
I-n30-f[n]-w[b]-##	617.28	724.39	14.79	710.86	13.16	275.88	297.94	7.40	293.65	6.05
I-n30-f[b]-w[b]-##	534.51	723.29	26.10	646.66	17.34	258.66	296.32	12.71	279.39	7.42
I-n60-f[n]-w[n]-##	1062.04	1286.06	17.42	1236.10	14.08	358.70	401.34	10.62	391.21	8.31
I-n60-f[b]-w[n]-##	914.47	1300.99	29.71	1112.02	17.76	333.45	406.67	18.00	370.47	9.99
I-n60-f[n]-w[b]-##	1066.99	1299.43	17.89	1249.05	14.58	354.25	399.97	11.43	387.02	8.47
I-n60-f[b]-w[b]-##	890.10	1282.74	30.61	1089.20	18.28	326.47	401.07	18.60	362.99	10.06

Table 1.3.: Comparison of heuristic and optimal solutions.

Table 1.3 shows the mean objective values for every type of instance. Clearly, the vertical assignment outperforms the horizontal assignment. However, even the former yields results that are significantly worse than the optimal solutions. Optimal total walking distances are on average between 12.80% and 18.28% lower than the ones found via the vertical assignment heuristic. Energy expenditures are between 5.95% and 10.06% lower in the mean. This comparison demonstrates that the optimization of U-shaped zones can also yield significant improvements in practical situations, where U-zones are configured using heuristics.

Regarding the optimal layout, there are two main observations. Firstly, for every instance, the U-zone that is as narrow as possible in p-direction (and that is, consequently, most elongated in q-direction), is optimal. Secondly, the base is most of the time located in front of the center of the fifth ($r^* = 5.80$ m) or occasionally sixth ($r^* = 7.10$ m) PC in q-direction for the small

instances and in front of the center of the eighth ($r^* = 9.70$ m) or seldom ninth ($r^* = 11.00$ m) discrete position for the large instances. There are only a few exceptions, which have been underlined in Table 1.2.

The reasons why the optimal U-zone tends to be narrow in p-direction are twofold. On the one hand, narrow U-zones allow for better discrimination of item placement. To demonstrate this, consider two possible U-zones with the same number of PCs, one narrow and one more quadratic in shape, both with the base in a centered position. In the more quadratic layout, the distance between the base and the closest PCs (at the middle of the sides) and the distance between the base and the PCs farthest away from the base (at the corners) differ only by a comparatively small amount. On the contrary, in the narrow layout, these distances differ by a much greater amount. Here, the distance to the closest PCs are smaller and the distances to the more peripheral PCs are greater than in the quadratic case. Therefore, placing frequently picked items at positions closer to the base and less frequently picked ones farther away can be realized better for narrow U-zones. On the other hand, each U-layout consists of two q-rows, but only one p-row, which leads to advantages for U-zones elongated in the former direction and narrow in the latter. To better illustrate this aspect, again consider a quadratic layout. Narrowing this quadratic shape in p-direction by two discrete positions moves every PC contained in the q-rows one position closer to the base and every PC contained in the p-row one position farther away. Since there are twice the number of PCs in the q-rows than in the p-row, the average distance to a PC decreases. We note, however, that the optimal solution is not guaranteed to be of minimal width in the p-dimension. For example, an instance with $n = 30$, $f_1 = f_2 = 30$ and $f_i = 1$ picks per hour, $\forall i \in \{3, ..., n\}$ and $w_i = 15$ kg, $\forall i \in \{1, ..., n\}$ as well as every other parameter set to the values proposed in Section 1.5.1 has the optimal layout of $\Delta_p^* = 5$, $\Delta_q^* = 7$ and $r^* = 7.29$ m.

For both instance sizes, the base tends to be placed in front of the middle of the first (or rarely the second) PC in the q-rows behind (i.e., in positive q-direction) the axis at $r = \frac{1}{2}\big((l^f + s)(\Delta_q^* - 1) - s\big)$, which lies in the middle of the PCs contained in the respective rows (cf. Section 1.4.2). If the U-zone was missing the p-row, we would expect the base to be placed at the exact position of this axis, as this minimizes the total unweighted distances to all PCs. However, due to the PCs contained in the p-row, the optimal base position tends to be slightly offset in positive p-direction.

The fact that the optimal base position is, most of the time, positioned in front of the middle of a PC can be explained by considering the determination of the optimal base position as a Weber or facility location problem. In the basic formulation of the facility location problem, a set of customers with attributed weights and respective positions in a plane are given. The objective is

to decide on the location of a single facility, such that the sum of its weighted distances to the customers is minimized (for a comprehensive review of facility location problem, see Owen and Daskin (1998)). Assume that we already know the optimal U-shape and the optimal assignment of PCs to discrete positions for a given instance of USLPP. The problem of finding the optimal base position is then equivalent to a facility location problem, with PCs corresponding to customers, the terms $f_i\, w_i^{w,c}$ corresponding to their respective weights and the base corresponding to the facility. For rectangular distance measurements, it has been shown that the optimal position of the facility is always on par with one of the customers in both directions; i.e., the p- and q-direction in the case of USLPP (Francis, 1963). Further, the facility's positions in both directions are mutually independent. For euclidean distance measurements, however, these properties do not hold.

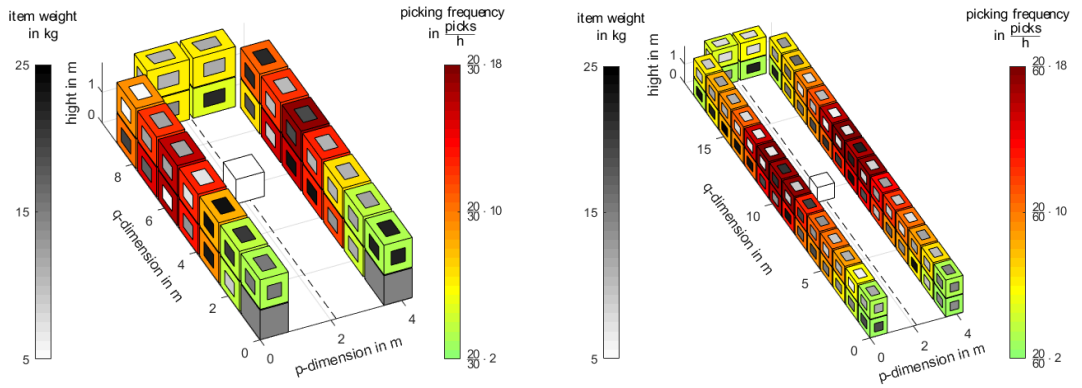
In USLPP, the base's position in p-direction is already fixed by the symmetry line of the U-zone. Its position in q-direction, $r^{\dagger}$, still needs to be determined. Since in USLPP, we measure distances according to a weighted mean of rectangular and euclidean distances (and allow for different weights α, cf. Equation (1.1)), the optimal base position does not necessarily obey the property of facility location problems with rectangular distance measurements (as, for example, the optimal solution for instance I-n30-f[b]-w[b]-07 shows). However, as the results in Table 1.2 indicate, for most cases, the property still holds.

All test instances were generated such that about 200 picks per hour need to be fulfilled, independent of the instances' sizes. Therefore, by comparing the optimization results of the small and large instances, we can draw conclusions on the influence of the U's size on pick efficiency. The optimal total travel distances of the large instances are on average 74% higher than those of the small ones. This increase can be attributed to the former's size, which is about twice the latter's size and, therefore, increases average walking distances. Increased average walking distances for larger instances are, however, partially compensated by the improved possibilities to assign less frequently picked item types to storage locations farther away from the depot (hence, the travel distances are not doubled). Due to the same reason, larger instances yield, on average, 31% higher optimal total ergonomic strains. Based on these findings, it is recommended to store items that are never picked in the same batch in multiple smaller U's instead of a single bigger one.

This result has further significance for the ergonomic evaluation of our test instances. While recommendations often differ, most experts agree that exceeding an energy expenditure rate of $5\frac{\text{kcal}}{\text{min}} = 300\frac{\text{kcal}}{\text{h}}$ over the course of an eight hour work day leads to significant health risks (Mi-

tal, 2017). The energy expenditure of an average, randomly generated solution exceeds this threshold for all instance sizes, which – since we generated instances with real-world oriented properties – further points to the necessity of an ergonomic optimization. By optimizing the small instances, we are able to reduce the total ergonomic strain to values below the critical threshold. On the contrary, for the large instances, the optimized total ergonomic strains still exceed the critical value and could only be further lowered by reducing total pick frequencies or introducing intermediate rest periods, and therewith decreasing system efficiency. We note that our analysis is based on quite restrictive assumptions on pick frequencies, item weights and the anthropometric characteristics of the worker. Further, we did not consider the generation of work plans that may schedule breaks or activities with low ergonomic strains in-between physically exhausting tasks, which could reduce average energy expenditures. The results, therefore, deviate from a real-world situation. Nevertheless, the general implications are still valid.

Finally, we analyze the properties of the optimal assignment. Figure 1.3 depicts the optimal solutions that minimize the total ergonomic strains for instance I-n30-f[b]-w[b]-01 and I-n60-f[b]-w[b]-01 exemplarily for both instance sizes.



(a) Optimal solution of I-n30-f[b]-w[b]-01.

(b) Optimal solution of I-n60-f[b]-w[b]-01.

Figure 1.3.: Exemplary optimal solutions for the minimization of total ergonomic strains.

In Figure 1.3, PCs are colored in a red-green color scheme according to the contained items' pick frequencies. Additionally, rectangular markings on the PCs' surfaces indicate the respective item weights on a grey scale. Empty PCs, which are only used to support regular PCs on their top, are colored grey on their entire surface.

For both instance sizes, pick frequency is the dominant attribute that influences PC placement. More frequently picked items strongly tend to be stowed in PCs located closer to the base.

However, due to the fact that Figure 1.3 depicts optimal solutions for the minimization of total ergonomic strains, the distances between PCs and the base are not solely dependent on item pick frequencies, but also (weakly) on the items' weights. Both observations, the high impact of pick frequencies and the low importance of item weights, are in accordance with our previous findings, for which we already discussed the underlying causes.

In addition, with respect to their pick frequencies, PCs are assigned approximately symmetrically in p- and q-direction, with the symmetry lines crossing at the base. Since the base tends to be located slightly above $r = \frac{1}{2}\left((l^f + s)(\Delta_q^* - 1) - s\right)$ in the optimal solution, the items contained in PCs in the first q-row (that is farthest away from the base) are among the least frequently picked.

1.6. Conclusion

This paper considered the problem of optimizing the layout and storage assignment of U-shaped order picking zones built from pallet cages, with the objective of minimizing either total walking distances or total ergonomic strains while picking. We further determined the optimal position of a base where picked items are dropped off. We assumed that the set of items that should be stored in the U-zone are given. Both versions of the problem were formalized as a MIP with different objective functions. Concerning the second objective, ergonomic strains were considered in a generalized manner allowing for different methods to be employed for quantifying them. This paper used the energy expenditure prediction model of Garg et al. (1978) for illustrative purposes.

We proposed an exact solution procedure based on the disaggregation of the problem into two consecutive problems that can be solved in hierarchical order. First, feasible U-zones and base positions are determined, and secondly, as a sub-problem, a storage assignment problem (formalized as a linear assignment problem) is solved for every respective layout and base position using the Hungarian algorithm. The resulting exact solution procedure has a polynomial asymptotic runtime of $O\left(\frac{1}{16}\frac{l^f+s}{\delta_r}n^5\right)$, where $l^f + s$ is a constant term, independent of the problem size, and δ_r is the numerical precision for determining the base position.

A computational study demonstrated the efficiency of our solution procedure and helped gaining insights into the properties of optimal solutions. Beyond that, we compared solutions optimized for both objectives considered in this paper, namely minimizing total walking distances or total ergonomic strains. The central findings of the computational experiments can be summarized as

follows.

- Our proposed solution procedure is sufficient to solve problems of realistic size in acceptable runtime. Instances with a problem size of 30 pallet cages were solved in 1.3 s on average. Instances with a problem size of 60 pallet cages took, in the mean, 40.5 s, and instances with a problem size of 100 pallet cages required 491.9 s on average to be solved.

- Optimized total ergonomic strains (measured via the worker's energy expenditure) can be reduced by between 17% and 34% compared to average randomly generated solutions. Likewise, total travel time can be reduced by between 33% and 50% compared to an average solution.

- Compared to solutions derived by simple heuristics that are often used to configure U-zones in practice, optimal solutions yield between 6% to 10% lower ergonomic strains and between 13% and 18% shorter total walking distances.

- Optimizing either total ergonomic strain or total traveling times yields close-to-optimal results for the respective other objective with optimality gaps below 1%. Hence, for the situation considered in this paper, both objectives are only marginally conflicting.

- The contribution of a certain item type to the total ergonomic strain mainly depends on the item's pick frequency. The influence of item weights is only minor (which we attribute to some extent to the ergonomic assessment method used in this paper). The contribution of a certain item type to the total traveling distance solely depends on the item's pick frequency.

- The higher an item's contribution to the respective objective, the closer it is assigned to the base. Since an item's contribution to either of the objectives is primarily/solely dependent on its pick frequency, both objectives result in similar optimal solutions.

- The optimal layout of the U-zone for both of the objectives strongly tends to be as narrow as possible in horizontal direction (i.e., such that the U's opening is as narrow as possible). Additionally, this optimizes the U's spacial utilization, i.e. the planar space enclosed by the U-shape.

- The optimal position of the base strongly tends to be aligned with the middle of pallet cages in horizontal direction. Additionally, the base is most of the time located close to the U-shape's middle in vertical direction with a slight shift away from the U's opening.

Three main research opportunities follow from the research presented in the work at hand. First and foremost, research should validate and possibly refine the evaluation of ergonomic strains resulting from the order picking activities. In this paper, we restricted ourselves to evaluating ergonomic strain via the energy expenditure prediction model of Garg et al. (1978). Although this assessment method is well-established in various fields of application including the evaluation of order picking activities (cf. Section 1.2), it also has several shortcomings. The most severe deficit in the case of this paper is that it neglects muscle fatigue of the back and upper extremities. Future research should, therefore, assess ergonomic strains during order picking activities with more refined methods.

Another option for future research lies in the extension and adoption of USLPP. A minor extension could consider the ergonomic strain resulting from placing the base at different positions (although we expect the influence to be marginal). Further adaptations could be made by considering alternative layouts, e.g. in the shape of a C. The model could also be adapted to consider pallet cages of half the regular height (cf., for example, Blackwoods, 2019) or regular shelves arranged in the shape of a U. However, for regular shelves with two distinct rack heights, the current model is already suitable.

Finally, future research should consider the optimization of warehouses with U-shaped layouts from a more holistic point of view, e.g. by including the optimization of the number and shape of zones and the partitioning of items between zones, although, due to its complexity, we expect a holistic problem to be hard to solve exactly. Heuristic procedures that utilize the findings of this paper could be a viable approach to solving such holistic problems.

1.A. Appendix

In the following, we exemplarily demonstrate how ergonomic strains, in the form of energy expenditure rates, are calculated in this paper. A schematic representation of an exemplary picking task, decomposed into its comprising tasks, is depicted in Figure 1.4.

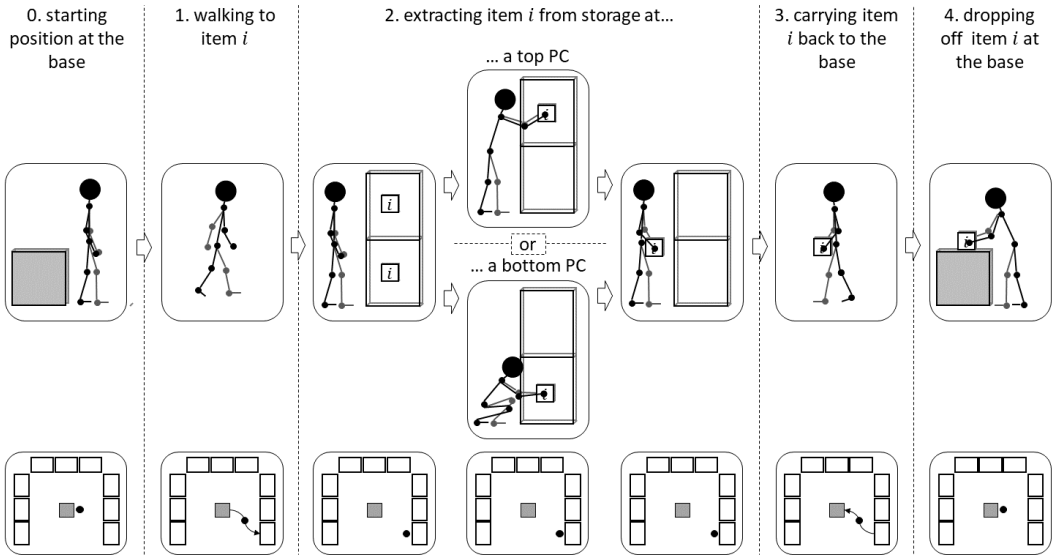


Figure 1.4.: Pictogram of the picking task.

In the exemplary picking task, we assume that the picker picks an item i stored at $(p, q) = (5, 1)$ (i.e. $x^t_{5,1,i} + x^b_{5,1,i} = 1$). For the sake of completeness, we demonstrate how the energy expenditure is calculated if i is stored either in a top or bottom PC. (We stress, however, that in a feasible solution of USSLP, for every item type we can only chose one of these two options.) The energy expenditure for picking item i in this exemplary case can then be derived as follows:

0. Initially, the picker stands in front of the base, ready to pick the next item. This marks the beginning of the picking process, but does not contribute to the energy expenditure.

1. As a first task, the picker walks a distance of $d_{p,q} = d_{5,1}$ to the storage location $(p, q) = (5, 1)$ of item i. We assume the picker walks at constant speed without accelerating or decelerating. Using the equations of Garg et al. (1978), we derive the worker's energy expenditure per distance walked as w^w, which depends on the worker's sex, body weight and walking speed. Multiplying the walking distance with the energy expenditure per distance walked yields the worker's energy expenditure for the first task as $d_{p,q}\, w^w$.

2. As soon as the picker arrives at the storage location, s/he begins extracting the item, which marks the second task. Depending on the item being either stored in a top PC ($x^t_{5,1,i} = 1$) or in a bottom PC ($x^b_{5,1,i} = 1$), the accumulated energy expenditure differs. However, in both cases, the extraction task can be split into two consecutive sub-task. Firstly, the picker brings his/her hands to the item to grab it. Secondly, s/he pulls/lifts the item out of the PC bringing it into a comfortable carrying position in front of his/her hips/frontal thighs. If the item is stowed in a bottom PC, the movement additionally includes squatting and sagittal bending of the upper body to reach the item as well as returning to an upright standing position when bringing the item to hip/thigh height. Using the equations of Garg et al. (1978), we are able to calculate the energy expenditures for both cases, w^t_i (top PC) and w^b_i (bottom PC), which further depend on the physical item weight and the picker's sex as well as body weight. Please note that both energy expenditures, w^t_i and w^b_i, are independent of the storage location of i in the (p, q)-plane.

3. After item i has been extracted, it is carried back to the base as a third task. We neglect any turning movements due to their minor contribution to the total energy expenditure. Analogous to the first task, the distance between the item's storage location and the base is $d_{p,q} = d_{5,1}$. Again, we assume that the picker walks at a constant speed without (de-)accelerating. Depending on the item's weight, we derive the energy expenditure per distance of carrying item i, w^c_i, using the equations of Garg et al. (1978). The energy expenditure to carry the item back to the base is then given by the term $d_{p,q}\, w^c_i$.

4. As a fourth task, the picker drops off the item at the base, where we assume that the item is (on average) deposited centered. Comparable to the second task, we split the fourth task into two sub-tasks. At first, the picker moves the item to the center of the base and releases his/her grip. Afterwards, s/he returns his/her upper extremities and upper body to a neutral upright position and turns to assume the initial position (enumerated with 0.), ready to begin the next picking task. Again, the energy expenditure of turning is neglected due to its minor contribution to the total energy expenditure. Depending on the item weight, we use the equations of Garg et al. (1978) to derive the energy expenditure for dropping off the item as w^d_i. Note that w^d_i is independent of both the item's storage location in the (p, q)-plane and of the item's storage in a top or bottom PC, rendering it irrelevant for both decisions. We nevertheless account for w^d_i to encompass total energy expenditure more accurately.

The total energy expenditure to pick item i then amounts to

$$\begin{cases} d_{p,q}\, w^w + w_i^t + d_{p,q}\, w_i^c + w_i^d & \text{for} \quad x_{5,1,i}^t = 1 \\ d_{p,q}\, w^w + w_i^b + d_{p,q}\, w_i^c + w_i^d & \text{for} \quad x_{5,1,i}^b = 1 \end{cases} = \begin{cases} d_{p,q}\, w_i^{w,c} + w_i^t + w_i^d & \text{for} \quad x_{5,1,i}^t = 1 \\ d_{p,q}\, w_i^{w,c} + w_i^b + w_i^d & \text{for} \quad x_{5,1,i}^b = 1 \end{cases}$$

with $w_i^{w,c} = w^w + w_i^c$. Since in USSLP, each item type is stored either in a top or at a bottom PC (i.e., $x_{5,1,i}^t + x_{5,1,i}^b = 1$), the resulting energy expenditure can be rewritten as

$$d_{p,q}\, w_i^{w,c} \left(x_{p,q,i}^b + x_{p,q,i}^t \right) + \left(w_i^t\, x_{p,q,i}^t + w_i^b\, x_{p,q,i} \right) + w_i^d$$

to equal the term in the objective function (1.5).

1.B. Bibliography

Aiyar, S., Ebeke, C., and Shao, X. (2017). The impact of workforce aging on european productivity. Technical Report WP/16/238, International Monetary Fund.

Battini, D., Delorme, X., Dolgui, A., Persona, A., and Sgarbossa, F. (2016a). Ergonomics in assembly line balancing based on energy expenditure: a multi-objective model. *International Journal of Production Research*, 54(3):824–845.

Battini, D., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2016b). Human energy expenditure in order picking storage assignment: A bi-objective method. *Computers & Industrial Engineering*, 94:147–157.

Bigos, S. J., Spengler, D. M., Martin, N. A., Zeh, J., Fisher, L., Nachemson, A., and Wang, M. (1986). Back injuries in industry: a retrospective study. II. injury factors. *Spine*, 11(3):246–251.

Borg, G. (1982). A category scale with ratio properties for intermodal and interindividual comparisons. *Psychophysical judgment and the process of perception*, pages 25–34.

Braam, I. T. J., van Dormolen, M., and Frings-Dresen, M. H. (1996). The work load of warehouse workers in three different working systems. *International Journal of Industrial Ergonomics*, 17(6):469–480.

Calzavara, M., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2017). Analysis of economic and ergonomic performance measures of different rack layouts in an order picking warehouse. *Computers & Industrial Engineering*, 111:527–536.

Calzavara, M., Glock, C. H., Grosse, E. H., and Sgarbossa, F. (2018). An integrated storage assignment method for manual order picking warehouses considering cost, workload and posture. *International Journal of Production Research*, pages 1–17.

Cardona, L. F., Rivera, L., and Martínez, H. J. (2012). Analytical study of the fishbone warehouse layout. *International Journal of Logistics Research and Applications*, 15(6):365–388.

Caron, F., Marchet, G., and Perego, A. (2000). Optimal layout in low-level picker-to-part systems. *International Journal of Production Research*, 38(1):101–117.

Chuang, Y.-F., Lee, H.-T., and Lai, Y.-C. (2012). Item-associated cluster assignment model on storage allocation problems. *Computers & Industrial Engineering*, 63(4):1171–1177.

Coyle, J. J., Bardi, E. J., Langley, C. J., et al. (1996). *The management of business logistics*, volume 6. West publishing company St Paul, MN.

De Koster, R. (2018). Automated and robotic warehouses: developments and research opportunities. *Logistics and Transport*, 38(2):33–40.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

Dempsey, P. G. (1998). A critical review of biomechanical, epidemiological, physiological and psychophysical criteria for designing manual materials handling tasks. *Ergonomics*, 41(1):73–88.

Diefenbach, H., Emde, S., and Glock, C. H. (2020). Loading tow trains ergonomically for just-in-time part supply. *European Journal of Operational Research*, 284(1):325–344.

Drury, C. G., Law, C.-H., and Pawenski, C. S. (1982). A survey of industrial box handling. *Human Factors*, 24(5):553–565.

Dukic, G. and Opetuk, T. (2008). Analysis of order-picking in warehouses with fishbone layout. *Proceedings of ICIL*, 8.

Ene, S., Küçükoğlu, İ., Aksoy, A., and Öztürk, N. (2016). A genetic algorithm for minimizing energy consumption in warehouses. *Energy*, 114:973–980.

European Commission (2017). The 2018 ageing report: Underlying assumptions and projection methodologies. Technical Report Institutional Paper 065, European Commission, Brussels.

European Pallet Association e.V. (2016). EPAL box pallet. Online: `https://www.epal-pallets.org/fileadmin/user_upload/ntg_package/images/Produktdownloads/Produktdatenbla__tter/GB/EPAL_Produktdatenblatt_GB_Gitterbox.pdf` [2019-01-14].

Fontana, M. E. and Nepomuceno, V. S. (2017). Multi-criteria approach for products classification and their storage location assignment. *The International Journal of Advanced Manufacturing Technology*, 88(9-12):3205–3216.

Francis, R. L. (1963). A note on the optimum location of new machines in existing plant layouts. *J. Indust. Engrg*, 14:57–59.

Gardner, L. I., Landsittel, D. P., and Nelson, N. A. (1999). Risk factors for back injury in 31,076 retail merchandise store workers. *American Journal of Epidemiology*, 150(8):825–833.

Garg, A. (2000). Ergonomic, biomechanical and physiological stresses from manual materials handling in grocery distribution centers. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 44, pages 431–432. SAGE Publications Sage CA: Los Angeles, CA.

Garg, A., Chaffin, D. B., and Herrin, G. D. (1978). Prediction of metabolic rates for manual materials handling jobs. *American Industrial Hygiene Association Journal*, 39(8):661–674.

Glock, C. H. and Grosse, E. H. (2012). Storage policies and order picking strategies in U-shaped order-picking systems with a movable base. *International Journal of Production Research*, 50(16):4344–4357.

Glock, C. H., Grosse, E. H., Abedinnia, H., and Emde, S. (2019). An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. *European Journal of Operational Research*, 273(2):516–534.

Goetschalckx, M. and Ratliff, H. D. (1988). An efficient algorithm to cluster order picking items in a wide aisle. *Engineering Costs and Production Economics*, 13(4):263–271.

Grosse, E. H. and Glock, C. H. (2013). An experimental investigation of learning effects in order picking systems. *Journal of Manufacturing Technology Management*, 24(6):850–872.

Grosse, E. H. and Glock, C. H. (2015). The effect of worker learning on manual order picking processes. *International Journal of Production Economics*, 170:882–890.

Grosse, E. H., Glock, C. H., and Jaber, M. Y. (2013). The effect of worker learning and forgetting on storage reassignment decisions in order picking systems. *Computers & Industrial Engineering*, 66(4):653–662.

Grosse, E. H., Glock, C. H., Jaber, M. Y., and Neumann, W. P. (2015). Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, 53(3):695–717.

Grosse, E. H., Glock, C. H., and Neumann, W. P. (2017). Human factors in order picking: a content analysis of the literature. *International Journal of Production Research*, 55(5):1260–1276.

Gu, J., Goetschalckx, M., and McGinnis, L. F. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1):1–21.

Gu, J., Goetschalckx, M., and McGinnis, L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539–549.

Hausman, W. H., Schwarz, L. B., and Graves, S. C. (1976). Optimal storage assignment in automatic warehousing systems. *Management science*, 22(6):629–638.

Henn, S., Koch, S., Gerking, H., and Wäscher, G. (2013). A U-shaped layout for manual order-picking systems. *Logistics Research*, 6(4):245–261.

Heragu, S. S., Du, L., Mantel, R. J., and Schuur, P. C. (2005). Mathematical model for warehouse design and product allocation. *International Journal of Production Research*, 43(2):327–338.

Institute of Medicine and National Research Council, NRC (2001). *Musculoskeletal Disorders and the Workplace: Low Back and Upper Extremities*. The National Academies Press, Washington, DC.

J. Blackwood & Son Pty Ltd (2019). Cage pallet half height 650mm. Online: `https://www.blackwoods.com.au/part/04044376/cage-pallet-half-height-650mm` [2019-02-04].

Jarvis, J. M. and McDowell, E. D. (1991). Optimal product layout in an order picking warehouse. *IIE transactions*, 23(1):93–102.

Kallina, C. and Lynn, J. (1976). Application of the cube-per-order index rule for stock location in a distribution warehouse. *Interfaces*, 7(1):37–46.

Kovács, A. (2011). Optimizing the storage assignment in a warehouse served by milkrun logistics. *International Journal of Production Economics*, 133(1):312–318.

Kusiak, A. and Heragu, S. S. (1987). The facility layout problem. *European Journal of Operational Research*, 29(3):229–251.

Larco, J. A., de Koster, R., Roodbergen, K. J., and Dul, J. (2017). Managing warehouse efficiency and worker discomfort through enhanced storage assignment decisions. *International Journal of Production Research*, 55(21):6407–6422.

Larsson, B., Søgaard, K., and Rosendal, L. (2007). Work related neck-shoulder pain: a review on magnitude, risk factors, biochemical characteristics, clinical picture and preventive interventions. *Best Practice & Research Clinical Rheumatology*, 21(3):447–463.

Lavender, S. A., Marras, W. S., Ferguson, S. A., Splittstoesser, R. E., and Yang, G. (2012). Developing physical exposure-based back injury risk models applicable to manual handling jobs in distribution centers. *Journal of Occupational and Environmental Hygiene*, 9(7):450–459.

Levine, J. A. (2005). Measurement of energy expenditure. *Public health nutrition*, 8(7a):1123–1132.

Liberty Mutual Research Institute for Safety, LMRIS (2018). *2018 Liberty Mutual Workplace Safety Index*. Liberty Mutual Research Institute for Safety, Hopkinton, MA.

Marras, W. S., Lavender, S. A., Ferguson, S. A., Splittstoesser, R. E., and Yang, G. (2010). Quantitative biomechanical workplace exposure measures: distribution centers. *Journal of Electromyography and Kinesiology*, 20(5):813–822.

Masae, M., Glock, C. H., and Grosse, E. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, 224:107564.

Mital, A. (2017). *Guide to manual materials handling*. CRC Press.

Moore, J. S. and Garg, A. (1995). The strain index: a proposed method to analyze jobs for risk of distal upper extremity disorders. *American Industrial Hygiene Association Journal*, 56(5):443–458.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Neumann, W. and Medbo, L. (2010). Ergonomic and technical aspects in the redesign of material supply systems: Big boxes vs. narrow bins. *International Journal of Industrial Ergonomics*, 40(5):541–548.

Neumann, W. P., Wells, R., and Norman, R. (1999). 4D WATBAK: adapting research tools and epidemiological findings to software for easy application by industrial personnel. In *Proceedings of the international conference on computer-aided ergonomics and safety, Barcelona, Spain*.

Otto, A., Boysen, N., Scholl, A., and Walter, R. (2017). Ergonomic workplace design in the fast pick area. *OR Spectrum*, 39(4):945–975.

Otto, A. and Scholl, A. (2011). Incorporating ergonomic risks into assembly line balancing. *European Journal of Operational Research*, 212(2):277–286.

Owen, S. H. and Daskin, M. S. (1998). Strategic facility location: A review. *European Journal of Operational Research*, 111(3):423–447.

Pan, J. C.-H., Shih, P.-H., Wu, M.-H., and Lin, J.-H. (2015). A storage assignment heuristic method based on genetic algorithm for a pick-and-pass warehousing system. *Computers & Industrial Engineering*, 81:1–13.

Petersen, C. G., Siu, C., and Heiser, D. R. (2005). Improving order picking performance utilizing slotting and golden zone storage. *International Journal of Operations & Production Management*, 25(10):997–1012.

Pohl, L. M., Meller, R. D., and Gue, K. R. (2009). Optimizing fishbone aisles for dual-command operations in a warehouse. *Naval Research Logistics (NRL)*, 56(5):389–403.

Punnett, L. and Wegman, D. H. (2004). Work-related musculoskeletal disorders: the epidemiologic evidence and the debate. *Journal of Electromyography and Kinesiology*, 14(1):13–23.

Qin, K., Chen, F. Y., and Ma, L. (2015). Cutting down the travel distance of put systems at kunming international flower auction market. *International Journal of Production Research*, 53(12):3573–3585.

Rao, S. S. and Adil, G. K. (2013). Class-based storage with exact s-shaped traversal routeing in low-level picker-to-part systems. *International Journal of Production Research*, 51(16):4979–4996.

Reyes, J., Solano-Charris, E., and Montoya-Torres, J. (2019). The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, 10(2):199–224.

Richards, G. (2017). *Warehouse management: a complete guide to improving efficiency and minimizing costs in the modern warehouse*. Kogan Page Publishers.

Roodbergen, K. J., Sharp, G. P., and Vis, I. F. (2008). Designing the layout structure of manual order picking areas in warehouses. *IIE Transactions*, 40(11):1032–1045.

Roquelaure, Y., Ha, C., Rouillon, C., Fouquet, N., Leclerc, A., Descatha, A., Touranchet, A., Goldberg, M., Imbernon, E., and of Occupational Health Services of the Pays de la Loire Region, M. (2009). Risk factors for upper-extremity musculoskeletal disorders in the working population. *Arthritis Care & Research*, 61(10):1425–1434.

Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G.-J., Mantel, R., and Zijm, W. H. (2000). Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122(3):515–533.

Sadiq, M., Landers, T. L., and Don Taylor, G. (1996). An assignment algorithm for dynamic picking systems. *IIE transactions*, 28(8):607–616.

Schneider, E. and Irastorza, X. (2010). Work-related musculoskeletal disorders in the EU – facts and figures. Technical report, European Agency for Safety and Health at Work, Bilbao, Spain.

Sharma, S. and Shah, B. (2015). A proposed hybrid storage assignment framework: a case study. *International Journal of Productivity and Performance Management*, 64(6):870–892.

Snook, S. (1989). The control of low back disability: the role of management. *Manual material handling: Understanding and preventing back trauma*.

Spengler, D. M., Bigos, S. J., Martin, N. A., Zeh, J., Fisher, L., and Nachemson, A. (1986). Back injuries in industry: a retrospective study. I. overview and cost analysis. *Spine*, 11(3):241–245.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons.

Waters, T. R., Putz-Anderson, V., Garg, A., and Fine, L. J. (1993). Revised niosh equation for the design and evaluation of manual lifting tasks. *Ergonomics*, 36(7):749–776.

Wilke, H.-J., Neef, P., Caimi, M., Hoogland, T., and Claes, L. E. (1999). New in vivo measurements of pressures in the intervertebral disc in daily life. *Spine*, 24(8):755–762.

Paper 2: Loading tow trains ergonomically for just-in-time part supply[5]

Abstract: Faced with an aging workforce, many manufacturing companies consider alleviating the ergonomic strain of material handling on their workers increasingly important. This is one of the reasons why frequent small-lot deliveries of parts to the assembly stations on the shop floor via small electric delivery vehicles – so-called tow trains – have become widespread in many industries. Deploying tow trains, however, does not automatically ease the ergonomic burden on logistics workers, but requires careful stowage planning in addition. In this paper, we consider the following problem. Given a set of bins of differing weight to be carried by tow train to a given set of stations on the shop floor, where should each bin be stowed on the tow train such that it can be unloaded efficiently from an economic perspective while also minimizing the ergonomic strain during loading and unloading? We investigate the physiological stress of handling bins on different levels of a tow train wagon by applying an established ergonomic evaluation method from the human factors engineering literature. We model the ensuing optimization problem as a special type of assignment problem and propose suitable exact and heuristic solution methods. In a computational study, our approaches are shown to perform well, delivering optimal solutions for instances of realistic size within fractions of a second in many cases. We show that optimal stowage plans can significantly ease the physiological burden on the workforce without compromising economic efficiency. We also derive some insights into the ideal layout of the tow train from an ergonomics perspective.

Keywords: Assignment; Tow trains; Ergonomics; Generalized assignment problem; Part feeding

[5]This chapter has been published as: Diefenbach, H., Emde, S., and Glock, C. H. (2020). Loading tow trains ergonomically for just-in-time part supply. *European Journal of Operational Research*, 284(1), 325-344. DOI: https://doi.org/10.1016/j.ejor.2019.12.009

2.1. Introduction

In many industries, feeding parts from a warehouse to the assembly lines such that neither transport frequencies nor line-side inventories are excessive has become a major problem. This is due to, on the one hand, an extreme product variety (that results from the customization of mass products), and, on the other hand, very limited space on the shopfloor, which prohibits large inventories and intense shopfloor traffic. Taking parts in large lots (e.g., entire pallets) straight to the assembly line via industrial trucks has therefore become highly unattractive for many companies (e.g., Medbo, 2003, Boysen et al., 2015). Instead, tow train systems are often used. Tow trains consist of a small electric towing vehicle attached to a handful of wagons, as depicted in Figure 2.1. Some tow trains operate as automated guided vehicles with minimal human intervention, although most tow trains are still operated by a driver (Golz et al., 2012, Emde and Gendreau, 2017, Lieb et al., 2017). More often than not, in addition to driving the train, its operator is also responsible for loading the tow train at a central warehouse or a just-in-time "supermarket" and unloading it at the assembly line. An intersectoral study by Lieb et al. (2017) found that the tow train driver is responsible for loading the tow train in 63% and for unloading the tow train in 89% of the cases.

Figure 2.1.: Tow train without load[6].

Tow trains have a higher transport capacity than forklifts, allowing frequent small-lot deliveries of parts to multiple stations in one tour. Parts are typically pre-sorted and packed into small standard-size bins and delivered just-in-time, such that workers at the assembly line need not waste any time searching for or unpacking parts. One central advantage of this type of part feeding system is that it is supposed to make it easier for workers to handle parts in an ergonomic

[6]The picture "Routenzuglösung" belongs to SSI Schäfer and was downloaded from `http://www.ssi-schaefer.ua/uploads/pics/071_routenzug_02.jpg`. SSI Schäfer permitted the usage of this content in the context of this paper.

manner (e.g., Neumann and Medbo, 2010, Emde and Boysen, 2012b, Battini et al., 2013). It is obviously less stressful from an ergonomics perspective to handle small bins than entire pallets (Neumann and Medbo, 2010). This aspect is becoming increasingly important as many manufacturers struggle with an aging workforce (e.g., Otto and Scholl, 2011, Aiyar et al., 2017, European Commission, 2017).

Tow trains do not automatically make logistics processes more ergonomic, however. Specifically, the tow trains still need to be loaded at the depot and unloaded at the stations by human workers. This requires lifting and setting down many individual bins of differing weight. Although tow train wagons are often designed as gravity flow racks, such that bins inserted at one end of the wagon slide to the front by themselves, not all tiers of a wagon can be accessed with the same ease. Typically, from an ergonomics perspective, the middle level of a rack is the least stressful to (un-)load (Petersen et al., 2005). Reaching overhead or bending down causes more strain (see Section 2.6 for more details).

This study is motivated by a problem we encountered at the main production facility of a major German machine manufacturer. This company supplies its assembly lines from a central warehouse via a fleet of ten tow trains, attached to wagons equipped with gravity flow racks. Parts are picked just-in-time in multiple order picking stations from pallets and crates into standardized bins destined for specific workstations at the assembly line. The pickers place the finished bins onto an automated sortation conveyor, which funnels them to the correct tow train to be loaded by the operator. Once equipped, the tow train sets off on a milk run through the production facility on one of multiple fixed routes. The whole system is computerized, meaning that the pickers know which parts to pick when for which station, the sortation conveyor knows where to send which bin, and the tow train operators know when the tow train visits which stations on what route. All bin movements onto and from the tow train are logged via a barcode scanner. Material, once requested, can be shipped to final assembly within two to four hours. While the company is quite satisfied with the responsiveness and efficiency of the system, the ergonomic stress caused by (un-)loading heavy bins has so far been taken into account only in a rudimentary manner.

To the best of our knowledge, until now, optimization and planning models focusing on situations like the one described above have only considered time and cost as objectives (for a more detailed review see Section 2.2). In this context, this paper makes the following contributions. First, we investigate the physiological stress of handling bins on different levels of a tow train wagon by applying an established ergonomic evaluation method from the human factors engineering literature to the loading and unloading of tow trains we observed in practice. Second,

we formulate, analyze, and solve the optimization problem of how to stow bins on a tow train such that they are readily accessible in the right order during unloading and such that the total ergonomic strain on the workforce is minimized during both loading and unloading. Third, in a series of computational experiments we show the efficacy of our solution approaches and demonstrate that incorporating ergonomic objectives into tow train (un-)loading greatly eases everyday part feeding. We also derive some recommendations as to how to operate tow trains ergonomically.

2.2. Literature review

Although just-in-time in-house logistics is quite a new topic in scientific research, various issues concerning the logistics of tow trains and, on a more general level, in-plant part feeding, have received attention in recent years. Problems that companies operating tow trains usually encounter range from planning the optimal location of in-plant logistics areas ("supermarkets") on a strategic level, to operational problems such as the routing, scheduling, and loading of tow trains (Emde and Boysen, 2012a). Holistic approaches are developed by Choi and Lee (2002) and Golz et al. (2012), who consider tour planning, scheduling and loading simultaneously and provide heuristic solution procedures for these problems. An overview of in-house milk run problems is provided by Alnahhal et al. (2014), while Battini et al. (2013) and Boysen et al. (2015) give an overview of the supermarket concept specifically in the automotive industry, where it is the most common.

The supermarket location problem is discussed by Battini et al. (2010), Emde and Boysen (2012a) as well as Alnahhal and Noche (2015). These authors consider the objective of optimally placing supermarkets on the shop floor, such that assembly line stations are supplied at minimal total cost, where the costs consist of fixed expenditures for erecting additional supermarkets and distance-dependent tow train travel costs.

Vaidyanathan et al. (1999) and Emde and Schneider (2018) deal with the problem of routing tow trains. The objective considered in these papers is to minimize the weighted sum of tow trains in use as well as line-side stocks that need to be held at workstations to satisfy demands in-between tow train re-visits, with the latter depending on the processing time of the planned tow train routes.

Tow train scheduling is considered by Emde and Boysen (2012b), Fathi et al. (2016) as well as Emde and Gendreau (2017). Following the concept of lean production, the problem, as regarded

by these authors, consists of deciding when to execute which tow train tour such that line-side demands are fulfilled and line-side stocks are minimized.

Finally, Emde et al. (2012) consider the problem of loading tow trains with the objective of minimizing line-side stocks.

Our brief review of the literature shows that research on just-in-time in-house logistics, and especially on tow train operations, has recently gained momentum. Ergonomic aspects have, however, not yet been investigated in this context, despite the high amount of manual human work that is still associated with the loading and unloading of tow trains in practice today (see Lieb et al., 2017). The handling of heavy loads that may be associated with the (un-)loading of tow trains exposes the tow train drivers to an increased injury risk that the company may mitigate by taking account of ergonomic measures in planning tow train operations. We note that Boysen et al. (2015) already highlighted a few years ago that taking account of ergonomic aspects in decision support models for material handling is an important research gap that requires further investigation.

While ergonomic aspects have not been considered in the planning of tow train operations, there is a substantial amount of research in the human factors engineering literature that investigates the manual handling of materials on worker health and safety. In this stream of research, manual material handling (MMH) activities, such as the ones encountered in (un-)loading tow trains, have been shown to increase the workers' risk of developing muscular-skeletal disorders (MSD) (see e.g., Punnett and Wegman, 2004, Larsson et al., 2007, Roquelaure et al., 2009). Some researchers estimate that between 50% and 75% of all MSD cases are directly related to MMH activities (Lavender et al., 2012). In the EU, MSD are assumed to account for up to 58% of all work-related illnesses, amounting to an estimated annual cost of 2% of the gross national product in the European Union (Schneider and Irastorza, 2010).

In the context of order picking, which shares some similarities with the (un-)loading of tow trains with respect to the MMH activities involved (but not with respect to the managerial characteristics of the decision problem), the increased risk of developing MSD has been confirmed in various field studies (Braam et al., 1996, Gardner et al., 1999, Garg, 2000, Marras et al., 2010, Lavender et al., 2012). Even though the increased risk of developing MSD from MMH activities (and hence order picking) is undisputed in the literature, Neumann and Medbo (2010), Grosse et al. (2015) as well as Grosse et al. (2017) point out that ergonomic aspects have scarcely been considered in decision support models for such activities, and that much work remains to be done. A few notable exceptions are discussed in the following.

Petersen et al. (2005) mention that there is a "golden" zone for shelf heights (between hip and shoulder height), where picking items is less exhausting for workers. The authors also state that this aspect has not yet been accounted for in stowage location planning models. Only recently, ergonomic considerations have been integrated into mathematical optimization and decision support models.

Battini et al. (2016b) investigate the optimal storage location for items in a single aisle consisting of a single (long) shelf. The authors consider a bi-criterial objective function, which minimizes total picking time as well as total ergonomic strain, depending on the vertical as well as horizontal placement of items. Ergonomic strain is quantified by calculating energy expenditure rates based on the concept developed by Garg et al. (1978). Another recent example of an application of the method by Garg et al. (1978) to quantify the ergonomic strain of picking items from different shelf configurations is the paper of Calzavara et al. (2017).

Larco et al. (2017) consider the problem of optimally stowing items in a warehouse with the objectives of minimizing travel time and ergonomic strain. They formulate this problem as an assignment problem, where every assignment of an item to a stowage location leads to a specific travel time and ergonomic strain. Besides the stowage location in the planar dimension, the authors also take the location in the vertical dimension (i.e., shelf heights) into account. Both travel times and ergonomic strains are determined in empirical experiments.

Otto et al. (2017) consider an item-to-storage assignment problem in a fast pick area using gravity flow racks. The authors develop a tabu search heuristic in order to minimizing the pickers' ergonomic strains, which they quantify using two different approaches; the NIOSH lifting equation (Waters et al., 1993) and a predetermined motion energy system (Battini et al., 2016a) based on the model by Garg et al. (1978).

Some researchers have recently started to investigate ergonomics in line-side operations in an in-house logistics and production context. Neumann and Medbo (2010), for example, compare the use of EURO pallets to small containers in the Swedish automotive industry. The authors find that using narrow containers for line-side operations offers economic advantages due to lower item access times, and that this concept also substantially lowers ergonomic strain on workers handling the materials. Ergonomic strains are evaluated with the help of a biomechanical model that measures the load on the lumbar spine, shoulders and hands of the worker, implemented in the software 4D WATBAK (Neumann et al., 1999).

Palmerud et al. (2012) conduct a case study at a Swedish car manufacturer, where the authors compare the ergonomic strains of two alternative production strategies, namely long-cycle par-

allelised flow assembly and serial flow assembly. The second strategy is shown to lead to lower MMH stress. To assess ergonomic strains, the authors measure relevant values, such as step frequency, cardiovascular load and body posture, by means of video recordings and direct technical measurements on the workers. The measurements are then statistically refined and directly compared to assess differences in ergonomic strains between both production strategies.

Otto and Scholl (2011) integrate an ergonomic risk factor into an optimization model for assembly line balancing. They show that their model is suited for various different ergonomic risk evaluation approaches, namely the revised NIOSH equation (Waters et al., 1993), OCRA (Occhipinti, 1998) and EAWS (Schaub et al., 2013), but settle for OCRA in their final computational study.

To the best of our knowledge, such techniques have never been applied to stowage planning for tow train operations.

2.3. Problem description

Regarding the planning hierarchy, we assume that the problem of optimizing the bin stowage of a tow train with regard to ergonomic aspects follows last, after tours, schedules and loads have been set. This is due to the observations we made in practice and those reported in the literature, where operating the tow train economically is the primary goal. Therefore, optimizing tow train operations from an ergonomics point of view may most likely only be approved within those boundaries set by economically optimal plans and schedules.

The ergonomic tow train loading problem (ETTLP) consists of assigning a given set of bins to storage slots on the tow train. Let $B = \{1, \ldots, n\}$ be the set of bins to be stowed on the tow train. Each bin $j \in B$ is assigned to a specific workstation $s(j) \in \mathbb{N}^{\neq 0}$ on the shopfloor. Note that one station may of course receive multiple bins, i.e., $s(j)$ may be identical for multiple j. Further, multiple bins may contain the same type of items, for example because of the items' high demand and/or size. Nevertheless, those bins can still be treated individually. If multiple bins are destined for the same station, they must not be divided among multiple wagons because this causes confusion and additional walking effort during unloading. We assume this constraint is already considered when the tow trains' routes, tours and schedules are planned according to economic objectives, such that emerging problem instances of ETTLP are guaranteed to be feasible. Nevertheless, if we face an economically optimized loading plan where it is impossible to store bins destined for a certain station on the same wagon, we can split the station and the

associated set of bins into two (or multiple) artificial stations and corresponding sets of bins to make the respective instance of ETTLP feasible. However, we assume that such situations occur rarely in practice due to tow trains mostly being filled only partially to capacity.

Let $W = \{1, \dots, w\}$ be the set of wagons to be loaded. Each wagon $w \in W$ carries a shelf with racks at different heights. Each rack is divided into slots, each μ bins deep, meaning that the total capacity of the tow train is $\sum_{w \in W} m_w \cdot \mu$ bins. For additional clarification, a schematic representation of such a wagon is provided in Figure 2.6 in the Appendix. Note that in many practical applications, we can assume $m_w = m_{w'}$, $\forall w, w' \in W$, i.e., all wagons are identical. Without loss of generality, we assume that $n = \sum_{w \in W} m_w \cdot \mu$, i.e., there are exactly as many bins as there is space. Note that this can always be imposed by adding "ghost" bins that do not contribute to the objective and have to be delivered each to a different "ghost" station. Also note that we assume that the total number of slots $\sum_{w \in W} m_w \cdot \mu$ is polynomially bounded by the number n of (non-ghost) bins. We denote the set of slots as $P = \{1, \dots, m\}$, where $m = \sum_{w \in W} m_w$. Each slot $p \in P$ is located on one wagon $w(p) \in W$. Placing a bin $j \in B$ in a slot $p \in P$ causes ergonomic strain on the logistics worker, denoted as $e(j, p)$. We explain in more detail in Section 2.6 how $e(j, p)$ can be calculated in practice.

A solution is defined as a mapping $\rho : B \to P$ such that $\rho(j) = p$ if bin $j \in B$ is assigned to slot $p \in P$. We call a solution feasible if and only if it satisfies the following conditions.

- No slot is loaded over capacity, i.e., since we use "ghost" bins to fill up empty spaces, for all $p \in P$, it must hold that $|\{j \in B \mid \rho(j) = p\}| = \mu$.
- Bins that are destined for the same station must be on the same wagon, i.e., for each pair of bins $j, j' \in B$, it must hold that if $s(j) = s(j')$, then $w(\rho(j)) = w(\rho(j'))$.

The latter condition enables us to stop the tow train at each station such that the wagon that stores the bins destined for the respective station is located right in front of it. This implies that no workstation receives more bins than can be stowed on a single wagon. Enforcing this condition minimizes the walking distance of the worker unloading the tow train, which, firstly, minimizes the ergonomic strain that results from carrying loads, and, secondly, minimizes unloading times. Hence, we only consider stowage plans that do not worsen the tow train's economic performance (i.e., unloading time), so that they are likely to be accepted in practice.

We further assume that slots are operated on a first-in-first-out principle, meaning that the bin that is put on a slot first is also the first one to be removed from it. This is the case, e.g., for gravity flow racks as described in Section 2.1. At each station, bins destined for this station

should be directly available, without the need for reshuffling bins at any given slot. Provided that all bins are available at the beginning of the planning horizon, for any feasible solution of ETTLP, we assume that the tow train is loaded in such a way that reshuffles are avoided. Any feasible solution for ETTLP can be converted to a solution obeying the no-reshuffle condition via a sorting algorithm (sorting the bins in each slot according to their destined stations) in polynomial time (e.g., $O(m \cdot \mu \cdot \log(\mu))$ using a comparison-based sorting algorithm). Therefore, we need not explicitly enforce this condition in our model.

A feasible solution already ensures that bins are stacked such that they can be accessed quickly. However, a feasible stowage plan does not guarantee that logistics workers can access bins in an ergonomic manner. To account for this, we seek among all feasible solutions one which minimizes

$$\sum_{j \in B} e(j, \rho(j)). \tag{2.1}$$

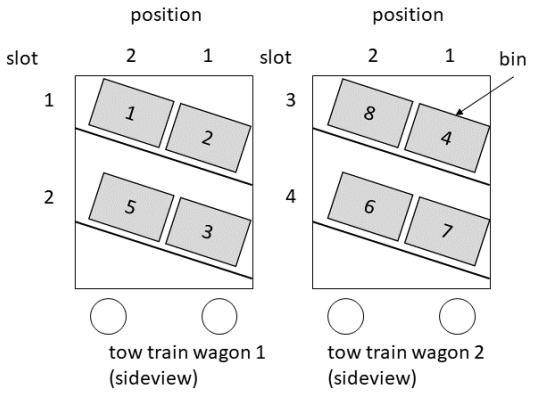
Note that, to formulate ETTLP concisely, we make the assumption that the capacity of the tow train is only limited by the number of bins it can carry, not by weight etc. This is usually a realistic assumption because tow trains are typically not used to carry bulky, heavy parts (Medbo, 2003). Moreover, we assume that the exact number of bins and their destinations on the shopfloor are known with certainty. This can be assumed to be given in many just-in-time assembly systems (e.g., Emde and Gendreau, 2017, Emde, 2017) and is a requirement of the IT control system at our industry partner.

2.3.1. Example of an ETTLP solution

Consider an example with $n = 8$ bins, which have to be loaded onto two wagons, each with $m_1 = m_2 = 2$ slots, $\mu = 2$ deep. Slots one and two are on wagon 1 (i.e., $w(1) = w(2) = 1$), and slots three and four are on wagon 2 (i.e., $w(3) = w(4) = 2$). The remaining parameters are given in Table 2.2a. A feasible and optimal solution is depicted in Figure 2.2b, corresponding to $\rho(1) = \rho(2) = 1$, $\rho(3) = \rho(5) = 2$, $\rho(4) = \rho(8) = 3$, $\rho(6) = \rho(7) = 4$. This leads to a total objective value of $1 + 2 + 2 + 3 + 7 + 2 + 9 + 11 = 37$.

j	1	2	3	4	5	6	7	8
$s_(j)$	3	1	1	2	3	5	4	5
$e_{j,1}$	1	2	1	3	5	1	7	11
$e_{j,2}$	3	4	2	5	7	2	9	15
$e_{j,3}$	1	2	1	3	5	1	7	11
$e_{j,4}$	3	4	2	5	7	2	9	15

(a) Example problem data.



(b) Optimal solution in the example.

Figure 2.2.: Example data and solution.

B	set of bins (index j)
P	set of slots (index p)
μ	capacity of each slot j
$s(j)$	station bin j is assigned to
$w(p)$	wagon slot p belongs to
$e(j,p)$	ergonomic strain if bin j is assigned to slot p
$x_{p,j}$	binary variable: 1, if bin j is placed in slot p; 0, otherwise

Table 2.1.: Notation.

2.3.2. MIP model for ETTLP

With the notation summarized in Table 2.1, we formalize ETTLP as a MIP model as follows.

$$\text{[ETTLP] Minimize } f(\boldsymbol{x}) = \sum_{p \in P} \sum_{j \in B} e(j,p) \cdot x_{p,j} \tag{2.2}$$

subject to

$$\sum_{p \in P} x_{p,j} = 1 \qquad \forall j \in B \tag{2.3}$$

$$\sum_{j \in B} x_{p,j} = \mu \qquad \forall p \in P \tag{2.4}$$

$$\sum_{p \in P} w(p) \cdot x_{p,j} = \sum_{p \in P} w(p) \cdot x_{p,j'} \qquad \forall j, j' \in B; j < j' \wedge s(j) = s(j') \tag{2.5}$$

$$x_{p,j} \in \{0,1\} \qquad \forall p \in P; j \in B \tag{2.6}$$

Objective function (2.2) minimizes the total ergonomic strain of the stowage plan. Constraints (2.3) enforce that each bin is assigned to exactly one slot. Similarly, Constraints (2.4) ensure that exactly μ bins (which can include "dummy" bins) are assigned to each slot, respectively. Equations (2.5) make it impossible for two bins destined for the same station to be on different wagons. Finally, (2.6) define the domain of the binary variables.

2.3.3. Time complexity

ETTLP is structurally similar to the generalized assignment problem (GAP, surveyed by Cattrysse and Van Wassenhove, 1992, Pentico, 2007, Burkard et al., 2012), which is well-known to be NP-hard. GAP is defined by a set of jobs and a set of agents, where agents have a limited capacity and assigning a job to an agent takes up capacity and incurs a certain cost. The goal is to assign each job to exactly one agent such that no agent's capacity is exceeded and the total cost is minimal.

If we interpret slots as agents, bins as jobs, and ergonomic strain as cost, ETTLP comes fairly close to this definition. However, there are a number of differences, which obfuscate the complexity status of ETTLP. On the one hand, ETTLP is a special case in that all slots/agents have the same capacity μ and every bin/job takes up the same space (1 unit). On the other hand, ETTLP is a generalization of GAP, because the slots are not independent of each other: if a bin bound for a specific station is assigned to a slot, then all other bins bound for the same station must be assigned to slots on the same wagon. Two papers we are aware of that study somewhat related constraints are those of Roy and Słowiński (2006) and Caramia and Guerriero (2010), who propose a GAP model where there are mutually exclusive jobs that must not be assigned to the same agent.

Constraints for the linear assignment problem that force disjoint pairs of assignment variables to take the same value are considered by Aboudi and Nemhauser (1991) and Aboudi et al. (1991), called the *couple constrained assignment problem*. This type of constraint is different from ETTLP, however, because we do not care about specific pairs of assignments, only that slots be on the same wagon. Felici and Mecoli (2007) consider the *assignment problem with preference conditions*, which is similar to the couple constrained assignment problem, except that setting pairs of assignment variables to different values does not make the solution infeasible but affects the objective value. The constraints of ETTLP have, to the best of our knowledge, not yet been considered.

Note that GAP is not NP-hard if the capacity of each agent is 1 and each job takes unit capacity;

in this case GAP is equivalent to the classic linear assignment problem, which is solvable in polynomial time (Burkard et al., 2012). However, this is not true for ETTLP, as we show in the following.

Theorem 2.2. *Finding a feasible solution to ETTLP is NP-complete in the strong sense, even if the slot capacity is restricted to* $\mu = 1$.

Proof. First, membership in NP is easy to see: an assignment of bins to slots constitutes a certificate. Under the assumption that the number of slots is polynomially bounded by the number of bins, such a certificate can obviously be verified in polynomial time.

We prove NP-hardness by pseudopolynomial reduction from 3-PARTITION, which is well known to be NP-hard in the strong sense (Garey and Johnson, 1979).

3-PARTITION is defined as follows: Given $3q$ positive integers a_i, $i = 1, \ldots, 3q$, and a positive integer Q with $Q/4 < a_i < Q/2$ and $\sum_{i=1}^{3q} a_i = qQ$, does there exist a partition of the set $\{1, \ldots, 3q\}$ into q sets $\{A_1, \ldots, A_q\}$, each having exactly three elements, such that $\sum_{i \in A_l} a_i = Q$ for each $l = 1, \ldots, q$?

We transform an instance I of 3-PARTITION to an instance I' of ETTLP in pseudopolynomial time as follows. We introduce a total of qQ bins, i.e., $B = \{1, \ldots, qQ\}$, and q wagons, each with Q slots, i.e., $P = \{1, \ldots, qQ\}$ and $w(p) = l, \forall l = 1, \ldots, q, p = (l-1) \cdot Q + 1, \ldots, l \cdot Q$. Each slot has a capacity of $\mu = 1$. The tow train supplies $3q$ stations, and each station's demand corresponds to one integer from the 3-PARTITION instance, i.e., $s(j) = i$, $\forall i = 1, \ldots, 3q$, $j = \sum_{i'=1}^{i-1} a_{i'} + 1, \ldots, \sum_{i'=1}^{i} a_{i'}$. We say that the bins $j \in B$ destined for station i (i.e., where $s(j) = i$) correspond with integer a_i from the 3-PARTITION instance.

A solution to 3-PARTITION instance I can be transformed to an ETTLP solution by assigning the bins corresponding to the integers in each set A_l, $l = 1, \ldots, q$, to one wagon each. The sum of integers in set A_l equals Q, which is also the number of slots on each wagon. Since the bins corresponding to the same integer are destined for the same station, the solution is feasible.

Conversely, a solution to ETTLP instance I' can also be converted to a solution for I. Each wagon holds exactly Q bins. Bins bound for the same station must not be split onto different wagons. The total capacity of all wagons is qQ, which is also the total number of bins. Therefore, the only feasible way of stowing the bins is to put exactly Q bins bound for exactly 3 destinations on each wagon. The correspondence with 3-PARTITION is hence apparent. An example is

depicted in Figure 2.3. □

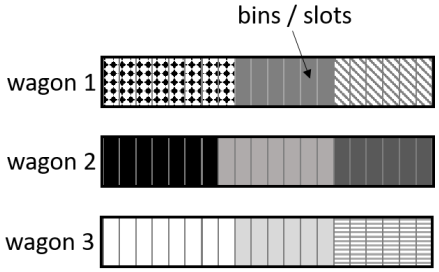


Figure 2.3.: Example ETTLP solution for a 3-PARTITION instance with $q = 3$, $Q = 20$ and integers $6, 6, 6, 6, 6, 7, 7, 8, 8$; bins of the same shade are destined for the same station.

2.4. Algorithms for ETTLP

Given Theorem 2.2, even finding a feasible solution is bound to be computationally challenging for instances of realistic size. Nevertheless, we propose an exact solution procedure based on the observation that if an assignment of bins to wagons is given, the remaining subproblem of placing the bins in individual slots can be modeled as a linear assignment problem. We explain the decomposition in Section 2.4.1 and propose an exact solution procedure based on dynamic programming in Section 2.4.2.

Even though our computational experiments (Section 2.5) bear out that our exact solution procedure is faster than a default solver (CPLEX) and is able to solve many large instances of realistic size in acceptable time, its runtime still grows exponentially. We therefore also propose a heuristic procedure in Section 2.4.3, exploiting the decomposability of ETTLP, to still be able to find solutions even for very large problem sizes.

2.4.1. Decomposition of ETTLP

To solve ETTLP, we decompose the problem into two stages: first, assigning bins to wagons and, second, assigning bins to slots. In the following sections we look at each decision individually, starting with the second problem.

2.4.1.1. Assigning a given set of bins to slots of a given wagon

For this subproblem, we assume that we are given a set $B_{w'} \subseteq B$ of bins to be put on some wagon w'. To ease notation, let $S = \{s(j) \mid j \in B\}$ be the set of all stations. Furthermore, let $P_{w'} = \{p \in P \mid w(p) = w'\}$ be the set of slots located on wagon w'. We assume that no more bins are assigned to wagon w' than it has space available, i.e., $|B_{w'}| \leq m_{w'} \cdot \mu$; otherwise, there is obviously no feasible assignment. If $|B_{w'}| < m_{w'} \cdot \mu$, we add "ghost" bins (with an ergonomic strain of zero) to $B_{w'}$ such that $|B_{w'}| = m_{w'} \cdot \mu$ becomes true. Finally, we define $\tilde{P}_{w'} = P_{w'} \times \{1, \ldots, \mu\}$ as the set of tuples $t = (p, k)$ of slots of set $P_{w'}$ and individual positions in those slots, respectively. We can then formulate the problem, to which we refer as the *bin to position assignment problem* (BTPAP), as a linear assignment problem, where the bins $j \in B_{w'}$ should be assigned to slot positions $t \in \tilde{P}_{w'}$ (Munkres, 1957). For completeness's sake, we provide an integer programming formulation of the problem in Section 2.6 in the Appendix. The problem can be solved in a polynomial runtime of $O((m_{w'} \cdot \mu)^3)$ using the formulation of Munkres (1957) of the Hungarian algorithm.

2.4.1.2. Assigning bins to wagons

The superordinate problem consists of assigning all bins to the tow train's wagons (but not to individual slots). A partition of bins among wagons is feasible if no two bins destined for the same station are in different sets, i.e., placed on different wagons, and no set contains more bins than a wagon has slots, i.e., no wagon is overloaded.

Seeing that bins destined for the same station cannot be split up anyway, instead of considering each bin individually, we can pool the bins according to their destined station. We define the set of bins destined for station s' as $J_{s'} = \{j \in B \mid s(j) = s'\}$, $\forall s' \in S$. We further define $\Lambda = \{\lambda_1, \ldots, \lambda_{|W|}\}$ as a $|W|$-partition of the set of stations S, with each λ_l, $l \in \{1, \ldots, |W|\}$, being a disjunct subset of S and $\bigcup_{l \in \{1,\ldots,|W|\}} \lambda_l = S$. The problem then becomes to find a $|W|$-partition Λ of the set of stations S, such that $\sum_{s \in \lambda_l} |J_s| \leq m_l \cdot \mu$, $\forall l \in \{1, \ldots, |W|\}$.

Note that to evaluate a given partition Λ, we need to solve the subordinate linear assignment problem from Section 2.4.1.1 for each of the wagonloads λ_l, $\forall l \in \{1, \ldots, |W|\}$. However, finding one feasible partition is adequate to obtain one feasible solution to the ETTLP, a fact which we will later use in our heuristic procedure.

In the general case where wagons of a single tow train posses different capacities, this subproblem is similar to variable-sized bin packing (Friesen and Langston, 1986), which is a general-

ization of bin packing where bins may have different capacities. Unlike bin packing, however, the exact number and capacity of bins (wagons) is given and not a variable. For the special case that all wagons have equal capacities, the problem is equivalent to the (decision version of) the regular bin packing problem with a given number of bins. The latter is in turn similar to the multi-way number partitioning problem, which consists of finding a $|W|$-partition of a set containing $|S|$ integer elements, such that the largest sum of the integers assigned to any of the partition's subsets is minimal. All of the above mentioned problems are known to be NP hard (or NP complete if referred to the decision versions) (Friesen and Langston, 1986, Korf, 2009).

2.4.2. An exact algorithm based on dynamic programming

To solve ETTLP exactly, we propose the following procedure. We use a dynamic programming scheme, based on the general idea formulated by Bellman (1954), to find the optimal partition of bins to wagons and assignment of bins to slots.

The dynamic program (DP) consists of $|W| + 1$ stages (with index $r = 0, \ldots, |W|$), each stage containing states (Γ), where $\Gamma \subseteq S$ denotes the set of stations whose bins have already been assigned to wagons. Starting from the initial stage $r = 0$ with state $\Gamma = \emptyset$, each successor in stage $r + 1$ is reached by adding a subset $\lambda \subseteq S \setminus \Gamma$ to Γ, i.e., the bins destined for the stations in set λ are assigned to wagon r. We only consider successors that still can lead to feasible assignments, and, thus, do not violate the following two criteria.

1. The number of bins destined for the stations contained in λ must not be greater than the wagon's capacity, i.e., $\sum_{s \in \lambda} |J_s| \leq m_r \cdot \mu$ must hold.

2. For the bins destined for the stations that are not yet assigned, there must still be enough space left on the remaining wagons, i.e., $\sum_{s \in S \setminus (\lambda \cup \Gamma)} |J_s| \leq \sum_{r'=r+1}^{|W|} m_{r'} \cdot \mu$ must hold.

Furthermore, we can accelerate our DP by breaking symmetries that may occur in many real-world instances, where all wagons of the tow train are identical. This implies that, first, every wagon has the same capacity, and second, each wagon holds ergonomically identical sets of slots, because the latter mostly depends on the vertical height of the respective slot; e.g., all slots on the middle shelf of each wagon may be equally accessible. In such cases, we do not care which of the identical wagons a given partition λ_l is assigned to. Therefore, if all wagons are equal, we break symmetries by only considering successors according to the following rule.

3. We only add λ to Γ if $\min \{s \in \lambda\} < \min \{s \in S \setminus (\Gamma \cup \lambda)\}$ holds true.

Using this rule, we enforce that the subsets in each generated partition are in a particular order. To be specific, λ has to contain a station with a smaller index than the smallest one of any λ' added at a later stage. By doing so, we avoid the need to evaluate every permutation of a given partition, which (in the symmetric case) all yield the same objective value. Instead we only evaluate one permutation; the one, whose subsets are ordered in the way we demand.

Let $V(\Gamma)$ be the set of states from which a transition to state Γ exists. The optimal objective value $h(\Gamma, r)$ can then be calculated recursively as

$$h(\Gamma, r) = \min_{\Gamma' \in V(\Gamma)} \left\{ h(\Gamma', r-1) + g^* \left(\bigcup_{s \in \Gamma \setminus \Gamma'} J_s, r \right) \right\},$$

with $h(\emptyset, 0) = 0$ for the initial state and g^* being the optimal objective value of the BTPAP from Section 2.4.1.1 for the given set of bins $\bigcup_{s \in \Gamma \setminus \Gamma'} J_s$ and the given wagon r. The objective value of a complete solution in final state $\Gamma = S$ equals $h(S, |W|)$, which is also the optimal objective value for ETTLP. We can obtain the corresponding optimal assignment by backward recovery along the optimal path.

Concerning the time complexity, note that the total number of different possible partitions is in $O(\{^{|S|}_{|W|}\})$, where $\{^{|S|}_{|W|}\}$ denotes the Stirling number of the second kind. The number of nodes in the DP graph is bounded by $O(|W| \cdot 2^{|S|})$, while the number of transitions cannot be greater than $O(|W| \cdot 2^{2|S|})$. For each transition, to calculate the contribution to the objective value, a linear assignment problem must be solved, which can be done in $O(n^3)$ time. Hence, the worst-case total number of steps required for DP is bounded by $O(n^3 \cdot |W| \cdot 2^{2n})$. Note, however, that due to the exclusion rules laid out above, the actual number of steps is usually much lower.

Example (cont.): The dynamic programming graph for the problem given in Section 2.3.1 is depicted in Figure 2.4. One optimal solution (all three possible solutions are optimal in this example) is bold, corresponding to the first wagon receiving the bins bound for stations 1 and 3, and the second wagon receiving the bins bound for the remaining stations 2, 4, and 5. This corresponds to the solution depicted in Figure 2.2b.

2.4.3. A GRASP metaheuristic

Seeing that the asymptotic runtime of our DP scheme is exponential, we propose a heuristic algorithm based on greedy randomized adaptive search (GRASP) to obtain good solutions for large instances in acceptable time. GRASP, as originally proposed by Feo and Resende (1995),

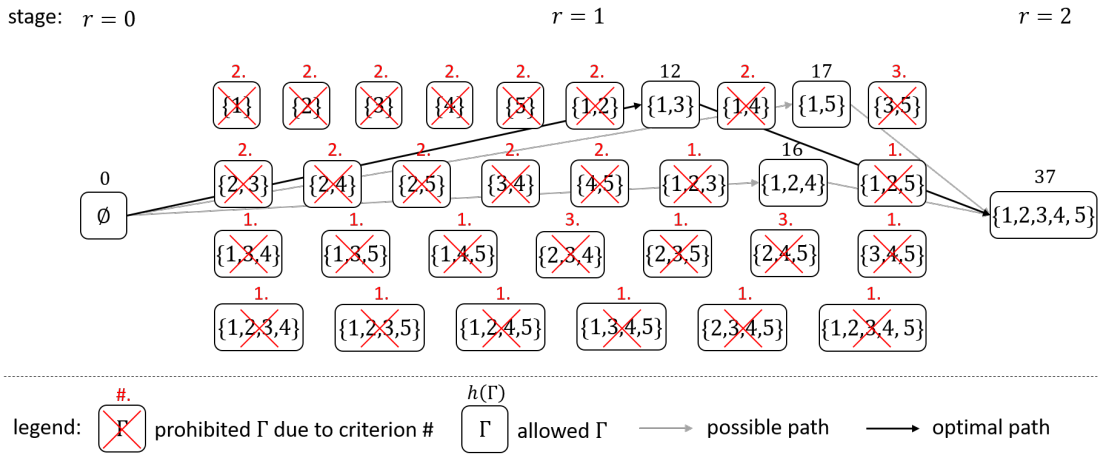


Figure 2.4.: Dynamic programming graph for the example given in Section 2.3.1.

consists of the following steps: first, an initial solution is obtained using a randomized constructive heuristic. Second, this solution is improved by performing a local search on it. Steps one and two are repeated several times until the stopping criterion is satisfied. Finally the best found solution is returned.

Adapting this scheme to ETTLP is not entirely straightforward because, by Theorem 2.2, even finding a feasible solution is already NP-hard. In Section 2.4.3.1, we describe how we generate initial solutions that are at least close to feasible. We repair and improve these solutions via an IP-based repair mechanism in Section 2.4.3.2, and put both components together in a GRASP framework in Section 2.4.3.3.

2.4.3.1. Randomized constructive heuristic

What makes ETTLP difficult is finding the best $|W|$-partition of S. By Theorem 2.2, even finding a single feasible partition is NP-hard. Therefore, we use the GRASP framework to obtain feasible partitions, for which we then find the optimal assignment in a successive step. To do this, we formulate a relaxed version of the partitioning problem from Section 2.4.1.2 by allowing the number of bins assigned to the wagons to be greater than their capacity, i.e., we accept partitions Λ for which the condition $\sum_{s\in\lambda_l} |J_s| \leq m_l \cdot \mu$, $\forall l \in \{1, \ldots, |W|\}$, does not hold.

To attain initial solutions that are at least close to feasible partitions for ETTLP, we propose the constructive heuristic outlined in Algorithm 2.1. Our proposed procedure is similar to the greedy

heuristic classically used in multi-way number partitioning. The greedy multi-way number partitioning heuristic first orders all integers according to non-increasing values. It then assigns them one by one to the subset with the currently smallest sum of integers until all integers are partitioned (Korf, 2009). This procedure is similar to the best fit decreasing heuristic commonly used to solve the bin packing problem. It has been shown to perform very well on average hard instances (Coffman et al., 1996) and, hence, appears suitable for our purpose. However, the procedure does not contain any randomization, which is needed for GRASP in order to obtain different initial solutions. We, therefore, alter the heuristic as follows.

First, we order all stations randomly, and thereby add the required randomness to the procedure. We go through the stations one-by-one to assign the bins destined for it to a wagon. Let $S_w \subseteq S$ be the set of stations whose bins have already been assigned to wagon w (initially, $S_w = \emptyset$). Furthermore, let s' be the station we are currently looking at. We use the following decision criteria to choose a wagon to assign station s' to. If there is a wagon w' that is filled exactly to capacity if all bins bound for station s' are added to the bins already assigned to that wagon, we choose w'. I.e., if $\sum_{s \in S_{w'} \cup \{s'\}} |J_s| = m_{w'} \cdot \mu$, we choose w'. If no such wagon exists, we choose the currently least filled one, i.e., we choose $w' = \arg\min_{w \in W} \left\{ \sum_{s \in S_w} |J_s| \right\}$. Ties are broken randomly. These steps are repeated until all bins are assigned.

Algorithm 2.1: Randomized constructive heuristic for the relaxed version of ETTLP.

Input: instance of ETTLP

$\zeta(s) := -1, \forall s \in S$; `// wagon to which bins bound for station` s `are assigned`

$\eta(w) := 0, \forall w \in W$; `// the number of bins currently assigned to wagon` w

foreach $s \in S$ *in random order* **do**

  $\Omega := \emptyset$; `// set of wagons for selection`

  for $w = 1$ **to** $|W|$ **do**

    if $\eta(w) + |J_s| = m_w \cdot \mu$ **then**

      $\Omega := \Omega \cup \{w\}$;

  if $\Omega = \emptyset$ **then**

    $\eta_{min} := \min_{w \in W} \{\eta(w)\}$;

    $\Omega := \{w \in W \mid \eta(w) = \eta_{min}\}$;

  $\zeta(s) := rand(w \in \Omega)$; `// random tie break`

  $\eta(\zeta(s)) := \eta(\zeta(s)) + |J_s|$;

Output: Assignment of stations to wagons $\boldsymbol{\zeta}$

2.4.3.2. Repairing infeasible solutions via integer programming

The solution constructed by Algorithm 2.1 may not be a feasible partition for ETTLP because individual wagons may be loaded over capacity. Thus, we present a repair algorithm to improve upon the initial solution and, thereby, hopefully make it a feasible partition for ETTLP. We propose an IP-based heuristic improvement method that takes advantage of the sophistication of modern off-the-shelf solvers, which are generally adept at quickly finding good solutions to integer programs.

δ_w	binary variable: 1, if the amount of bins assigned to wagon w exceeds its capacity; 0, otherwise
$z_{w,s}$	binary variable: 1, if the set of bins destined for station s is assigned to wagon w; 0, otherwise

Table 2.2.: Additional notation for RBTWP.

Let $\overline{\boldsymbol{\zeta}}$ be a solution obtained from Algorithm 2.1, and let $\overline{S}_w$ be the corresponding set of stations whose bins are assigned to wagon w, $\forall w \in W$. Assume that this solution is infeasible with regard to the non-relaxed problem, where the capacity of the wagons is limited. Let $\overline{W} = \left\{w \in W \mid \sum_{s \in \overline{S}_w} |J_s| > m_w \cdot \mu\right\}$ be the set of wagons whose capacity is exceeded. Using the notation in Table 2.2, we formulate the IP given below. We refer to this subproblem as the *relaxed bin to wagon problem* (RBTWP).

$$\text{[RBTWP] Minimize } \hbar(\boldsymbol{z}, \boldsymbol{\delta}) = \sum_{w \in \overline{W}} \delta_w \tag{2.7}$$

subject to

$$\sum_{w \in W} z_{w,s} = 1 \qquad \forall s \in S \tag{2.8}$$

$$\sum_{s \in S} |J_s| \cdot z_{w,s} - n \cdot \delta_w \leq m_w \cdot \mu \qquad \forall w \in \overline{W} \tag{2.9}$$

$$\sum_{s \in S} |J_s| \cdot z_{w,s} \leq m_w \cdot \mu \qquad \forall w \in W \setminus \overline{W} \tag{2.10}$$

$$z_{w,s} \in \{0, 1\} \qquad \forall w \in W; s \in S \tag{2.11}$$

$$\delta_w \in \{0, 1\} \qquad \forall w \in \overline{W} \tag{2.12}$$

The objective (2.7) is to minimize the number of wagons whose capacity is exceeded. Constraints (2.8) ensure that every set of bins destined for the same station is assigned to exactly one

wagon. Constraints (2.9) force δ_w to assume value 1 if the capacity is exceeded at wagon w, whereas Inequalities (2.10) enforce the capacity constraint for the non-critical stations. Finally, Constraints (2.11) and (2.12) define the domain of the decision variables.

We solve RBTWP with a modern default solver "warm" started from the solution denoted by $\overline{\boldsymbol{\zeta}}$. To ensure short computation times, we set a narrow time limit for the solver, which restricts it to only evaluate a few solutions. This allows us to use the full advantage of modern default solvers, which generally perform well at finding good (or even optimal) solutions quickly and take most of their time to prove optimality. Note that we do not restrict the solution space of RBTWP via any cuts but only by setting a restrictive time limit on the solution procedure. Our computational test shows this approach to work very well on all evaluated instances (see Section 2.5). However, if the problem size gets too large, additionally adding invalid cuts as proposed in the local branching approach by Fischetti and Lodi (2008) might be a reasonable extension of our algorithm. This turns out to not be necessary for even the largest instances we tested, however.

2.4.3.3. GRASP framework

Algorithm 2.2: GRASP heuristic for ETTLP.

Input: instance of ETTLP
$f^* := \infty$; `// best found objective value for ETTLP`
instance of RBTWP := create instance of RBTWP(instance of ETTLP) ;
while *stopping criterion not satisfied* **do**
  $\boldsymbol{z}$:= randomized constructive heuristic(instance of RBTWP);
  $\boldsymbol{z}$:= local search($\boldsymbol{z}$);
  if $h(\boldsymbol{z}) = 0$ **then** `// solution is feasible`
    $f := 0$;
    for $w \in W$ **do**
      $\boldsymbol{y}$:= solve BTPAP for given partition $\boldsymbol{z}$ and wagon w ;
      $f := f + g(\boldsymbol{y})$;
    if $f < f^*$ **then**
      store new best solution;
      $f^* := f$;
Output: ETTLP solution

We can now perform a classic GRASP to find solutions for RBTWP, such that each obtained solution that yields an objective value of zero is a feasible partition for ETTLP. Therefore, each time we find such a solution, we solve the emerging BTPAP linear assignment problem for each wagon (see Section 2.4.1.1) to get a viable solution for ETTLP. Else, if no feasible partition can

be found within the time limit of T, we scrap the current solution and perform the next iteration of our GRASP. The framework of this heuristic is outlined in Algorithm 2.2.

2.5. Computational study

In this section, we test the computational performance of our exact and heuristic solution procedures. We compare both procedures to each other as well as to a default solver (CPLEX). To do so, we randomly generate realistic test instances of various sizes, which we describe in detail in the following subsection. We also derive some managerial insights, namely to what extent pursuing ergonomic objectives can actually relieve the strain on the workforce compared to classic purely economic objectives, and how the layout and the filling of the tow train affects its ergonomic performance.

2.5.1. Benchmark instances and ergonomic assessment

Since, to the best of our knowledge, there has not yet been any research into stowing bins ergonomically on tow trains, there are no benchmark instances available to test our proposed procedures. Therefore, we create new ones. In the following, we first explain the general procedure of generating our instances. Afterwards, we explain how we obtained realistic ergonomic values.

2.5.1.1. Generating instances

ETTLP is the last of a sequence of optimization problems concerning tow trains. At first, routing, scheduling, and loading problems have to be solved (see Section 2.2). The results of those problems then define the input for ETTLP. In practice, tow train routing, scheduling and loading are solved in a way that leaves enough flexibility to properly stow the bins on the tow train. We therefore create our instances by first randomly generating a tow train stowage plan, imitating the way tow trains are typically loaded in practice, that is, without considering ergonomic stress. We call this the "default" or "status-quo" stowage plan.

The size of an instance is defined by parameters n, the number of bins, m, the number of slots, μ, the capacity per slot, $|W|$, the number of wagons and $|S|$, the number of stations. Note, that $n \leq \mu \cdot m$ must hold, or else the tow train's capacity is exceeded. Further, if $n = \mu \cdot m$, the

tow train is filled completely. Else, if $n < \mu \cdot m$, the tow train is only partially filled. Beyond that, $\frac{m}{|W|}$ is the number of slots per wagon and must be commensurate with the wagons' layout. For example, if each wagon has three different shelf heights, with each shelf holding three slots, giving a total of 9 slots per wagon, $\frac{m}{|W|} = 9$ must hold.

An instance is generated by assigning every bin $j = 1, \ldots, n$ to a random slot and a random position within this slot that has not yet been filled. We then assign every station $s = 1, \ldots, |S|$ randomly to a wagon, where we make sure that to every wagon at least one station is assigned. Stations are attached to bins by randomly choosing one of the stations assigned to the respective wagon, ensuring that every station is assigned to at least one bin, and every bin to exactly one station.

Finally, we need to tackle the ergonomic strain. Since the ergonomic strain caused by placing a certain bin in a certain slot is dependent on the weight of the bin and the vertical location of the slot, we need to assign weights to every bin and heights to every slot. For every bin we randomly draw a weight from a discrete set of weights Θ. Vertical heights are assigned to slots according to the layout of the tow train's wagons. In each wagon, the slot with the lowest index is located on the bottom shelf on the left. With increasing indices, the slots are first located further to the right until the shelf has no slots left. After that, slots are located on the next higher shelf until the slot with the highest index is located at the right end of the top shelf of each wagon, respectively. Note that bin weights and slot heights are only auxiliary parameters we use to calculate the ergonomic strains $e(j, p), \forall j \in J; p \in P$. We calculate ergonomic strains for two representative workers, one male and one female, via the energy expenditure prediction model by Garg et al. (1978). We choose this approach mainly due to its suitability to evaluate the tasks considered in this paper and its widespread use for similar problems in the literature (see Section 2.2). A more detailed explanation of the calculation of the ergonomic strains and possible alternative approaches are presented in the Appendix in Section 2.6.

In accordance with our observations in practice, we choose our default wagon to have three different shelf heights, 30 Centimeter, 95 Centimeter and 160 Centimeter at the side from which the shelf is unloaded, with additional 15 Centimeter of height at the opposite side since the shelf is constructed as a gravity flow rack. Each shelf holds three slots next to each other, and each slot has space for $\mu = 3$ bins, such that a wagon can hold $3 \cdot 3 \cdot 3 = 27$ bins in total. Bin weights are drawn from $\Theta = \{5, 10, 15, 20, 30\}$ Kilogram, since those are quite representative weights in industrial environments (e.g., Drury et al., 1982). Bin measurements are set to 30 Centimeter in width, 40 Centimeter in length and 21.3 Centimeter height, which is in accordance with a standardized recommendation of the German automotive industry (VDA, 2018).

Once we are done with those assignments, we can derive all parameters needed for an instance of ETTLP, namely B, P, μ, $s(j)$, $\forall j \in J$, $w(p)$, $\forall p \in P$, and $e(j,p)$, $\forall j \in J; p \in P$. To test the performance of our proposed solution procedures we generate and solve instances of various realistic sizes. According to Boysen et al. (2015), in practice, the number of wagons per tow train rarely exceeds more than a handful. A production facility may hold up to several hundreds of productions stations (Emde and Boysen, 2012a). However, typically, multiple tow trains are employed, such that each tow train visits only a fraction of all stations and not every station a tow train is assigned to is visited in every tour (Emde and Gendreau, 2017). For their computational study, Emde and Gendreau (2017), for example, assume that a tow train visits between seven and 14 station per tour, based on their observations in the German automotive industry. Emde and Schneider (2018) report in their computational study that a tow train visits on average five to nine stations per tour in an optimal route. Therefore, to represent different scenarios, we create instances of four different sizes, small, medium, large and very large with $|W| = 4$ and $|S| = 6$, $|W| = 6$ and $|S| = 9$, $|W| = 7$ and $|S| = 16$ as well as $|W| = 9$ and $|S| = 24$, respectively. We note that for the large and very large instances, we choose the number of wagons $|W|$ such that the Stirling number becomes maximal for the selected value of $|S|$ in order to especially challenge our proposed DP (see Section 2.4.2).

The number of slots per instance is derived by multiplying $|W|$ with a factor of 9, since this is the amount of slots our default tow train wagon holds. With each slot having a capacity of $\mu = 3$, the maximum number of bins the tow train can load amounts to $|W| \cdot 9 \cdot 3$. Since tow trains are commonly not loaded to their full capacity, for each instance size, we create two kinds of instances: one, where every single slot of the tow train is occupied by bins, and a more realistic one, where the tow train is loaded only to about 80% capacity. Note that instances of the latter kind do in fact contain the full number of bins, where we fill the remaining 20% with "ghost" bins, as described in Section 2.3. For each instance size and loading kind, we create ten random instances.

The instances described above assume that all tow train wagons are identical and that (un-)loading a certain bin from/onto each slot located on the same rack causes identical ergonomic strains. Though both assumptions are reasonable for a lot of practical cases, they are not mandatory. It is possible to attach different kinds of wagons to the same tow train, and ergonomic strains can vary between slots on the same rack (see Section 2.6). In order to evaluate the performance of our solution procedures for such cases, we create ten additional instances to which we refer as randomized instances in the following. The randomized instances are not meant to emulate any real-world situation. Instead, they are designed to be as challenging as possible to provide an upper estimate on the solution procedures' performances for any practical situation

that may be encountered. We assume a tow train with $|W| = 6$ wagons, set the tow train capacity to $n = 168$ and assume an 80% filling (cf. the medium-sized instances). Each wagon's individual capacity is randomized and each ergonomic strain for (un-)loading a certain bin from/onto a certain slot is drawn randomly from a uniform distribution in the same realm as the previously derived real-world ergonomic strains. The number of stations visited by the tow train is set to random values between ten and 16.

Each instance is named in the following way. From left to right, first, a letter characterizes the instance size, small (S), medium (M), large (L), very large (XL) and randomized (R). Then, another letter denotes if a representative male (m) or female (f) worker is used for determining ergonomic strains. (This is omitted in the randomized instances.) Afterwards, a set of numbers characterizes the instance's parameters in the following order: the number of bins n, followed by the number of non-"ghost" bins in brackets, the number of slots m, the slots capacity μ, the number of wagons $|W|$ and finally the number of stations $|S|$. The two right-most digits are continuous counting numbers to identify each instance.

In total, we generate 230 instances, which are available from `http://doi.org/10.5281/zenodo.3515732`.

2.5.2. Computational results

2.5.2.1. Computational performance

We solve each instance with our proposed DP and GRASP, and the MIP model with CPLEX (version 12.8) at default settings, all implemented in C#. Computational testing was performed on an Intel Core i7-6700 CPU @ 3.40 Gigahertz and with 8 Gigabyte of RAM. For all solution procedures, we set a maximum runtime of 3600 seconds (i.e., 1 hour). For GRASP, we set the number of iterations to 50 and the maximum runtime for the local search at each iteration to $T = 1$ second for all instances. Detailed reports of the results are available from `http://doi.org/10.5281/zenodo.3515732`. In the following, however, we mainly present the results of those instances that use the anthropometric measurements of a representative male worker for quantifying ergonomic strain. If not stated otherwise explicitly, we always refer to those instances. The main purpose of the instances, where the anthropometric measurements of an average female worker are used to quantify ergonomic strain, is to demonstrate, by comparison with the former instances, that the achievable relative reduction in ergonomic strains by (un-)loading a tow train ergonomically is virtually independent of the worker's an-

thropometrics; hence that our evaluation is generalizable. The results for the instances that use the anthropometric measurements of the representative male worker are summarized in Tables 2.3, 2.4 and 2.6. A comparison between the relative reduction in ergonomic strains for anthropometric measurements of either a representative male or female worker is given in Table 2.5.

Table 2.3 summarizes the results for instances whose sizes are comparable to problem sizes in practical application. While CPLEX is able to prove optimality for all small instances within a runtime of 3600 seconds, it fails to do so for most medium and all large instances, though the optimality gaps reported by CPLEX at the point of termination are quite low. DP, on the other hand, performs much faster. It can solve all small and medium instances to proven optimality in not more than a tenth of a second; all large instances were solved in less then 35 seconds. GRASP performs well, too, solving every instance in less then two seconds. The relative optimality gaps are zero for all 100% filled instances and about half of the 80% filled instances. In total, only one optimality gap exceeds 1%.

According to our tests, if the tow train is loaded to its maximum capacity, optimizing the bin stowage yields an averaged ergonomic improvement of about 11%. In the more realistic case of an 80% loaded tow train, optimizing the bin placement improves the average ergonomic assessment by between about 14% to 15% on average. Taking into account that the economic performance of the solutions is identical (both solutions are feasible) and that no investment in any further equipment is needed, these results may be quite relevant for those practical cases where workers have to handle high loads over the course of a workday and where they are hence exposed to an increased risk of job-related injuries.

The instances designed to challenge our solution procedures, especially the proposed DP, are summarized in Table 2.4. CPLEX fails to solve all 100% filled very large instances, but is able to solve all 80% filled ones – some even to proven optimality. Contrarily, DP is able to solve the former within at most two minutes, but fails to solve the latter. This is due to the fact that DP needs to evaluate a much larger number of feasible partitions of bins to wagons in the 80% filled instances than in the completely filled ones. GRASP is able to solve every very large instance in below three seconds. The heuristic solutions are within at most 1.28% from the optimum.

All solution procedures were able to solve all randomized instances, except for DP, which fails to solve instance R-162(130)x54x3-6x16-10 within one hour. While DP solves instances with a lower amount of stations (as denoted by the next-to-last number in the instance names) more efficiently than CPLEX, the latter performs better on instances with more stations. The optimality gaps of the solutions found by GRASP range from 0.99% to 6.29% and tend to grow with

instances			CPLEX				DP			GRASP		
label	primary value	impr. (in %)	value	gap (in %)	C. gap (in %)	time (in s)	value	gap (in %)	time (in s)	value	gap (in %)	time (in s)
S-m-108(108)x36x3-4x6-01	137.09	9.87	123.56	0.00	0.01	68.85	123.56	0.00	0.00	123.56	0.00	0.03
S-m-108(108)x36x3-4x6-02	142.10	12.96	123.69	0.00	0.01	20.04	123.69	0.00	0.00	123.69	0.00	0.12
S-m-108(108)x36x3-4x6-03	136.20	10.78	121.51	0.00	0.00	0.99	121.51	0.00	0.00	121.51	0.00	0.01
S-m-108(108)x36x3-4x6-04	138.33	12.64	120.85	0.00	0.00	7.12	120.85	0.00	0.00	120.85	0.00	0.01
S-m-108(108)x36x3-4x6-05	130.65	11.84	115.18	0.00	0.00	6.50	115.18	0.00	0.00	115.18	0.00	0.03
S-m-108(108)x36x3-4x6-06	136.15	9.18	123.65	0.00	0.01	140.06	123.65	0.00	0.00	123.65	0.00	0.01
S-m-108(108)x36x3-4x6-07	138.71	11.19	123.20	0.00	0.01	8.46	123.20	0.00	0.00	123.20	0.00	0.12
S-m-108(108)x36x3-4x6-08	142.20	13.11	123.55	0.00	0.00	0.67	123.55	0.00	0.00	123.55	0.00	0.06
S-m-108(108)x36x3-4x6-09	142.68	10.54	127.65	0.00	0.01	110.11	127.65	0.00	0.00	127.65	0.00	0.03
S-m-108(108)x36x3-4x6-10	142.50	13.26	123.61	0.00	0.01	26.14	123.61	0.00	0.00	123.61	0.00	0.14
mean	**138.66**	**11.54**	**122.65**	**0.00**	**0.01**	**38.89**	**122.65**	**0.00**	**0.00**	**122.65**	**0.00**	**0.06**
S-m-108(86)x36x3-4x6-01	109.88	15.12	93.26	0.00	0.01	73.56	93.26	0.00	0.00	93.26	0.00	0.34
S-m-108(86)x36x3-4x6-02	111.17	14.43	95.13	0.00	0.01	216.93	95.13	0.00	0.00	95.13	0.00	0.33
S-m-108(86)x36x3-4x6-03	107.63	14.32	92.22	0.00	0.01	104.06	92.22	0.00	0.00	92.40	0.19	0.33
S-m-108(86)x36x3-4x6-04	107.98	14.64	92.18	0.00	0.01	63.96	92.18	0.00	0.00	92.18	0.00	0.33
S-m-108(86)x36x3-4x6-05	110.77	16.58	92.40	0.00	0.01	31.09	92.40	0.00	0.00	92.40	0.00	0.25
S-m-108(86)x36x3-4x6-06	109.10	15.57	92.11	0.00	0.00	73.05	92.11	0.00	0.00	92.11	0.00	0.21
S-m-108(86)x36x3-4x6-07	109.15	13.73	94.16	0.00	0.01	247.78	94.16	0.00	0.00	94.16	0.00	0.36
S-m-108(86)x36x3-4x6-08	107.71	13.17	93.53	0.00	0.01	91.03	93.53	0.00	0.00	93.53	0.00	0.31
S-m-108(86)x36x3-4x6-09	103.65	12.02	91.19	0.00	0.01	55.50	91.19	0.00	0.00	91.19	0.00	0.24
S-m-108(86)x36x3-4x6-10	107.41	14.05	92.33	0.00	0.00	56.21	92.33	0.00	0.00	92.33	0.00	0.33
mean	**108.44**	**14.36**	**92.85**	**0.00**	**0.01**	**101.32**	**92.85**	**0.00**	**0.00**	**92.87**	**0.02**	**0.30**
M-m-162(162)x54x3-6x9-01	209.29	10.56	187.19	0.00	0.28	3600.00	187.19	0.00	0.00	187.19	0.00	0.23
M-m-162(162)x54x3-6x9-02	205.08	11.65	181.19	0.00	0.07	3600.00	181.19	0.00	0.00	181.19	0.00	0.23
M-m-162(162)x54x3-6x9-03	211.45	10.87	188.48	0.00	0.53	3600.00	188.48	0.00	0.00	188.48	0.00	0.24
M-m-162(162)x54x3-6x9-04	208.81	11.21	185.40	0.00	0.01	956.99	185.40	0.00	0.00	185.40	0.00	0.33
M-m-162(162)x54x3-6x9-05	212.71	11.45	188.36	0.00	0.07	3600.00	188.36	0.00	0.00	188.36	0.00	0.32
M-m-162(162)x54x3-6x9-06	207.68	11.44	183.93	0.00	0.15	3600.00	183.93	0.00	0.00	183.93	0.00	0.31
M-m-162(162)x54x3-6x9-07	206.83	11.60	182.84	0.00	0.00	20.36	182.84	0.00	0.00	182.84	0.00	0.22
M-m-162(162)x54x3-6x9-08	208.90	10.46	187.04	0.00	0.15	3600.00	187.04	0.00	0.00	187.04	0.00	0.27
M-m-162(162)x54x3-6x9-09	209.70	11.34	185.93	0.00	0.28	3600.00	185.93	0.00	0.00	185.93	0.00	0.18
M-m-162(162)x54x3-6x9-10	199.95	10.60	178.76	0.00	0.01	2669.92	178.76	0.00	0.00	178.76	0.00	0.23
mean	**208.04**	**11.12**	**184.91**	**0.00**	**0.16**	**2884.73**	**184.91**	**0.00**	**0.00**	**184.91**	**0.00**	**0.26**
M-m-162(130)x54x3-6x9-01	156.74	14.60	133.85	0.00	0.19	3600.00	133.85	0.00	0.04	134.07	0.16	0.55
M-m-162(130)x54x3-6x9-02	157.41	12.68	137.45	0.00	0.75	3600.00	137.45	0.00	0.03	137.45	0.00	0.47
M-m-162(130)x54x3-6x9-03	170.71	15.86	143.63	0.00	0.90	3600.00	143.63	0.00	0.05	143.63	0.00	0.53
M-m-162(130)x54x3-6x9-04	169.35	15.68	142.79	0.00	0.29	3600.00	142.79	0.00	0.08	143.04	0.18	0.49
M-m-162(130)x54x3-6x9-05	160.84	15.49	135.93	0.00	0.58	3600.00	135.93	0.00	0.04	136.23	0.22	0.50
M-m-162(130)x54x3-6x9-06	164.65	13.42	142.55	0.00	1.30	3600.00	142.55	0.00	0.03	142.55	0.00	0.66
M-m-162(130)x54x3-6x9-07	165.48	14.73	141.10	0.00	0.78	3600.00	141.10	0.00	0.04	141.81	0.50	0.61
M-m-162(130)x54x3-6x9-08	168.69	15.49	142.56	0.00	0.01	1111.37	142.56	0.00	0.03	143.10	0.38	0.49
M-m-162(130)x54x3-6x9-09	161.81	14.42	138.48	0.00	0.94	3600.00	138.48	0.00	0.04	138.48	0.00	0.51
M-m-162(130)x54x3-6x9-10	162.72	15.98	136.71	0.00	0.48	3600.00	136.71	0.00	0.06	136.76	0.04	0.52
mean	**163.84**	**14.84**	**139.50**	**0.00**	**0.62**	**3351.14**	**139.50**	**0.00**	**0.04**	**139.71**	**0.15**	**0.53**
L-m-189(189)x63x3-7x16-01	224.52	12.47	-	-	-	3600.00	196.53	0.00	0.04	196.53	0.00	0.48
L-m-189(189)x63x3-7x16-02	234.59	9.83	-	-	-	3600.00	211.52	0.00	0.03	211.52	0.00	0.43
L-m-189(189)x63x3-7x16-03	242.75	11.29	-	-	-	3600.00	215.35	0.00	0.05	215.35	0.00	0.52
L-m-189(189)x63x3-7x16-04	237.64	11.18	-	-	-	3600.00	211.07	0.00	0.08	211.07	0.00	0.44
L-m-189(189)x63x3-7x16-05	233.20	12.11	-	-	-	3600.00	204.96	0.00	0.04	204.96	0.00	0.56
L-m-189(189)x63x3-7x16-06	245.72	11.08	218.54	0.02	0.60	3600.00	218.49	0.00	0.03	218.49	0.00	0.49
L-m-189(189)x63x3-7x16-07	241.48	12.61	-	-	-	3600.00	211.04	0.00	0.04	211.04	0.00	0.50
L-m-189(189)x63x3-7x16-08	243.82	12.10	-	-	-	3600.00	214.33	0.00	0.03	214.33	0.00	0.43
L-m-189(189)x63x3-7x16-09	238.22	12.15	-	-	-	3600.00	209.27	0.00	0.04	209.27	0.00	0.34
L-m-189(189)x63x3-7x16-10	242.93	10.44	-	-	-	3600.00	217.57	0.00	0.06	217.57	0.00	0.48
mean	**238.49**	**11.53**	**-**	**-**	**-**	**3600.00**	**211.01**	**0.00**	**0.04**	**211.01**	**0.00**	**0.47**
L-m-189(151)x63x3-7x16-01	186.40	14.01	160.28	0.00	0.11	3600.00	160.28	0.00	27.21	161.86	0.99	1.09
L-m-189(151)x63x3-7x16-02	189.62	15.69	159.87	0.00	0.04	3600.00	159.87	0.00	26.85	160.80	0.59	1.23
L-m-189(151)x63x3-7x16-03	184.75	14.94	157.15	0.00	0.00	355.79	157.15	0.00	34.45	158.64	0.95	1.15
L-m-189(151)x63x3-7x16-04	192.09	15.44	162.44	0.00	0.08	3600.00	162.44	0.00	32.83	164.98	1.56	1.06
L-m-189(151)x63x3-7x16-05	185.27	12.50	162.11	0.00	0.04	3600.00	162.11	0.00	26.99	163.48	0.85	1.04
L-m-189(151)x63x3-7x16-06	183.86	13.92	158.26	0.00	0.00	518.19	158.26	0.00	23.89	159.43	0.74	1.19
L-m-189(151)x63x3-7x16-07	186.10	15.20	157.80	0.00	0.00	319.27	157.80	0.00	24.72	159.11	0.83	1.22
L-m-189(151)x63x3-7x16-08	190.32	14.92	161.93	0.00	0.04	3600.00	161.93	0.00	25.29	163.21	0.79	1.31
L-m-189(151)x63x3-7x16-09	190.18	14.99	161.68	0.00	0.04	3600.00	161.68	0.00	21.02	163.13	0.90	1.06
L-m-189(151)x63x3-7x16-10	186.92	14.70	159.45	0.00	0.00	1062.76	159.45	0.00	25.73	160.41	0.60	1.15
mean	**187.55**	**14.63**	**160.10**	**0.00**	**0.03**	**2385.60**	**160.10**	**0.00**	**26.90**	**161.50**	**0.88**	**1.15**

explanations: primary value = objective value of the status-quo solution; impr. = relative improvement of the optimal objective value compared to the primary value; gap = relative gap to the optimal objective value; time = runtime until termination; C. gap = optimality gap as reported by CPLEX at the point of termination

Table 2.3.: Computational test on realistic-sized instances for a representative male worker.

an increasing amount of stations. While the randomized instances start to show the limits of our solution procedures' capabilities, the results suggest their suitability for even particular difficult real-word cases.

instances			CPLEX				DP			GRASP		
label	primary value	impr. (in %)	value	gap (in %)	C. gap (in %)	time (in s)	value	gap (in %)	time (in s)	value	gap (in %)	time (in s)
XL-m-243(243)x81x3-9x24-01	309.02	13.24	-	-	-	3600.00	268.12	0.00	49.23	268.24	0.05	0.64
XL-m-243(243)x81x3-9x24-02	321.50	12.66	-	-	-	3600.00	280.81	0.00	54.29	281.16	0.12	0.60
XL-m-243(243)x81x3-9x24-03	316.87	12.31	-	-	-	3600.00	277.88	0.00	93.24	278.41	0.19	0.56
XL-m-243(243)x81x3-9x24-04	305.41	11.01	-	-	-	3600.00	271.79	0.00	42.17	271.91	0.05	0.63
XL-m-243(243)x81x3-9x24-05	312.86	11.25	-	-	-	3600.00	277.65	0.00	56.84	277.78	0.05	0.61
XL-m-243(243)x81x3-9x24-06	311.72	12.32	-	-	-	3600.00	273.32	0.00	110.95	273.32	0.00	0.63
XL-m-243(243)x81x3-9x24-07	315.29	11.58	-	-	-	3600.00	278.78	0.00	65.24	278.85	0.02	0.65
XL-m-243(243)x81x3-9x24-08	306.73	10.08	-	-	-	3600.00	275.80	0.00	117.14	276.23	0.15	0.55
XL-m-243(243)x81x3-9x24-09	302.06	11.39	-	-	-	3600.00	267.66	0.00	21.11	267.77	0.04	0.64
XL-m-243(243)x81x3-9x24-10	314.17	11.43	-	-	-	3600.00	278.27	0.00	88.34	278.62	0.13	0.57
mean	**311.56**	**11.73**	**-**	**-**	**-**	**3600.00**	**275.01**	**0.00**	**69.85**	**275.23**	**0.08**	**0.61**
XL-m-243(194)x81x3-9x24-01	245.85	16.07*	206.35	n.k.	0.21	3600.00	-	-	3600.00	207.55	0.58*	2.43
XL-m-243(194)x81x3-9x24-02	245.46	16.70	204.47	0.00	0.00	3089.96	-	-	3600.00	205.57	0.54	2.35
XL-m-243(194)x81x3-9x24-03	246.55	15.95	207.21	0.00	0.00	3492.17	-	-	3600.00	207.90	0.33	1.86
XL-m-243(194)x81x3-9x24-04	247.65	16.33	207.21	0.00	0.00	2912.64	-	-	3600.00	208.67	0.70	2.24
XL-m-243(194)x81x3-9x24-05	245.40	15.74	206.77	0.00	0.00	403.41	-	-	3600.00	209.21	1.18	2.26
XL-m-243(194)x81x3-9x24-06	248.84	17.87	204.37	0.00	0.00	1057.47	-	-	3600.00	206.36	0.97	2.73
XL-m-243(194)x81x3-9x24-07	248.45	16.99	206.24	0.00	0.00	1966.14	-	-	3600.00	208.46	1.07	2.12
XL-m-243(194)x81x3-9x24-08	238.68	15.06	202.74	0.00	0.00	413.07	-	-	3600.00	204.44	0.84	2.03
XL-m-243(194)x81x3-9x24-09	249.78	16.08*	209.62	n.k.	0.03	3600.00	-	-	3600.00	212.31	1.28*	2.49
XL-m-243(194)x81x3-9x24-10	250.95	16.71*	209.02	n.k.	0.06	3600.00	-	-	3600.00	211.00	0.95*	2.39
mean	**246.76**	**16.35**	**206.40**	**n.k**	**0.03**	**2413.49**	-	**-**	**3600.00**	**208.15**	**0.71**	**2.29**
R-162(130)x54x3-6x10-01	225.65	35.61	145.31	0.00	0.01	91.27	145.31	0.00	1.91	146.74	0.99	0.69
R-162(130)x54x3-6x11-02	229.34	37.21	144.01	0.00	0.01	162.80	146.27	0.00	25.97	148.09	2.84	0.64
R-162(130)x54x3-6x12-03	233.88	37.44	146.31	0.00	0.01	121.27	146.31	0.00	36.18	152.18	4.01	0.70
R-162(130)x54x3-6x12-04	219.03	34.16	144.21	0.00	0.01	21.45	144.86	0.00	14.75	150.38	4.28	0.45
R-162(130)x54x3-6x13-05	224.32	35.54	144.60	0.00	0.01	33.56	144.60	0.00	67.80	150.83	4.31	0.46
R-162(130)x54x3-6x14-06	226.30	36.10	144.61	0.00	0.01	41.93	144.61	0.00	125.04	152.23	5.27	0.41
R-162(130)x54x3-6x14-07	230.90	37.59	144.11	0.00	0.01	9.90	144.11	0.00	393.79	150.99	4.77	0.49
R-162(130)x54x3-6x15-08	228.01	37.58	142.32	0.00	0.00	30.51	142.60	0.00	581.64	150.34	5.63	0.42
R-162(130)x54x3-6x15-09	230.48	37.79	143.37	0.00	0.01	33.68	143.37	0.00	1068.08	152.39	6.29	0.41
R-162(130)x54x3-6x16-10	227.50	36.87	143.63	0.00	0.01	14.50	-	-	3600.00	152.63	6.27	0.44
mean	**227.54**	**36.59**	**144.25**	**0.00**	**0.01**	**56.09**	**144.67**	**0.00**	**591.52**	**150.68**	**4.47**	**0.51**

explanations: primary value = objective value of the status-quo solution; impr. = relative improvement of the optimal objective value compared to the primary value; gap = relative gap to the optimal objective value; time = runtime until termination; C. gap = optimality gap as reported by CPLEX at the point of termination; n.k. = not known; * = in comparison to the best found objective value (instead of the known optimum)

Table 2.4.: Computational test on very large and randomized instances.

Finally, we compare the achieved relative improvement in total ergonomic strain for two types of instances, one were the anthropometric measurements of a representative male worker and another one where the anthropometric measurements of a representative female worker were used for the ergonomic evaluation (see Section 2.6). The results are summarized in Table 2.5, which depicts the mean relative improvement (between the primary and optimal objective) for every class of instances for both, the representative male as well as female worker. Depending on the assumed worker, relative improvements only vary by a few tenths of a percentage point (at most by 0.62 percentage points) for each instance class, even though the anthropometrics of both workers differ significantly. For most instance classes, the difference in relative improvements is even below a tenth of a percentage point. The main purpose of this comparison is to demonstrate that relative improvements in total ergonomic strains are roughly independent of the worker's assumed anthropometric measurements. Hence, this comparison suggests that the implications

derived in our computational study – like, for example, the amount of improvement that can be expected by (un-)loading tow trains ergonomically – are valid for a broad spectrum of practical cases with varying workers.

instances	mean rel. impr. male (in %)	mean rel. impr. female (in %)	instances	mean rel. impr. male (in %)	mean rel. impr. female (in %)
S-X-54(54)x18x3-2x3-##	11.54	11.49	XL-X-189(151)x63x3-7x16-##	16.35	16.33
S-X-54(43)x18x3-2x3-##	14.36	14.27	M-X-108(86)x36x3-3x6-##		
M-X-108(108)x36x3-4x6-##	11.12	11.07	(1 extra rack per wagon)	8.30	7.68
M-X-108(86)x36x3-4x6-##	14.84	14.70	M-X-108(86)x27x4-3x6-##		
L-X-162(162)x54x3-6x9-##	11.53	11.47	(1 extra capacity per slot)	1.55	1.60
L-X-162(130)x54x3-6x9-##	14.63	14.58	M-X-108(86)x36x3-3x6-##		
XL-X-189(189)x63x3-7x16-##	11.73	11.68	(1 extra slots per rack)	1.55	1.60

Table 2.5.: Comparison of achieved relative reductions in ergonomic strains for a representative male and female worker.

2.5.2.2. Influence of different tow train designs, set-ups and managerial decisions

This section takes a closer look at how the total ergonomic strain of a tow train is influenced by its design and by managerial decisions. As a first experiment to attain some design insights, we compare reference instances to instances we derive by varying different parameters that define the tow train's set-up. All other parameters, such as bin weights, for example, stay unchanged. The results of our study are summarized in Table 2.6.

instances		results		
label	short description	value	abs. impr.	rel. impr. (in %)
S-m-108(86)x36x3-4x6-##	reference	92.85	0.00	0.00
S-m-108(86)x36x3-3x6-##	1 extra rack per wagon	85.14	7.71	8.30
S-m-108(86)x27x4-3x6-##	1 extra capacity per slot	91.41	1.44	1.55
S-m-108(86)x36x3-3x6-##	1 extra slots per rack	91.41	1.44	1.55

Table 2.6.: Computational test on varying tow train set-ups.

Our reference instances are the S-m-108(86)x36x3-4x6-## instances we also use in our computational performance study. Since in practice, wagons with four shelves are quite common, we first investigate how four-shelved wagons compare to the three-shelved ones of our reference instances. To get comparable results, we set the shelf heights of our four-shelved instances to be in the same realm as the shelf heights of our three-shelved ones, which results in shelf heights of 30 Centimeter, 75 Centimeter, 120 Centimeter and 165 cm. Furthermore, since increasing the number of shelves by one increases the capacity of each wagon by nine, we reduce the number of wagons per tow train by one, so that the tow train has the same overall capacity.

The results in Table 2.6 suggest that it is ergonomically advantageous to use wagons with four shelves, since on average, they cause about 8.3% less ergonomic strain. This is most likely due to the fact that there is more space on shelves of medium height, which enable the worker to pick bins at lower energy expenditure levels. Beyond that, tow trains with four-shelved wagons may offer an additional advantage in practice: since they are generally shorter, they are better suited if space is scarce and paths are narrow.

Our next experiment regards the capacity μ of each slot. By broadening the wagon, and therefore each shelf, it is possible to increase μ. While in our reference instances, we have set $\mu = 3$, a slot capacity of four is also reasonable. We therefore create instances, again based on our reference instances, with an increased slot capacity of $\mu = 4$. This increases each wagons' capacity by nine bins, which is why we again shorten the tow train by one wagon to maintain the same overall capacity. All other parameters remain unchanged.

The results depicted in Table 2.6 show that increasing μ might be beneficial from an ergonomic perspective, but only slightly, since the objective was only improved by about 1.6%. By increasing μ and reducing $|W|$ as described, the amount of space on shelves of a specific height stays constant. This implies that only the altered way of partitioning bins to wagons can influence the objective value. As our results show, this influence is quite low.

Next, we investigate how the total ergonomic strain is influenced by the number of slots on each shelf. Again, starting from our reference instances, we create new instances by adding an additional slot per shelf, i.e., making the wagons longer. As before, we again shorten the tow train by one wagon to retain the same total capacity.

The results of this experiment are shown in Table 2.6. They mirror the results of the previous experiment, which is not coincidental: Increasing the slot depth and lengthening the wagons both add the same number of additional slots to each shelf level. Therefore, both kinds of altered instances yield the same change of the optimal objective value, since we assume (un-)loading a bin from/onto slots at the same shelf height results in equal ergonomic strains.

Finally, we investigate how the relative filling of the tow train influences total ergonomic strain. Clearly, loading fewer bins on a tow train reduces the total ergonomic strain for (un-)loading. However, assuming line-side demands stay unchanged, tow trains that are loaded to a lesser degree have to execute delivery cycles more frequently to fulfill demands. Similar trade-offs

have already been reported in the literature with the objective of minimizing line-side stocks (see Section 2.2), where more frequent smaller lot deliveries prove to be advantageous (Emde et al., 2012).

To assess the effects of this tradeoff on ergonomics, we conduct the following experiment. We begin with an instance of a completely filled trow train and solve it to optimality. The result is used as a reference in the subsequent evaluation. In the next step we reduce the tow train's filling by a relative amount Φ by deleting $(1-\Phi)\cdot n$ bins (i.e., we transform them into "dummy" bins) from the respective instance, while we keep the bin distribution regarding destined stations and weights as similar as possible. We solve the modified instance to optimality and calculate an adjustment factor $\frac{1}{\Phi}$, which denotes how much more frequent the tow train of the modified instance needs to execute delivery runs compared to the reference instance's tow train. We then multiply the optimal objective value of the modified instance by the adjustment factor and divide the product by the objective value of the reference instance. This yields an index that compares the ergonomic strain of the modified instance and the reference instance. We acknowledge that this experiment is based on quite restrictive assumptions, for example, that the number of tow trains, their capacity and their assigned drivers are fixed. We further do not account for the fact that decreasing the filling of a tow train by a certain amount reduces the time to complete a delivery cycle only by a sub-proportional time span, which leads to infeasible schedules for low fillings. Nevertheless, our experiment can provide some valuable insight into how just-in-time lean production strategies, which emphasize more frequent small lot deliveries, influence ergonomics.

We evaluated all small-, medium- and large-sized instances with the described procedure for different values of $\Phi = \{\frac{1}{9}, \frac{2}{9}..., \frac{9}{9}\}$. The averaged results are depicted in Figure 2.5.

Independent of the instance size, Figure 2.5 shows some clear trends. Between $\Phi = \frac{1}{9}$ and $\Phi = \frac{3}{9}$, the index of the total ergonomic strain stays approximately constant. Afterwards, between $\Phi = \frac{3}{9}$ and $\Phi = 1$, the index of the adjusted total ergonomic strain increases approximately linearly with increasing tow train filling. We explain this trend as follows. If the tow train is filled to a high degree, all ergonomically advantageous places are occupied by bins, such that the remaining bins are forced to be stored on ergonomically disadvantageous rack heights. Reducing the filling of the tow train decreases the amount of bins that are forced to be stored on the latter. Hence, even though the tow train needs to deliver more often, the total amount of bins, which are transported (and, hence, (un-)loaded) on ergonomically disadvantageous racks, decreases. Once the tow train is filled to a degree of $\Phi = \frac{3}{9}$ or less, storage space is sufficient to store all bins on the ideal rack height (since there are only three different rack heights at

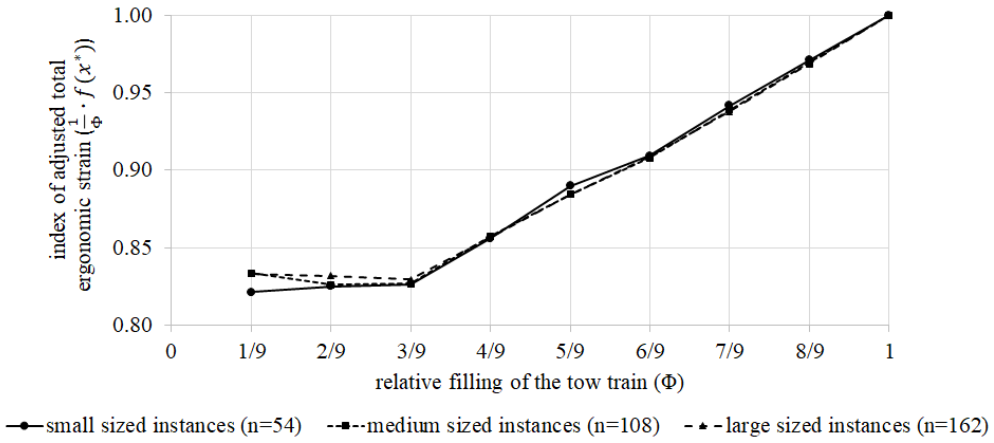


Figure 2.5.: Influence of tow train filling on total ergonomic strain.

our instances' tow trains), which explains why the decreasing trend becomes approximately constant. Deviations from this trend, which are more pronounced the less the tow train is filled, are due to our experimental design: the less the tow train is filled, the less accurate we are able to mirror the primary distribution, which causes some disturbances. In conclusion, the results suggest that lean production strategies – which are often implemented where tow trains are deployed – can have positive effects on the reduction of ergonomic strains.

2.6. Conclusion

In this paper we consider the novel problem of improving the stowage of bins on tow trains from an ergonomic point of view without worsening its economical performance. We formalize this problem as a MIP model and prove that finding feasible solutions is NP-hard.

We decompose the problem into two sub-problems: first a partitioning problem, and, second, an assignment problem. While the assignment problem can be efficiently solved, the partitioning step is a bottleneck. We develop two solution procedures, an exact dynamic programming algorithm as well as a heuristic solution procedure embedded in a GRASP framework, both using the problem's decomposability. Computational tests show that our dynamic programming procedure is able to solve most instances of realistic size within a runtime of below a tenth of a second. The GRASP-based heuristic procedure performs well, especially on very large instances. Optimality gaps are below 1% in nearly all cases and the runtime never exceeds 3 seconds.

We further derive the following take-home conclusions from a managerial point of view.

- Total ergonomic strain can be lessened, in our examples by up to about 14% to 15%, by simply optimizing bin stowage.
- From a design perspective, tow train wagons with more racks seem to be advantageous, as they offer more storage capacity at easy-to-access heights. By comparing two kinds of tow trains, one with three racks per wagons and one with four racks, both having the same overall loading capacity, the latter allowed for optimal stowage plans causing on average 8.3% less total ergonomic strain compared to the former.
- As a further design issue, we found that increasing each wagon's capacity by broadening or lengthening it allows for the reduction of the total amount of wagons per tow train without worsening ergonomics. In fact, in our tests, optimal total ergonomic strains even slightly decreased by about 1.6% on average.
- Decreasing a tow train's relative filling while increasing its delivery frequency reduces total ergonomic strain. Hence, high-frequency small-lot delivery strategies, which are emphasized in lean production concepts, are also advantageous from an ergonomic perspective.

Future research may aim to refine or validate the ergonomic evaluation method we use to assess (un-)loading tow trains. This can either be done by using alternative ergonomic assessment methods – a biomechanical approach, for example – or via actual field measurements. In addition, research might try to quantify the health benefits of stowing bins on a tow train in an ergonomic manner, possibly by means of a case study.

For additional managerial insights, assessing the weight distribution of bins may be of interest to conclude if it is beneficial to fill bins in a way that ensures equal weights, or if an unequal distribution offers ergonomic advantages.

Regarding the mathematical model, alternative objectives may be considered. One possibility is to minimize the maximum sum of ergonomic strains per station if the tow train is unloaded by the worker at the respective stations. Another option is to minimize the maximum ergonomic strain, which aims on preventing overload injuries from (un-)loading especially heavy bins. Further, it is possible to investigate additional conditions, like, for example, a given inbound sequence of bins that has to be taken account of.

2.A. Appendix

2.A.1. Additional figure

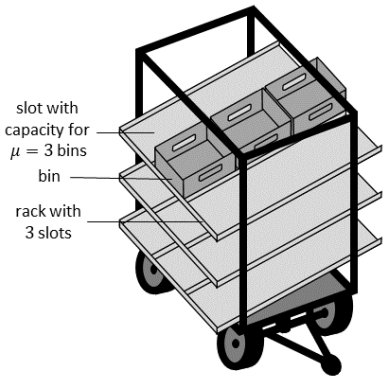


Figure 2.6.: Schematic representation of a typical tow train wagon's design.

2.A.2. Integer program of BTPAP

Using the notation in Table 2.7, the BTPAP can be formulated as the following integer program.

$B_{w'}$	set of bins assigned to wagon w' (index j)
$\tilde{P}_{w'}$	set of positions on wagon w' (index t)
$y_{t,j}$	binary variable: 1, if bin j is assigned to slot-position t; 0, otherwise

Table 2.7.: Additional notation.

$$\text{[BTPAP] Minimize } g(\boldsymbol{y}) = \sum_{t=(p,k)\in\tilde{P}_{w'}} \sum_{j\in B_{w'}} e(j,p)\cdot y_{t,j} \tag{2.13}$$

subject to

$$\sum_{t\in\tilde{P}_{w'}} y_{t,j} = 1 \qquad \forall j \in B_{w'} \tag{2.14}$$

$$\sum_{j\in B_{w'}} y_{t,j} = 1 \qquad \forall t \in \tilde{P}_{w'} \tag{2.15}$$

$$y_{t,j} \in \{0,1\} \qquad \forall t \in \tilde{P}_{w'}; j \in B_{w'} \quad (2.16)$$

Objective function (2.13) minimizes the total ergonomic strain of the assignment. Constraints (2.14) and (2.15) ensure that each bin is assigned to exactly one position and vice versa. Finally, Constraints (2.16) restrict the domain of the variables to $\{0,1\}$.

2.A.3. Measuring ergonomic strain

The manual unloading of racks – and, similarly, tow train wagons as in the case of this paper –, consists of three main activities that may expose the worker to a significant physical load: the pulling, lifting and lowering of objects. The manual handling of materials and its impact on the human body has traditionally been studied in the human factors engineering literature, see, for example, the basic textbooks of Helander (2005) and Winter (2009). To assess MMH activities' risks of increasing workers' likelihood to develop MSD, the literature differentiates between four major approaches (Moore and Garg, 1995):

- the observation and subjective evaluation of MMH activities by highly qualified professionals,
- the examination of biomechanical, physiological and/or psychophysical critical threshold responses,
- the analysis of epidemiological data on the correlation of jobs, activities or other variables with increased risk of MSD,
- and combinations of the methods mentioned above.

Although professional judgment of MMH activities by specialists is a valuable method in practice, this approach, due to its subjectiveness, is less suitable to determine quantitative ergonomic strains (Moore and Garg, 1995). The evaluation of epidemiological data, on the other hand, is a much more objective approach. However, epidemiological studies typically do not assess MMH via exact, quantitative values. Rather, they are designed to determine or verify the link between certain variables, activities or tasks to certain MSD (Dempsey, 1998). Therefore, and due to the fact, that epidemiological studies are very costly in terms of time, this approach is also less suited for our purposes.

The determination of different critical threshold responses is based on the observation that ex-

ceeding certain thresholds fosters the development of various MSD; the higher the limit is exceeded, the higher the likelihood of developing MSD. The emphasis of the biomechanical approach is the force or torque applied to certain joints or muscles during various MMH tasks, most commonly the lower back joints L_4/L_5 or L_5/S_1. The physiological approach focuses on a worker's energy expenditure during a MMH task, i.e., the cardiovascular effort associated with performing the task. Lastly, psychophysical approaches consider a worker's perceived strain for evaluating the task (Dempsey, 1998).

While it is possible to assess various threshold responses of workers performing MMH activities directly via measurements (e.g., via body-mounted sensors), this approach may interfere with and hence disturb the very activity it tries to evaluate. Therefore, to ease evaluation and to avoid intervening with task performance, models have been developed to calculate respective responses for a variety of MMH activities (Dempsey, 1998).

A physiological model that has enjoyed some popularity in the past both in research and in practice is the energy expenditure prediction model developed by Garg et al. (1978). It is based on the assumption that the energy expenditure rate of a composite task, such as (un-)loading a shelf, for example, can be calculated by summing up its comprising basic tasks' energy expenditure rates. Energy expenditure rates of basic tasks depend on various task characteristics such as distances, heights, weights, forces and body positions, as well as anthropological characteristics of the person performing the task, like sex and body weight, and they can be calculated using equations that were derived by the authors via regression analyses of data obtained in laboratory experiments. The higher a composite task's calculated energy expenditure rate, the greater its ergonomic strain. Prior research assumed that the maximum average energy expenditure rate an average person can tolerate over the course of an eight-hour workday is $5\frac{\text{kcal}}{\text{min}}$, though this value is debated (Dempsey, 1998).

One of the advantages of the model of Garg et al. (1978) is its suitability for the evaluation of a multitude of MMH activities, making it very flexible in application (Dempsey, 1998). In addition, it is easier to handle than many biomechanical models. It offers the possibility to account for worker-individual data and has previously been used in various application areas (e.g., Garg et al., 1978, Waters et al., 1998, Glock et al., 2018) as well as in recent managerial approaches taking account of ergonomic aspects (see. Section 2.2). Due to this flexibility and widespread use, we use the energy expenditure prediction model of Garg et al. (1978) to calculate the ergonomic strain for our instances.

The first step is to analyze the task of (un-)loading a tow train wagon, which in our case, is equipped with a gravity flow shelf, and to break this task down into basic tasks. To do this,

we observed different workers (un-)loading shelves of different heights. We subdivided their movements into a set of basic tasks and derived the following generalized sequence for loading:

1. At the beginning of the loading process, the worker adopts an upright standing posture about 50 Centimeter in front of the shelf, facing it. His[7] legs are positioned slightly shifted. The worker grasps the bin with both hands at the centers of its sides, holding it at hip height, so that it is in contact with his hip and thighs.

2. The worker moves the bin right to the front of the shelf on which he wants to place it. This includes a movement of the hands, arms and shoulders in horizontal and in vertical direction. If the destined shelf is located below hip height, the worker has to squat in addition.

3. The worker places the bin's adverted bottom side on the rear edge of the shelf.

4. The worker moves his hands, one after the other, to the front of the bin to change grip.

5. The worker pushes the bin about halfway over the edge of the shelf.

6. The worker releases the bin, so that it is pulled onto the rack by gravity.

7. The worker returns to an upright standing posture, while aligning his hands, arms and shoulders with his torso, so that his hands come in contact with his outer thighs.

8. The loading movement ends.

For unloading, we observed the following sequence:

1. At the beginning of the unloading process, the worker adopts an upright standing posture about 50 Centimeter in front of the shelf, facing it. His legs are positioned slightly shifted. The worker's arms, hands and shoulders are hanging sideways, aligned with his torso. His hands are in contact with his outer thighs just below hip height.

2. The worker moves his hands to the front of the bin that he wants to unload. This includes a movement of the hands, arms and shoulders in horizontal and vertical direction. If the destined shelf is located below hip height, the worker has to squat in addition.

3. The worker lifts the bin's front up slightly, so that its lower edge slips over the frontal edge of the shelf.

[7] We use the male gender to refer to individuals of arbitrary gender.

4. The worker pulls the bin about halfway over the edge of the shelf.

5. The worker moves his hands, one after the other, to the middle of the sides of the bin to change grip.

6. The worker pulls the bin off the shelf's frontal edge to its full length.

7. The worker brings the bin vertically to hip height and horizontally to contact with his hip and thighs. Depending on the shelf height from which he gathered the bin, the vertical movement consists of lowering or lifting. If the bin was located below hip height, the latter also includes bringing the body to an upright posture again.

8. The unloading movement ends.

For additional clarification, both the loading and the unloading process are schematically depicted in Figures 2.7 and 2.8.

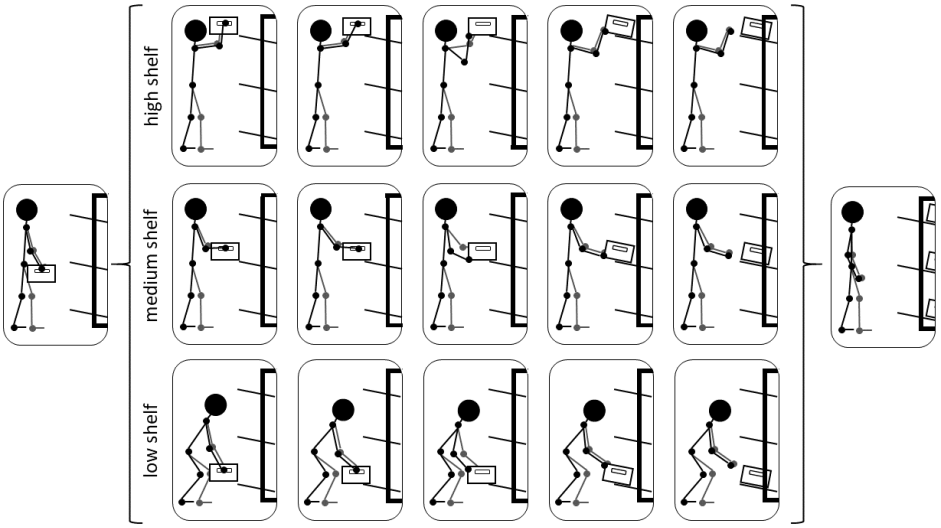


Figure 2.7.: Schematic sequence of tasks to load a tow train wagon.

In addition to the basic tasks described above, the worker has to carry bins to the tow train during loading and carry bins to their destination during unloading. However, these activities are not part of the basic (un-)loading movement and cannot be influenced by optimizing the bins' stowage on the tow train. Therefore, we do not include them in our ergonomic evaluation. Note that the general sequence of steps as listed above is independent of shelf height and bin weight. Only in some steps, it is necessary to differentiate between the precise movements (for example lifting versus lowering a bin in step 7 during unloading) that depend on the shelf height. Further,

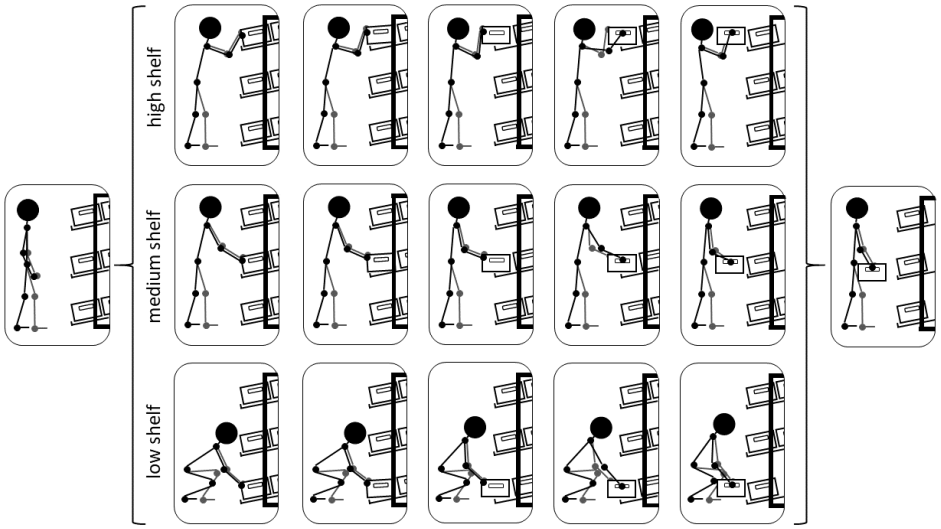


Figure 2.8.: Schematic sequence of tasks to unload a tow train wagon.

we assume the movement to (un-)load bins from slots at the same shelf height is identical, which is reasonable in a lot of practical situations. We note, however, that this may not always be true. Bars of the rack's frame (as depicted in Figure 2.6) may, for example, hinder gripping bins at the outer slots. Another possibility may be that the worker deposits/receives bins at a small depot centrally in front of the wagon, such that to move from the wagon to the depot, he has to turn his body by different angles, depending on the horizontal slot position where a bin is stored. In both examples, the sequences of movements and, hence, the provoked ergonomic strains may vary between slots at the same shelf height.

Applying the equations of Garg et al. (1978) to the sequence of movements described above enables us to calculate the average ergonomic strain associated with loading and unloading an arbitrary weighted bin to/from an arbitrary shelf height. Hence, we calculate the ergonomic strain $e(j, p)$ arising from placing bin j in slot p as a function of the bin's weight and the shelf's height slot p is located at.

For the sake of generality, in the proceedings of the computational study, we primarily use the anthropometric measurements of a representative worker, derived from the average worker's anthropometrics we observed in practice. Our representative worker is male, weighs 75 Kilogram and measures 178 Centimeter in height, with a hip height of 80 Centimeter (Glock et al., 2019). A similar approach of defining a representative worker is taken by Battini et al. (2016b) and Glock et al. (2019), for example. We acknowledge that individual workers (with different anthropometric measurements) can experience different ergonomic strains and may be able to tolerate

varying thresholds without an increased risk of developing MSD. However, in the course of our computational study, we aim to provide a general estimate of the reduction in ergonomic strains that can be achieved by loading tow trains ergonomically. This reduction, expressed in relative terms, is approximately equal independent of the worker's anthropometric measurements. We demonstrate the latter by additionally evaluating and optimizing the ergonomic strains for an average female worker weighting 61.9 Kilogram and measuring 161.4 Centimeter in height (Tehard et al., 2002), which results in almost identical relative reductions (see Section 2.5.2). We further note that our model is not restricted to those anthropometric measurements; in practical applications different anthropometrics could be used.

Despite the mentioned advantages of the energy expenditure prediction model of Garg et al. (1978), this approach also has shortcomings. First of all, the assumption that a composite task's energy expenditure rate is the sum of its basic tasks' energy expenditure rates turned out to be questionable in various studies (Genaidy et al., 1985, Taboun and Dutta, 1989). This may lead to a systematic over-prediction of energy expenditure rates (Ayoub, 1992). However, since we compare very similar tasks to each other, i.e., (un-)loading from different rack heights, this systematic bias should be roughly the same for every task, so that comparisons of differences in the tasks' energy expenditures should be still adequately accurate.

A second problem is that while we can compare the total energy expenditures resulting from different bin stowages on the tow train, this does not tell us exactly how the risk of developing MSD changes for the worker (un-)loading it (Ayoub, 1992, Dempsey, 1998). This is due to multiple reasons. First, an activity harmful for one worker may not negatively influence the health of another worker (Snook and Ciriello, 1991). Secondly, we do not know which activities the worker performs between loading and unloading the tow train. Reducing the ergonomic strain of (un-)loading tow trains is more beneficial, the higher the worker's energy expenditure during further activities. Thirdly, even if we knew a worker's exact energy expenditure rates over the course of his complete shift, there is a lack of statistics on how the risk of developing MSD is precisely linked to certain levers of energy expenditure rates (Ayoub, 1992, Dempsey, 1998).

The limitations mentioned above are not unique to the energy expenditure concept, but apply to ergonomic assessment methods in general (Ayoub, 1992, Dempsey, 1998); hence, selecting a different evaluation method would not solve this problem. We note, however, that we formulate the ETTLP in a way that ensures that improving the ergonomics assessment of the tow train does not worsen economic performance. The reduction in energy expenditure induced by our method improves the workplace quality for the tow train driver, even if we are unable to calculate an

exact injury risk. Hence, the explanatory power of the energy expenditure concept is sufficient for the scope of this paper.

A last minor problem concerns the physical dimension of energy expenditure rates. Energy expenditure rates have the physical dimension of power (energy per unit of time). Summing up values in the dimension of power, as we do in our objective function, to find the optimal bin stowage results in an outcome value not representing any real-world physical property, which is why we use absolute energy expenditures instead. The equations of Garg et al. (1978) actually calculate a complex task's energy expenditure rate by accumulating the comprising tasks' absolute energy expenditures, before dividing them through the task's duration. The only exception to this is the energy expenditure to keep the body in an upright standing position, which is directly dependent on the task's duration. According to our observations in practice, there was no clear correlation between bin weights or shelf heights and the (un-)loading task's duration, which, in general, was somewhat fluctuating. Hence, to calculate the energy expenditure to keep the body in a standing position, we assume that every basic (un-)loading task's duration is three seconds, which is the mean duration we observed. Due to the minor contribution of the energy expenditure to keep the body in a standing position to the task's total energy expenditure, we regard this assumption to be acceptable. Using absolute energy expenditures allows us to interpret the objective value as the total energy needed to (un-)load the tow train. We can then re-transform the objective into a value in energy expenditure rates by dividing it through the total duration of (un-)loading the tow train, which is n multiplied by three seconds.

2.B. Bibliography

Aboudi, R., Hallefjord, Å., and Jörnsten, K. (1991). A facet generation and relaxation technique applied to an assignment problem with side constraints. *European Journal of Operational Research*, 50(3):335–344.

Aboudi, R. and Nemhauser, G. L. (1991). Some facets for an assignment problem with side constraints. *Operations Research*, 39(2):244–250.

Aiyar, S., Ebeke, C., and Shao, X. (2017). The impact of workforce aging on european productivity. Technical Report WP/16/238, International Monetary Fund.

Alnahhal, M. and Noche, B. (2015). A genetic algorithm for supermarket location problem. *Assembly Automation*, 35(1):122–127.

Alnahhal, M., Ridwan, A., and Noche, B. (2014). In-plant milk run decision problems. *Facilities*, 35(22):25–45.

Ayoub, M. (1992). Problems and solutions in manual materials handling: the state of the art. *Ergonomics*, 35(7-8):713–728.

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.

Battini, D., Delorme, X., Dolgui, A., Persona, A., and Sgarbossa, F. (2016a). Ergonomics in assembly line balancing based on energy expenditure: a multi-objective model. *International Journal of Production Research*, 54(3):824–845.

Battini, D., Faccio, M., Persona, A., and Sgarbossa, F. (2010). "supermarket warehouses": stocking policies optimization in an assembly-to-order environment. *The International Journal of Advanced Manufacturing Technology*, 50(5-8):775–788.

Battini, D., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2016b). Human energy expenditure in order picking storage assignment: A bi-objective method. *Computers & Industrial Engineering*, 94:147–157.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60:503–515.

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.

Braam, I. T. J., van Dormolen, M., and Frings-Dresen, M. H. (1996). The work load of warehouse workers in three different working systems. *International Journal of Industrial Ergonomics*, 17(6):469–480.

Burkard, R., Dell'Amico, M., and Martello, S. (2012). *Assignment problems*, volume 106. SIAM, Philadelphia, revised edition.

Calzavara, M., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2017). Analysis of economic and ergonomic performance measures of different rack layouts in an order picking warehouse. *Computers & Industrial Engineering*, 111:527–536.

Caramia, M. and Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces*, 40(2):130–143.

Cattrysse, D. G. and Van Wassenhove, L. N. (1992). A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272.

Choi, W. and Lee, Y. (2002). A dynamic part-feeding system for an automotive assembly line. *Computers & Industrial Engineering*, 43(1-2):123–134.

Coffman, E. G., Garey, M. R., and Johnson, D. S. (1996). Approximation algorithms for bin packing: A survey. *Approximation algorithms for NP-hard problems*, pages 46–93.

Dempsey, P. G. (1998). A critical review of biomechanical, epidemiological, physiological and psychophysical criteria for designing manual materials handling tasks. *Ergonomics*, 41(1):73–88.

Drury, C. G., Law, C.-H., and Pawenski, C. S. (1982). A survey of industrial box handling. *Human Factors*, 24(5):553–565.

Emde, S. (2017). Scheduling the replenishment of just-in-time supermarkets in assembly plants. *OR Spectrum*, 39(1):321–345.

Emde, S. and Boysen, N. (2012a). Optimally locating in-house logistics areas to facilitate JIT-supply of mixed-model assembly lines. *International Journal of Production Economics*, 135(1):393–402.

Emde, S. and Boysen, N. (2012b). Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines. *European Journal of Operational Research*, 217(2):287–299.

Emde, S., Fliedner, M., and Boysen, N. (2012). Optimally loading tow trains for just-in-time supply of mixed-model assembly lines. *IIE Transactions*, 44(2):121–135.

Emde, S. and Gendreau, M. (2017). Scheduling in-house transport vehicles to feed parts to automotive assembly lines. *European Journal of Operational Research*, 260(1):255–267.

Emde, S. and Schneider, M. (2018). Just-in-time vehicle routing for in-house part feeding to assembly lines. *Transportation Science*, 52(3):497–737.

European Commission (2017). The 2018 ageing report: Underlying assumptions and projection methodologies. Technical Report Institutional Paper 065, European Commission, Brussels.

Fathi, M., Rodríguez, V., Fontes, D. B., and Alvarez, M. J. (2016). A modified particle swarm optimisation algorithm to solve the part feeding problem at assembly lines. *International Journal of Production Research*, 54(3):878–893.

Felici, G. and Mecoli, M. (2007). Resource assignment with preference conditions. *European Journal of Operational Research*, 180(2):519–531.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Fischetti, M. and Lodi, A. (2008). Repairing MIP infeasibility through local branching. *Computers & Operations Research*, 35(5):1436–1445.

Friesen, D. K. and Langston, M. A. (1986). Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222–230.

Gardner, L. I., Landsittel, D. P., and Nelson, N. A. (1999). Risk factors for back injury in 31,076 retail merchandise store workers. *American Journal of Epidemiology*, 150(8):825–833.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: a guide to NP-completeness*. WH Freeman and Company, San Francisco.

Garg, A. (2000). Ergonomic, biomechanical and physiological stresses from manual materials handling in grocery distribution centers. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 44, pages 431–432. SAGE Publications Sage CA: Los Angeles, CA.

Garg, A., Chaffin, D. B., and Herrin, G. D. (1978). Prediction of metabolic rates for manual materials handling jobs. *American Industrial Hygiene Association Journal*, 39(8):661–674.

Genaidy, A., Asfour, S., Khalil, T., and Waly, S. (1985). Physiologic issues in manual materials handling. *Trends in Ergonomics/Human Factors II*, 1(1):571–576.

Glock, C. H., Grosse, E. H., Abedinnia, H., and Emde, S. (2019). An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. *European Journal of Operational Research*, 273(2):516–534.

Glock, C. H., Grosse, E. H., Kim, T., Neumann, W. P., and Sobhani, A. (2018). An integrated cost and worker fatigue evaluation model of a packaging process. *International Journal of Production Economics*.

Golz, J., Gujjula, R., Günther, H.-O., Rinderer, S., and Ziegler, M. (2012). Part feeding at high-variant mixed-model assembly lines. *Flexible Services and Manufacturing Journal*, 24(2):119–141.

Grosse, E. H., Glock, C. H., Jaber, M. Y., and Neumann, W. P. (2015). Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, 53(3):695–717.

Grosse, E. H., Glock, C. H., and Neumann, W. P. (2017). Human factors in order picking: a content analysis of the literature. *International Journal of Production Research*, 55(5):1260–1276.

Helander, M. (2005). *A guide to human factors and ergonomics*. CRC Press, Boca Raton, second edition.

Korf, R. E. (2009). Multi-way number partitioning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 538–543, Palo Alto. IJCAI Organization, AAAI Press.

Larco, J. A., de Koster, R., Roodbergen, K. J., and Dul, J. (2017). Managing warehouse efficiency and worker discomfort through enhanced storage assignment decisions. *International Journal of Production Research*, 55(21):6407–6422.

Larsson, B., Søgaard, K., and Rosendal, L. (2007). Work related neck–shoulder pain: a review on magnitude, risk factors, biochemical characteristics, clinical picture and preventive interventions. *Best Practice & Research Clinical Rheumatology*, 21(3):447–463.

Lavender, S. A., Marras, W. S., Ferguson, S. A., Splittstoesser, R. E., and Yang, G. (2012). Developing physical exposure-based back injury risk models applicable to manual handling jobs in distribution centers. *Journal of Occupational and Environmental Hygiene*, 9(7):450–459.

Lieb, C., Klenk, E., Galka, S., and Keuntje, C. (2017). Einsatz von Routenzugsystemen zur Produktionsversorgung – Studie zu Planung, Steuerung und Betrieb. Technical report, Technische Universität München, Garching.

Marras, W. S., Lavender, S. A., Ferguson, S. A., Splittstoesser, R. E., and Yang, G. (2010). Quantitative biomechanical workplace exposure measures: distribution centers. *Journal of Electromyography and Kinesiology*, 20(5):813–822.

Medbo, L. (2003). Assembly work execution and materials kit functionality in parallel flow assembly systems. *International Journal of Industrial Ergonomics*, 31(4):263–281.

Moore, J. S. and Garg, A. (1995). The strain index: a proposed method to analyze jobs for risk of distal upper extremity disorders. *American Industrial Hygiene Association Journal*, 56(5):443–458.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

Neumann, W. and Medbo, L. (2010). Ergonomic and technical aspects in the redesign of material supply systems: Big boxes vs. narrow bins. *International Journal of Industrial Ergonomics*, 40(5):541–548.

Neumann, W. P., Wells, R., and Norman, R. (1999). 4D WATBAK: adapting research tools and epidemiological findings to software for easy application by industrial personnel. In *Proceedings of the International Conference on Computer-Aided Ergonomics and Safety*, Barcelona, Spain.

Occhipinti, E. (1998). OCRA: a concise index for the assessment of exposure to repetitive movements of the upper limbs. *Ergonomics*, 41(9):1290–1311.

Otto, A., Boysen, N., Scholl, A., and Walter, R. (2017). Ergonomic workplace design in the fast pick area. *OR Spectrum*, 39(4):945–975.

Otto, A. and Scholl, A. (2011). Incorporating ergonomic risks into assembly line balancing. *European Journal of Operational Research*, 212(2):277–286.

Palmerud, G., Forsman, M., Neumann, W. P., and Winkel, J. (2012). Mechanical exposure implications of rationalization: a comparison of two flow strategies in a swedish manufacturing plant. *Applied Ergonomics*, 43(6):1110–1121.

Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793.

Petersen, C. G., Siu, C., and Heiser, D. R. (2005). Improving order picking performance utilizing slotting and golden zone storage. *International Journal of Operations & Production Management*, 25(10):997–1012.

Punnett, L. and Wegman, D. H. (2004). Work-related musculoskeletal disorders: the epidemiologic evidence and the debate. *Journal of Electromyography and Kinesiology*, 14(1):13–23.

Roquelaure, Y., Ha, C., Rouillon, C., Fouquet, N., Leclerc, A., Descatha, A., Touranchet, A., Goldberg, M., Imbernon, E., and of Occupational Health Services of the Pays de la Loire Region, M. (2009). Risk factors for upper-extremity musculoskeletal disorders in the working population. *Arthritis Care & Research*, 61(10):1425–1434.

Roy, B. and Słowiński, R. (2006). Multi-criteria assignment problem with incompatibility and capacity constraints. *Annals of Operations Research*, 147(1):287–316.

Schaub, K., Caragnano, G., Britzke, B., and Bruder, R. (2013). The european assembly worksheet. *Theoretical Issues in Ergonomics Science*, 14(6):616–639.

Schneider, E. and Irastorza, X. (2010). Work-related musculoskeletal disorders in the EU – facts and figures. Technical report, European Agency for Safety and Health at Work, Bilbao, Spain.

Snook, S. H. and Ciriello, V. M. (1991). The design of manual handling tasks: revised tables of maximum acceptable weights and forces. *Ergonomics*, 34(9):1197–1213.

Taboun, S. and Dutta, S. (1989). Energy cost models for combined lifting and carrying tasks. *International Journal of Industrial Ergonomics*, 4(1):1–17.

Tehard, B., Van Liere, M. J., Nougué, C. C., and Clavel-Chapelon, F. (2002). Anthropometric measurements and body silhouette of women: validity and perception. *Journal of the American Dietetic Association*, 102(12):1779–1784.

Vaidyanathan, B. S., Matson, J. O., Miller, D. M., and Matson, J. E. (1999). A capacitated vehicle routing problem for just-in-time delivery. *IIE Transactions*, 31(11):1083–1092.

VDA (2018). *VDA 4500 Kleinladungsträger (KLT-) System*. Verband der Automobilindustrie, Berlin.

Waters, T. R., Putz-Anderson, V., and Baron, S. (1998). Methods for assessing the physical demands of manual lifting: a review and case study from warehousing. *American Industrial Hygiene Association Journal*, 59(12):871–881.

Waters, T. R., Putz-Anderson, V., Garg, A., and Fine, L. J. (1993). Revised niosh equation for the design and evaluation of manual lifting tasks. *Ergonomics*, 36(7):749–776.

Winter, D. A. (2009). *Biomechanics and motor control of human movement*. John Wiley & Sons, Hoboken, NJ.

Paper 3: Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines[8]

Abstract: This paper 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.

Keywords: Scheduling; Unrelated parallel machines; Time windows; Benders decomposition; Berth allocation; Truck scheduling

[8]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. DOI: `https://doi.org/10.1007/s00291-020-00586-w`. Reproduced with permission from Springer Nature.

3.1. 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 3.2.

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

$$C(S) = \max_{(i,j,t)\in S}\{t + p_{ij}\}. \tag{3.1}$$

- *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 3.2, we review the pertinent literature and discuss real-world applications of our scheduling problem in more detail. In Section 3.3, 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 3.4. In Section 3.5 performance of our algorithms is tested on benchmark instances in a computational study. Finally, Section 3.6 concludes the paper.

3.2. 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 3.4, 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.

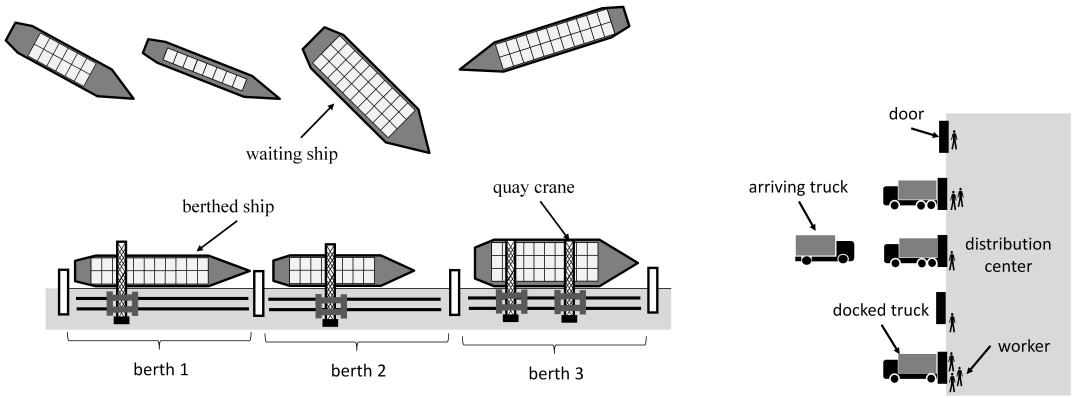
In summary, most papers employing logic-based Benders decomposition for parallel machine scheduling use a constraint programming solver for the subproblems, the exception being Tran et al. (2016), who use a TSP solver. In this paper, we develop a novel bounded dynamic programming-based approach, which is shown to perform quite well and can easily be adapted to a broad range of machine scheduling problems (see Section 3.4).

3.2.1. 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 3.1(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.



(a) Berth allocation.

(b) Truck scheduling.

Figure 3.1.: Applications of $[R|r_j, \overline{d}_j|C_{\max}]$.

3.2.2. 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. 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 pro-

cessed 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 3.1(b).

3.3. 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 3.4.1) and alternative minmax objectives (Sections 3.4.2.1, 3.4.2.2). The exact algorithm, based on logic-based branch and Benders cut, is presented in Section 3.3.1. 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 3.3.2.

3.3.1. 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.

3.3.1.1. 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.

$$\text{[Master] Minimize } F^M(Y, Z) = z \tag{3.2}$$

subject to

$$\sum_{i \in P} y_{ij} = 1 \qquad \forall j \in J \quad (3.3)$$

$$r_j + \sum_{j' \in N(j)} y_{ij'} \cdot p_{ij'} \leq z \qquad \forall i \in P; j \in J \quad (3.4)$$

$$r_j + \sum_{j' \in N(j)} y_{ij'} \cdot p_{ij'} \leq \overline{d}_{\max}(j) \qquad \forall i \in P; j \in J \quad (3.5)$$

$$y_{ij} \in \{0; 1\} \qquad \forall i \in P, j \in J \quad (3.6)$$

Objective (3.2) minimizes the makespan of the relaxed problem. Constraints (3.3) make sure that each job is assigned to exactly one machine. Valid inequalities (3.4) 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 (3.5) 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 (3.6) 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 (3.4) and (3.5). 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.

3.3.1.2. 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 3.2, 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., $\mathcal{C}(\Sigma) + \sum_{j \in \Gamma_i \setminus \Sigma} p_{ij} \leq \max_{j \in \Gamma_i \setminus \Sigma}\{\overline{d}_j\}$ 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.

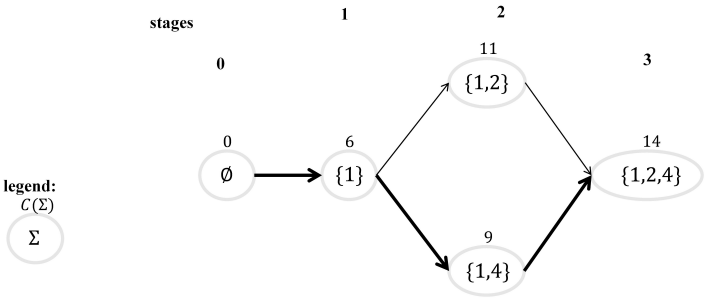
Example: Consider the example data given in Figure 3.2(a), consisting of $n = 4$ jobs and $m = 2$ processors. Assume that for some [Master] solution $\Gamma_2 = \{1, 2, 4\}$, i.e., jobs 1, 2, and 4 are assigned to processor 2. The resulting dynamic programming graph is in Figure 3.2(b). One of the two optimal solutions is bold, corresponding to job sequence $\langle 1, 4, 2]$.

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]:

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

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 3.2.: Example data and dynamic programming.

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 $I = \{i \in P \mid \mathcal{C}(\Gamma_i) \geq \mathrm{UB}\}$. 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 3.5.2, we investigate whether it is beneficial to add cuts as so-called lazy or regular constraints.

3.3.2. 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 3.1.: 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 3.1. We formulate our GSPP as the following integer program.

$$\text{[GSPP] Minimize } G(V) = \max_{(i,j,t)=m \in M} \{v_m \cdot (t + p_{ij})\} \tag{3.7}$$

subject to

$$\sum_{\substack{(i,j',t)=m \in M: \\ j'=j}} v_m = 1 \qquad \forall j \in J \tag{3.8}$$

$$\sum_{\substack{(i',j,t')=m\in M:\\ i'=i\wedge t'\leq t<t'+p_{ij}}} v_m \leq 1 \qquad \forall t=1,\ldots,T, i\in P \quad (3.9)$$

$$v_m \in \{0;1\} \qquad \forall m \in M \quad (3.10)$$

Objective function (3.7) minimizes the makespan by minimizing the completion time of the latest job. Constraints (3.8) ensure that for each job exactly one column is selected. Furthermore, inequalities (3.9) take care that in no period more than one job is processed on the same processor. Finally, (3.10) 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 3.5.2. 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 3.5.2). 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.

3.4. 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.

3.4.1. 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 (3.4) and (3.5) 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 3.3.1.2 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 (3.1) 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 (3.4) and (3.5) 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. (3.7)) 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., $M = \{(i,j,t) \in P \times J \times \{1,\ldots,T\} \mid r_j \leq t \leq \overline{d}_j - p_{ij} - q_j\}$.

3.4.2. 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.

3.4.2.1. 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 (3.4) of [Master] with (3.11)

$$\min_{j' \in N_k(j)}\{w_{j'}\} \cdot \left(r_j + \sum_{j' \in N_k(j)} y_{ij'} \cdot p_{ij'} - \max_{j' \in N_k(j)}\{r_{j'}\} \right) \leq z$$
$$\forall i \in P; j \in J; k \in \{1,\ldots,|N(j)|\} \tag{3.11}$$

Valid inequalities (3.11) 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 (3.11): 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 (3.11).

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 3.3.1.2.

Example (cont.): Consider the example from Section 3.3.1.2, 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 3.3. The optimal solution is bold, corresponding to job sequence $\langle 1, 2, 4 \rangle$.

To adjust our GSPP heuristic, only objective function (3.7) of the GSPP model has to be substi-

tuted by (3.12).

$$\text{Minimize } G(V) = \max_{(i,j,t)=m \in M} \{w_j \cdot v_m \cdot (t + p_{ij} - r_j)\} \tag{3.12}$$

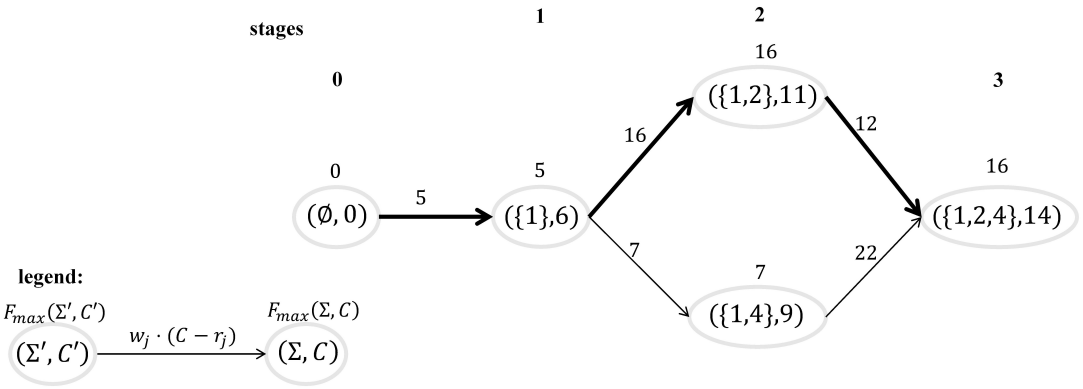


Figure 3.3.: Dynamic programming graph for the slave problem in the example minimizing $wF_{\max}$.

3.4.2.2. 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 (3.4) of [Master] with (3.13)

$$\min_{j' \in V_k(j)} \{w_{j'}\} \cdot \left(\min_{j' \in V_k(j)} \{r_{j'}\} + \sum_{j' \in V_k(j)} y_{ij'} \cdot p_{ij'} - d_j \right) \leq z$$
$$\forall i \in P; j \in J; k \in \{1, \ldots, |V(j)|\}, \tag{3.13}$$

which determines the lower bound on the weighted maximum lateness. The idea behind inequalities (3.13) 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 3.4.1). 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 (3.13) 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 3.3.1.2 have to be ignored because the deadlines are no longer strict.

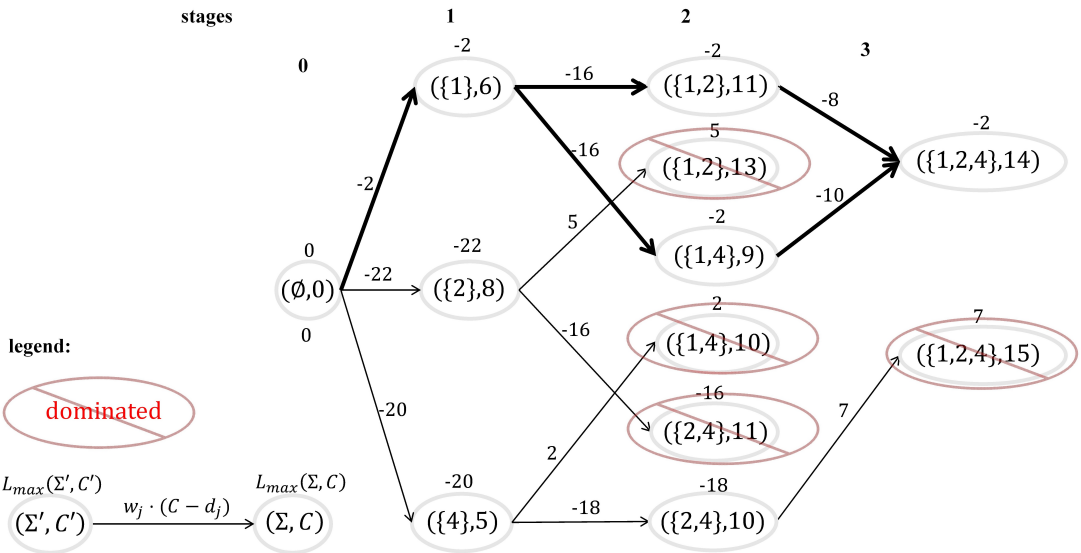


Figure 3.4.: Dynamic programming graph for the slave problem in the example minimizing $L_{\max}$.

Example (cont.): The dynamic programming graph for the example slave problem from Section 3.3.1.2, minimizing the maximum weighted lateness is depicted in Figure 3.4.

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 (3.7) 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 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.

3.5. 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 3.6. 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.

3.5.1. 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 instances 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 3.2 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 3.2.: 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`.

3.5.2. 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 3.3 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 $\mathrm{gap} = \frac{\mathrm{val}' - \mathrm{val}^*}{\mathrm{val}^*} \cdot 100$ 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

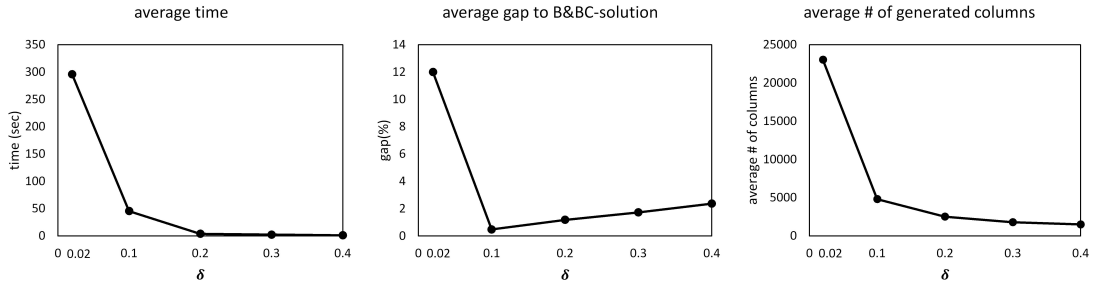


Figure 3.5.: Influence of δ on the solution quality of the GSPP heuristic.

		B&BC		GSPP				
lInstance	$C^*_{\max}$	Lazy cuts Gap/time	Regular cuts Gap/time	$\delta = 0.02$ Gap/time	$\delta = 0.1$ Gap/time	$\delta = 0.2$ Gap/time	$\delta = 0.3$ Gap/time	$\delta = 0.4$ 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 3.3.: Calibration of B&BC (type of cuts) and GSPP heuristic (value of δ).

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 3.5 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 3.5 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 $\delta = 0.1$.

3.5.3. 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 3.4 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 3.5.2, 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

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

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	0.0	0.2	314.9	1.0	1.0	0.0	0.3	1265.3	0.3	0.8	267.9
20x4	3	1.0	0.0	0.1	33.6	0.9	1.0	0.0	5.4	2465.2	0.3	0.2	226.2
20x4	4	1.0	0.0	9.3	851.6	0.6	1.0	0.3	13.7	3689.6	0.3	0.6	225.3
20x6	2	0.9	0.0	30.3	256.4	0.8	1.0	0.2	0.6	1882.7	0.2	0.9	276.3
20x6	3	0.9	0.0	30.5	246.6	0.8	1.0	0.1	9.9	3691.8	0.1	0.1	277.4
20x6	4	0.9	0.0	30.1	918.7	0.7	1.0	0.2	7.8	5489.2	0.4	0.3	215.9
20x8	2	1.0	0.0	0.7	13.6	0.8	1.0	0.2	2.9	2524.8	0.4	0.4	232.6
20x8	3	1.0	0.0	2.1	14.6	0.9	1.0	0.2	10.3	4925.7	0.4	0.0	203.6
20x8	4	1.0	0.0	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	0.0	76.7	31.4	0.2	0.3	8.5	269.5	9642.5	-	-	-
80x15	4	0.6	0.0	121.0	38.8	0.0	0.0	21.0	300.6	14188.2	-	-	-
80x20	2	0.6	0.0	122.4	316.3	0.8	1.0	0.2	111.1	6820.6	-	-	-
80x20	3	0.8	0.0	89.0	74.0	0.3	0.4	8.8	262.0	12877.1	-	-	-
80x20	4	0.8	0.0	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	0.0	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 3.4.: Comparison between B&BC, GSPP and MIP-OP on newly generated instances.

out of memory even before generating the corresponding MIP-OP models for large instances from dataset L.

3.5.4. 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 3.4.1. 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 3.5, 3.6 and 3.7 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 3.5, 3.6 and 3.7 have the same meaning as in Table 3.4 from Section 3.5.3. 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 3.5, 3.6 and 3.7, 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 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.

		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 3.5.: 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 3.6.: 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 3.7.: Comparison between B&BC, GSPP and MIP-OP on BAP minimizing maximum weighted lateness.

3.5.5. 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 3.8 and 3.9 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 3.8.: 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 3.9.: 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.

3.6. 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.

3.A. Appendix

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 3.10.: 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

3.10 summarizes the notation.

$$\text{[MIP-OP] Minimize } C(X,U) = \max_{i \in P} \left\{ \sum_{k \in K} \left(u_{ik} + \sum_{j \in J} p_{ij} \cdot x_{ijk} \right) \right\} \tag{3.14}$$

subject to

$$\sum_{i \in P} \sum_{k \in K} x_{ijk} = 1 \qquad \forall j \in J \tag{3.15}$$

$$\sum_{j \in J} x_{ijk} \leq 1 \qquad \forall i \in P, k \in K \tag{3.16}$$

$$\sum_{j \in J} r_j \cdot x_{ijk} - \sum_{\substack{l \in K: \\ l<k}} \left(u_{il} + \sum_{j \in J} p_{ij} \cdot x_{ijl} \right) \leq u_{ik} \qquad \forall i \in P, k \in K \tag{3.17}$$

$$\sum_{\substack{l \in K: \\ l \leq k}} \left(u_{il} + \sum_{j' \in J} p_{ij'} \cdot x_{ij'l} \right) \leq \overline{d}_j + \mathcal{M} \cdot (1 - x_{ijk}) \qquad \forall i \in P, j \in J, k \in K \tag{3.18}$$

$$u_{ik} \geq 0 \qquad \forall i \in P, k \in K \tag{3.19}$$

$$x_{ijk} \in \{0; 1\} \qquad \forall j \in J, i \in P, k \in K \tag{3.20}$$

Objective function (3.14) 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 (3.15) ensure that every job is assigned to exactly one processor at exactly one service position. Constraints (3.16) make it impossible for multiple jobs to be assigned the same service position on the same machine. Inequalities (3.17) 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 (3.17) and (3.18) indicates the completion time of the $(k-1)$th job on machine i. Constraints (3.18) make sure that the deadlines are not violated. Finally, (3.19) and (3.20) define the domain of the variables.

To extend MIP-OP to account for machine availability restrictions, constraints (3.21) and (3.22)

are added to the model.

$$u_{i1} \geq s_i \qquad \forall i \in P \quad (3.21)$$

$$\sum_{k \in K} \left(u_{ik} + \sum_{j \in J} p_{ij} \cdot x_{ijk} \right) \leq e_i \qquad \forall i \in P \quad (3.22)$$

Constraints (3.21) guarantee that on each machine $i \in P$, processing of its first job cannot be started before its availability time s_i. Constraints (3.22) 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 (3.15)-(3.20) 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 (3.15)-(3.17), (3.19), (3.20), and

$$F^{\text{late}} \geq w_j \left(\sum_{\substack{k' \in K: \\ k' \leq k}} \left(u_{ik'} + \sum_{j' \in J} p_{ij'} \cdot x_{ij'k'} \right) - d_j \right) - \mathcal{M} \cdot (1 - x_{ijk}), \forall i \in P, j \in J, k \in K.$$

3.B. Bibliography

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

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.

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.

Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research*, 37(1):32–41.

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.

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.

Carlier, J. (1982). The one-machine sequencing problem. *European Journal of Operational Research*, 11(1):42 – 47. Third EURO IV Special Issue.

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.

Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

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.

Ebenlendr, T., Krčál, M., and Sgall, J. (2014). Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80.

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.

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.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: a guide to NP-completeness*. WH Freeman and Company, San Francisco.

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.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326.

Hall, N. G. and Posner, M. E. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

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.

Hooker, J. (2011). *Logic-based methods for optimization: combining optimization and constraint satisfaction*, volume 2. John Wiley & Sons, Hoboken, NJ.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55(3):588–602.

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.

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.

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)*.

Konur, D. and Golias, M. M. (2017). Loading time flexibility in cross-docking systems. *Procedia Computer Science*, 114:491–498.

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.

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.

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.

Li, C.-l., Cai, X., and Lee, C.-y. (1998). Scheduling with multiple-job-on-one-processor pattern. *IIE Transactions*, 30(5):433–445.

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.

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.

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, Berlin, Germany, 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.

Sanlaville, E. and Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795–811.

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.

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.

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.

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.

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.

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*.

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.

Paper 4: New solution procedures for the order picker routing problem in U-shaped pick areas with a movable depot[9]

Abstract: This paper develops new solution procedures for the order picker routing problem in U-shaped order picking zones with a movable depot, which has so far only been solved using simple heuristics. The paper presents the first exact solution approach, based on combinatorial Benders decomposition, as well as a heuristic approach based on dynamic programming that extends the idea of the venerable sweep algorithm. In a computational study, we demonstrate that the exact approach can solve small instances well, while the heuristic dynamic programming approach is fast and exhibits an average optimality gap close to zero in all test instances. Moreover, we investigate the influence of various storage assignment policies from the literature and compare them to a newly derived policy that is shown to be advantageous under certain circumstances. Secondly, we investigate the effects of having a movable depot compared to a fixed one and the influence of the effort to move the depot.

Keywords: Order picking; Routing; Storage assignment; U-shaped pick area; Benders decomposition; Dynamic programming

[9]This chapter has been published as: Diefenbach, H., Emde, S., Glock, C. H., and Grosse, E. H. (2022). New solution procedures for the order picker routing problem in U-shaped pick areas with a movable depot. *OR Spectrum*, 44(2), 535-573. DOI: `https://doi.org/10.1007/s00291-021-00663-8`

4.1. Introduction

The management of warehouse operations has received ample attention over many years. Especially order picking, which is commonly described as the retrieval of products from storage locations to fulfill customer orders, is on the top of many research agendas (Van Gils et al., 2018). This is because of, firstly, the high amount of manual human labor that is usually associated with picking orders (Grosse et al., 2015), and, secondly, the fact that order picking is a very time-intensive activity with direct impact on customer service (De Koster et al., 2007). For example, in the United States, more than 3.7 million people are employed in warehousing as manual laborers and material movers (Bureau of Labor Statistics, 2016). In the European Union's warehousing and transport support sector, 2.6 million persons are employed (Eurostat, 2016). These facts render order picking one of the most important cost factors in warehousing (Tompkins et al., 2010, Rushton et al., 2014).

To reduce the cost of order picking, researchers have developed various mathematical models in the past that support warehouse managers in assigning products to shelf locations, in restructuring incoming orders and in routing order pickers through the warehouse (Van Gils et al., 2018). It is usually advisable to adapt planning procedures to the specific layout of the warehouse (Roodbergen et al., 2015). Warehouse layouts that have been studied in research on order picking in the past include layouts of rectangular shape, which are often denoted as conventional warehouses, either with a single block (e.g., Petersen et al. (2005), Grosse et al. (2014)) or with two or more blocks (e.g., Roodbergen and Koster, 2001a, Roodbergen et al., 2015). Non-conventional warehouses, such as leaf, chevron or flying-V, are employed less frequently in practice but still play a significant role (e.g., Masae et al., 2020b, 2021).

U-shaped layouts of warehouse zones, which are sometimes also referred to as "picker nests", can be observed quite often in practice, but have not received much attention in the literature so far. Glock and Grosse (2012), for example, studied order picking in a U-shaped zone and developed procedures for assigning products to shelf locations, for finding a location for the depot of the order picker, and for routing the order picker through the U-zone. Inspired by a practical case in automotive part picking, the authors also introduced the concept of a movable depot: the depot from which the picker sets off and to which she returns need not be at a fixed location, but can be moved within certain limits, potentially shortening pick tours. Put differently, the depot location becomes itself a variable to be optimized. Given the special structure of the pick zone, the authors used a simple sweep algorithm to solve the routing problem. Diefenbach and Glock (2019) also studied a U-shaped warehouse and optimized the layout and item assignment

for single command picking with regard to two different objectives, namely pick efficiency and ergonomics. They did, however, not study the routing problem since it is not relevant for single command picking.

The paper at hand revisits the setting studied by Glock and Grosse (2012) and extends the existing work by the following contributions:

- We develop the first exact solution procedure for the picker routing problem in U-shaped order picking zones, namely an algorithm based on combinatorial Benders decomposition.
- In a comprehensive numerical study, we compare the solutions of our newly developed exact procedure to the solutions of the sweep algorithm developed by Glock and Grosse (2012) to analyze the latter's solution quality, which had not been done yet.
- We develop a new heuristic approach extending the idea of the sweep algorithm, based on dynamic programming. The newly developed procedure compares favorably in theory and in our numerical experiments.
- In addition, we derive some managerial insights from our numerical experiments. We propose a new radial storage assignment policy that better matches the specific characteristics of a U-shaped order picking zone, compare it to storage assignment policies from the literature, and demonstrate its advantage in certain situations. Furthermore, we investigate the effects of having a movable compared to a fixed depot and the influence of the effort for moving the depot.

The remainder of this paper is structured as follows: Section 4.2 discusses the related literature. Section 4.3 formally defines the picker routing problem studied in this paper, while Section 4.4 presents exact and heuristic solution methods. Section 4.5 presents the results of numerical experiments, and Section 4.6 concludes the paper.

4.2. Literature review

Researchers have developed numerous mathematical models and algorithms to provide managerial decision support in planning manual order picking operations. The aim of these works has mainly been the reduction of order picking time or travel distance and thus the minimization of costs (see, for reviews, Gu et al., 2007, De Koster et al., 2007, Grosse et al., 2017, Masae et al., 2020a)). To reach this goal, several planning problems have to be addressed. These include lay-

out design, routing, storage assignment, and order batching. The reader is referred to the review of Van Gils et al. (2018) for a detailed overview of order picking planning models.

Works on layout design mostly deal with the definition of a suitable warehouse layout, which includes decisions about the number of storage blocks, cross aisles, parallel aisles, as well as height and depth of racks (e.g., Vaughan, 1999, Roodbergen and Vis, 2006, Roodbergen et al., 2008, 2015). Here, it is important to keep space requirements and other types of restrictions in mind when determining the width of aisles, which can be narrow, wide or mixed (Mowrey and Parikh, 2014). The majority of works studies rectangular/conventional layouts, whereas alternative layouts for manual order picking areas are rather rare (Masae et al., 2020a). However, alternative layouts for order picking areas are quite common in practice. These include U-shaped layouts, where shelves or pallets are arranged in the shape of a U within the order picking area (Glock and Grosse, 2012, Diefenbach and Glock, 2019). We note that Henn et al. (2013) also refer to their considered layout as U-shaped. It is, however, fundamentally different from the one considered in this paper as it resembles a more conventional warehouse layout, where the cross aisles are arranged in the shape of an H or U. Other alternative layouts such as *fishbone* (Gue and Meller, 2009) or *flying-V* designs (Öztürköğlu et al., 2014) are proposed for unit-load warehouses, where products are picked in pallet quantities.

Routing methods that guide order pickers through the warehouse on preferably shortest routes are mainly developed for conventional warehouses. An exact algorithm that calculates shortest routes exists for one-block warehouses, solving a special case of the traveling salesman problem (Ratliff and Rosenthal, 1983, Scholz et al., 2016, Lu et al., 2016, Chabot et al., 2017, Masae et al., 2020a). Although this exact algorithm exists, many authors studied simple routing heuristics (such as the well-known s-shape heuristic) because these easy-to-follow patterns are often applied in practice (Petersen and Aase, 2004, Glock et al., 2017). For special cases of rectangular warehouses with more cross aisles, extensions of this exact algorithm (Roodbergen and Koster, 2001a, Pansart et al., 2018) as well as heuristic solution approaches exist (Roodbergen and Koster, 2001b, Theys et al., 2010, Çelik and Süral, 2019, Chen et al., 2019). For alternative layouts, Masae et al. (2020b), for example, propose an exact algorithm to solve the picker routing problem in the chevron layout as well as in the leaf warehouse (Masae et al., 2021), and Çelk and Süral (2014) for the fishbone layout. In U-shaped order picking areas, Glock and Grosse (2012) propose a sweep algorithm to calculate order picking routes. In a similar setting, Glock et al. (2019) assume a sufficient capacity of the order picker to transport all requested items in a single tour, which simplifies the routing problem.

Storage assignment methods assign items to storage positions (Reyes et al., 2019). This can

either be random or according to some criteria, such as item demand or volume (Füßler et al., 2019). Common advice for practitioners is to assign frequently requested items to storage locations close to the depot (also denoted as *pick-up/drop-off point*, see Petersen and Aase (2004)). In rectangular warehouses, a common approach is to define item classes (typically A, B, and C items according to demand frequency), which are then assigned to specific aisles or zones (De Koster et al., 2007). Petersen et al. (2004) propose several patterns for class-based storage to classify aisles, e.g., within-aisle, diagonal or rectangular strategies. Several algorithms for class-based storage assignment exist (Muppani and Adil, 2008). For U-shaped layouts, Glock and Grosse (2012) propose dedicated storage assignment methods (i.e. horizontal, vertical, and upper/lower assignments). Moreover, further factors can be considered for storage assignment models, such as precedence constraints, item weight (Žulj et al., 2018a), or other objectives than the minimization of travel distance, for example, the minimization of human energy expenditure (Battini et al., 2016, Calzavara et al., 2017, 2019) or workload (Otto et al., 2017, Glock et al., 2019).

Consolidating or splitting up orders, which is commonly denoted as order batching, can save on travel distance (Cergibozan and Tasan, 2019). Only small instances of order batching problems can be solved optimally in reasonable time (Gademann, 2005), which is why many researchers propose heuristic or metaheuristic approaches to address this problem (Hong et al., 2012, Henn and Wäscher, 2012, Matusiak et al., 2014, Pan et al., 2015, Žulj et al., 2018b). Other authors develop metaheuristic algorithms to solve the combined order batching and picker routing problems in an integrated fashion (Kulak et al., 2012, Grosse et al., 2014, Van Gils et al., 2019).

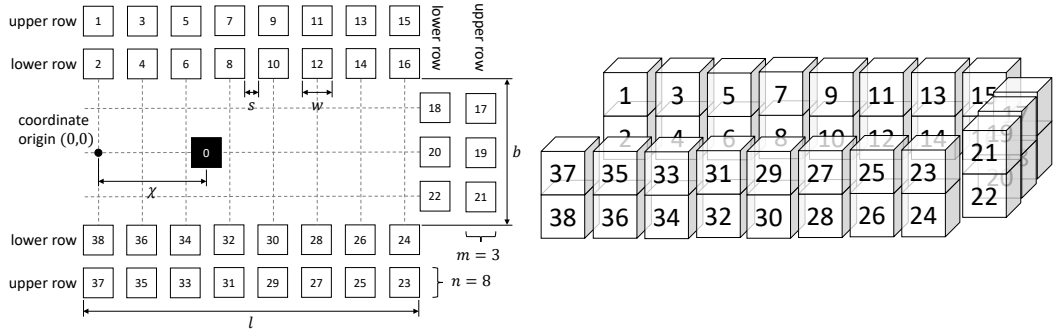
This paper addresses two out of the discussed four planning problems, contributing new insights in this research field, namely developing a new heuristic and an exact algorithm based on combinatorial Benders decomposition for the picker-routing problem within the U-zone and proposing a new radial storage assignment method for this special layout.

4.3. Problem description

This paper studies order picking in a U-shaped order picking area as outlined in Glock and Grosse (2012). In the considered warehouse setting, items are stored in stillages (i.e., large boxes that can be accessed from the front), with two rows of stillages stacked one atop the other. Each U-zone consists of two horizontal and one vertical shelf as illustrated in Figure 4.1. The depot, where each order picking tour starts and ends, is also a stillage that is brought to and removed from the U-zone by a forklift truck, and it is represented by the black box in Figure

4.1a. The picker travels on foot along the shelves of the U-zone, possibly pushing or pulling a cart or a related device. U-shaped order picking areas, as the one studied in this paper, can frequently be observed in practice, for example in the automotive or chemical industries (Glock and Grosse, 2012, Glock et al., 2019).

In these industries, it is common to prepare so-called kits to supply assembly workplaces with the required materials. For preparing the kits, compact work zones are established in the warehouse that contain the items required at one or more assembly workplaces (e.g., Hanson et al., 2017). Especially in cases where only a small number of items is stored in the kitting zones, U-zones are beneficial because of a clear separation of items and good item accessibility.



(a) Two-dimensional representation.

(b) Three-dimensional representation.

Figure 4.1.: Layout of one U-zone with 38 stillages ($n = 8$, $m = 3$) and one depot.

4.3.1. Formal description of the picker routing problem

To model the problem concisely, we make the same assumptions as in Glock and Grosse (2012) and assume a U-zone arranged as in Figure 4.1:

- We consider a picker processing a single order in a single U-zone. An order constitutes a set Ω of items that need to be collected and placed together in the depot – for example, a kit destined for a single production station. During a shift, a picker processes multiple orders. We assume orders are planned beforehand and given from the perspective of our problem, such that order picker routing for a single order is independent from other orders.

- The U-zone's coordinate system is two-dimensional, and the depot can be placed anywhere on the center line of the U-zone (i.e., the y-coordinate of the depot is always 0). The location of the depot has to be fixed before the order picker starts processing the order. Moving the depot from the open end of the U deeper into the zone consumes a certain

amount of time proportional to the distance that the depot is moved.

- Euclidean distances are used to calculate the travel distance of the order picker, as we assume that this is the most intuitive way to travel through the pick zone. Formally, we define the Euclidean distance between two points $P_1 = (x_1^p, y_1^p)$ and $P_2 = (x_2^p, y_2^p)$ as $D^e(P_1, P_2) = \sqrt{(x_1^p - x_2^p)^2 + (y_1^p - y_2^p)^2}$. The distance between stillages is calculated to the center of each stillage or the depot.

- We do not consider service/picking times, because they can be considered constant for a given picklist and are therefore not affected by the picker routes. Moreover, we equate travel distances with travel times. Note that, given a fixed average movement speed of the picker, distances can be transformed into durations by simple multiplication with a constant factor.

- Stillages are numbered in a clockwise manner starting at the upper left corner (see Figure 4.1).

- The storage assignment policy is selected prior to the start of the order picking process (see Section 4.5.4). Once items have been assigned to storage locations, the assignment is kept constant until all orders have been completed. Each kind of item is stored in a single stillage and each stillage contains only one kind of item. This implies that items have to fit in a single stillage, which is usually the case in practice for U-shaped order picking zones.

- The order picker must return to the depot when he/she has finished his/her order or when the transport capacity of his/her picking device has been reached. In the latter case, after returning to the depot to drop off items there, he/she can continue picking items. After an order has been completed, the depot is removed, and a new depot is brought to the U-zone together with a new order (pick-by-order).

- The demand of any item does not exceed the carrying capacity of the picker.

We consider a pick area with a layout as depicted in Figure 4.1, where stillages are arranged in a U-shape around a picking station (depot). The width of a stillage is w, and to facilitate picking/exchanging stillages, the gap between them is s. The entire length of a zone can then be determined as $l = n \cdot w + (n-1) \cdot s$, where n is the number of horizontal stillages in the zone. The width of the zone is $b = m \cdot w + (m+1) \cdot s$, where m is the number of vertical shelves in the zone. The coordinates of the depot are $(\chi, 0)$. For the first 1 to $2 \cdot n$ stillages, the coordinates of the i-th stillage are $\left(\lceil \frac{i-1}{2} \rceil \cdot (w+s), \frac{b}{2}\right)$. For stillages numbered $i = 2 \cdot n + 1, \ldots, 2 \cdot n + m$, the coordinates are $\left(l - \frac{w}{2}, \lceil \frac{2 \cdot n + m - i}{2} \rceil \cdot (w+s)\right)$ for uneven values

of m, and $\left(l - \frac{w}{2}, \left(\lceil \frac{2 \cdot n + m - i - 1}{2} \rceil + \frac{1}{2}\right) \cdot (w + s)\right)$ for even values of m. Stillages with index numbers $i = 2 \cdot n + m + 1, \ldots, 4 \cdot n$ are mirror images of the first $2 \cdot n$ stillages along the x-axis. The distance between any two locations i and i' is

$$d_{i,i'} = D^e\left((x_i, y_i), (x_{i'}, y_{i'})\right) = \sqrt{(x_i - x_{i'})^2 + (y_i - y_{i'})^2}. \tag{4.1}$$

Let $I = \{1, \ldots, |I|\}$ be the set of stillages, and let $\Omega = \{1, \ldots, |\Omega|\}$ be the set of items that need to be picked for a given order with $|\Omega|$ items. Let $\iota(j)$ be the stillage i where item j is stored, and let the items be indexed in the same clockwise order as their respective stillages, i.e., let $\iota(j) \leq \iota(j'), \forall j, j' \in \Omega : j < j'$, be true. Then, item j is located at coordinates $(x_{\iota(j)}, y_{\iota(j)})$. To shorten notation, we define $\tilde{x}_j = x_{\iota(j)}$ and $\tilde{y}_j = y_{\iota(j)}$ as well as $d_{\iota(j),\iota(j')} = \tilde{d}_{j,j'}$. Moreover, let Q be the limited carrying capacity of the picker, and let q_j be the weight of item $j \in \Omega$.

	sets
I	set of stillages
Ω	set of items to be picked (indices i, j)
ω_k	variable: set of items to be picked on tour $k \in \{1, \ldots, r\}$
$\tilde{\omega}_k$	auxiliary variable: set of stillages to be visited on tour $k \in \{1, \ldots, r\}$
	parameters
$d_{i,i'}$	distance from stillage i to stillage i' (meters)
$\tilde{d}_{j,j'}$	distance between the stillage containing item j and the stillage containing item j'
Q	maximum carrying capacity of the order picker (kilograms)
q_j	weight of item j (kilograms)
v	penalty distance factor for moving the depot
b	width of the aisle (meters)
l	length of the aisle (meters)
n	number of stillages in one row of horizontal shelves
m	number of stillages in the vertical shelf
s	distance/gap between two adjacent stillages (meters)
w	width of a stillage (meters)
x_i	x-coordinate of stillage i (meters)
y_i	y-coordinate of stillage i (meters)
$\tilde{x}_j$	x-coordinate of the stillage containing item j (meters)
$\tilde{y}_j$	y-coordinate of the stillage containing item j (meters)

Table 4.1.: Notation for the picker routing problem.

We look for a partition of Ω into r subsets $\{\omega_1, \ldots, \omega_r\}$ such that the total weight of the items in each set ω_k does not exceed the carrying capacity of the picker, i.e., $\sum_{j \in \omega_k} q_j \leq Q, \forall k = 1, \ldots, r$. Note that the number r of pick tours is not given in advance. Each ω_k stands for one pick tour the picker makes, starting from the depot, visiting all stillages implied by ω_k,

and returning to the depot. For brevity of notation, we introduce sets $\tilde{\omega}_k = \{\iota(j) \mid j \in \omega_k\}$, $\forall k = 1, \ldots, r$, to denote the stillages to be visited to pick the items in ω_k.

Furthermore, we look for a position χ of the depot. The depot can be moved along the center line of the U-zone with the default position located at the open end of the U-zone with $\chi = 0$ (see Figure 4.1a). If the depot is moved along the aisle, the warehouse worker transporting the depot has to travel an extra distance into the U-zone both when bringing and removing the depot, which leads to a time penalty that has to be considered in the model and which depends on the extra travel distance equaling $2 \cdot \chi$. Assuming that the picker can walk $2 \cdot v$ times faster (or slower, as the case may be) when placing the depot than his/her speed during the order picking process, the additional distance that has to be covered just to move the depot is $\frac{1}{v} \cdot \chi$. All symbols are summarized in Table 4.1.

A solution to our picker routing problem thus consists of a partition $\{\omega_1, \ldots, \omega_r\}$ of Ω and the depot position $0 \leq \chi \leq l - \frac{w}{2}$. Among all feasible solutions we seek one where the total distance travelled by the picker – including the penalty distance to move the depot and the distance to visit the stillages – is minimal.

Routing the picker for a given set of stillages $\tilde{\omega}_k$ is technically a travelling salesman problem, which is well-known to be strongly NP-hard (Garey and Johnson, 1979). However, due to the special structure of the U-shaped pick area, the routing problem is actually tractable.

Proposition 4.1. *For a given set $\tilde{\omega}_k$ of stillages to be visited, an optimal route (i.e., sequence of visits) with regard to total travel distance is to move through the stillages in $\tilde{\omega}_k$ in clockwise order in the shape of a polygon without intersecting edges.*

Proof. Barachet (1957) shows that, in a Euclidean TSP, an optimal TSP tour never crosses itself. Hence, the optimal tour is in the shape of a polygon without intersecting edges, where an edge touching another edge (at a vertex) counts as an intersection as well. Moreover, this polygon is contained within the convex hull around all points to be visited.

Let the points that lie on the convex hull be labeled in clockwise order. Two points i and i', where $i' \geq i + 2$, can never be connected directly in the optimal route. This is because a connection between i and i' would separate the convex hull into two parts, where one contains the points $i'' \in \{i + 1, \ldots, i' - 1\}$ and the other contains the remaining points. Since the optimal route does not cross itself, there exists no optimal route that connects the points from the separate parts, because it would intersect the connection between i and i'. It follows that in the optimal

route, each point i on the convex hull can only be connected with its neighboring points $i-1$ and $i+1$ on the convex hull or with points that do not lie on the convex hull.

Clearly, all stillages in a U-shaped picking zone lie on a convex polygon, which also constitutes the convex hull. The only point that can possibly not lie on the convex hull is the depot. Hence, in the optimal route, every stillage must be connected to its neighboring stillages (i.e., its predecessor and successor in a clockwise order) or to the depot. Furthermore, the optimal route has only two connections to the depot, one outgoing and one incoming. If the depot lies on the convex hull, it must be connected to its neighboring stillages. If not, the depot must be connected to two consecutive stillages, due to the same argument as before. If the depot would not be connected to two consecutive stillages, the connection would separate the convex hull into two parts that cannot be connected by a route without an intersection. Consequently, the optimal route is to move from the depot to a stillage, move through the stillages in $\tilde{\omega}_k$ in clockwise (or counter-clockwise) order in the shape of a polygon without intersecting edges, and move back to the depot. □

Let the pair (j, j') denote an edge between stillages $\iota(j)$ and $\iota(j')$ and let $\eta_k = \big\{(j,j') \in \omega_k \times \omega_k : \iota(j) \leq \iota(j') \wedge \{j'' \in \omega_k \mid \iota(j) \leq \iota(j'') \leq \iota(j')\} = \emptyset\big\} \cup \{(\max\{\omega_k\}, \min\{\omega_k\})\}$ be the set of all edges of the convex polygon spanned by the stillages in $\tilde{\omega}_k$. The optimal route's length can then be formalized as

$$\begin{aligned} g(\omega_k, \chi) &= \sum_{(j,j')\in\eta_k} \tilde{d}_{j,j'} + \min_{(j,j')\in\eta_k} \left\{ -\tilde{d}_{j,j'} + \sqrt{(\tilde{x}_j - \chi)^2 + \tilde{y}_j^2} + \sqrt{(\tilde{x}_{j'} - \chi)^2 + \tilde{y}_{j'}^2} \right\} \\ &= \sum_{(j,j')\in\eta_k} \tilde{d}_{j,j'} + \min_{(j,j')\in\eta_k} \left\{ \gamma_{j,j'}(\chi) \right\}, \end{aligned} \tag{4.2}$$

where we define $\gamma_{j,j'}(\chi) = -\tilde{d}_{j,j'} + \sqrt{(\tilde{x}_j - \chi)^2 + \tilde{y}_j^2} + \sqrt{(\tilde{x}_{j'} - \chi)^2 + \tilde{y}_{j'}^2}$ for ease of notation. Note that the first term of Eq. (4.2) stands for the travel distance of the picker along the U, while the second term is the distance from and to the depot, where, by Proposition 4.1, it is optimal to insert the depot visit in-between the two neighboring stillages from $\tilde{\omega}_k$ that minimize the total distance.

The total objective value of a solution consists of the travel distance of the pick tours plus the time to position the depot in the first place, and it is hence

$$G(\omega_1, \ldots, \omega_r, \chi) = \sum_{k\in\{1,\ldots,r\}} g(\omega_k, \chi) + \frac{1}{v} \cdot \chi. \tag{4.3}$$

Among all feasible solutions consisting of partition $\{\omega_1, \ldots, \omega_r\}$ and depot location χ, we seek one which minimizes $G(\omega_1, \ldots, \omega_r, \chi)$. We refer to this problem as the *picker routing problem in a U-shaped pick area* (PRP-UA).

4.3.2. Computational complexity

Given that, by Proposition 4.1, routing in a U-shaped pick zone is computationally easier than on general graphs, it may seem that picker routing in U-shaped zones may not be a hard problem at all. However, PRP-UA is intractable as can be seen by the following proposition.

Proposition 4.2. *Solving PRP-UA is NP-hard in the strong sense.*

Proof. We prove Proposition 4.2 by reduction from bin packing, which is well known to be strongly NP-hard (Garey and Johnson, 1979).

The decision version of bin packing is concerned with the following question. Given a set S of items i with associated weight w_i and bins with capacity C, does there exists a partition of items into bins such that no subset of items assigned to the same bin exceeds the bin's capacity C and at most k bins are used?

We propose the following transformation from an instance of bin packing to an instance of PRP-UA. Firstly, we set the measurements l and b of the U-zone such that $2 \cdot k \cdot l < b$ holds (i.e., we set n and m such that $2 \cdot k \cdot (n \cdot w + (n-1) \cdot s) < m \cdot w + (m+1) \cdot s$ holds). Secondly, we associate each item $i \in S$ of the bin packing instance with an item $j \in \Omega$ of the PRP-UA instance (with $w_i = q_j$ for each associated pair of items) and assign the items $j \in \Omega$ to (arbitrary) stillages in the upper row (i.e., in positive y-direction, cf., Figure 4.1a) of the U-layout. Finally, we set $Q = C$ and $v \to \infty$ (i.e., $\frac{1}{v} \to 0$).

A solution of an instance of PRP-UA corresponds to a solution of an instance of bin packing if and only if the objective value is less than $(k+1) \cdot b$, as is shown in the following. Clearly, each route cannot contain a subset of items that exceeds the maximum capacity Q, such that we can associate routes with bins. For PRP-UA, by Proposition 4.1, the length of a single optimal route is bounded from above by $b + 2 \cdot l$, i.e., the length of the route if the picker visits every single stillage in the upper row on one tour cannot be longer than this. Likewise, a route's minimum length is b, i.e., visiting only a single stillage directly above the depot cannot be shorter than this. Hence, a solution using $k+1$ tours or more has at least an objective value of $(k+1) \cdot b$

and a solution using at most k tours has at most an objective value of $k \cdot (b + 2 \cdot l) < k \cdot b + b$ (since we set $2 \cdot k \cdot l < b$). Hence, if and only if the objective value of a solution to PRP-UA is less than $(k + 1) \cdot b$, it contains no more than k tours, which corresponds to a solution for the corresponding instance of bin packing with at most k bins.

Since the decision version of PRP-UA is strongly NP-complete, the corresponding optimization version is NP-hard in the strong sense, which completes the proof. □

4.4. Algorithms

In the following, we present new algorithms to solve PRP-UA. Section 4.4.1 presents the first exact solution approach based on combinatorial Benders decomposition. With PRP-UA being NP-hard, we can expect that larger problem instances cannot be solved exactly, as is also shown in our computational study later on (cf. Section 4.5.2). We therefore present a new heuristic solution approach based on the concept of dynamic programming in Section 4.4.2.

4.4.1. Logic-based Benders decomposition for the picker routing problem

In addition to being NP-hard, PRP-UA is further complicated by the presence of non-linear (Euclidean) distances, which depend on a variable, namely the position of the depot. There is therefore no obvious way of formulating a compact (mixed-integer) linear programming model without discretization of the depot location χ. For discrete depot locations, the problem would become a capacitated vehicle routing problem. If we disregard the routing aspect and the moveable depot, the remaining problem, i.e., batching items on tours such that the picker capacity is not violated, is a bin packing problem. To avoid making a heuristic choice regarding discretization intervals, in the following, we focus on an exact approach where we consider the depot location χ as a continuous variable.

To make the problem more tractable, we propose a decomposition scheme in the spirit of logic-based and combinatorial Benders decomposition (Codato and Fischetti, 2006, Hooker, 2007), which has seen success dealing with difficult combinatorial optimization problems (e.g., Kress et al., 2019, Tadumadze et al., 2020, Fang et al., 2021, Huang et al., 2021). The general idea consists of splitting the original problem into a master and a slave component. The master problem is modeled as a mixed-integer linear programming model that is solved by an off-the-shelf default solver. Whenever the solver finds a candidate integer solution for this model, the

solution is passed to the slave problem, which calculates the optimal objective value for the given master solution. From the slave solution, combinatorial cuts are generated, which remove suboptimal solutions from the master model. The solver then continues working on the master model with the newly added cuts, passing candidate solutions to the slave problem until no more feasible, undiscarded solutions remain. The best incumbent solution at this point is optimal. For brevity, we refer to this algorithm as *CBD* (combinatorial Benders decomposition).

In Section 4.4.1.1, we describe the master model for our picker routing problem in detail. In Section 4.4.1.2 we present the slave model and describe how it can be efficiently solved. Section 4.4.1.3 describes how we generate cuts from solutions of the slave problem.

4.4.1.1. Master problem

The master problem (MP) consists of batching items on tours, i.e., effectively, determining sets ω_k, $\forall k$. For a given batching, the exact objective value and optimal depot location is then determined by solving the slave problem described in Section 4.4.1.2. We use binary variables $z_{j,j'}$, which have value 1 if and only if item j' is on the same tour as item j and j' is the item with the greatest index on that tour. Formally, the feasible search space of the master model is described by the following constraints.

$$\sum_{\substack{j' \in \Omega: \\ j \leq j'}} z_{j,j'} = 1 \qquad \forall j \in \Omega \quad (4.4)$$

$$z_{j,j'} \leq z_{j',j'} \qquad \forall j, j' \in \Omega : j < j' \quad (4.5)$$

$$\sum_{\substack{j \in \Omega: \\ j \leq j'}} q_j \cdot z_{j,j'} \leq Q \qquad \forall j' \in \Omega \quad (4.6)$$

$$z_{j,j'} \in \{0, 1\} \qquad \forall j, j' \in \Omega : j \leq j' \quad (4.7)$$

Constraints (4.4) ensure that each item is on exactly one pick tour. If some item j' is on the same tour as item j such that j' has the highest index in that tour, then $z_{j,j'}$ is forced to 1, as ensured by Inequalities (4.5). Inequalities (4.6) ensure that the picker's carrying capacity is not exceeded, while Constraints (4.7) define the domain of the decision variables.

While the master model can be solved as a pure feasibility problem, this may not be advisable from a performance viewpoint. Without any objective to guide the search, we can expect many mediocre solutions to be evaluated. We therefore use a lower bound on the objective value as a

subproblem relaxation (Hooker, 2007) based on the following idea.

Proposition 4.1 states that the optimal route to visit a set $\tilde{\omega}_k$ of stillages and the depot is in the form of a polygon without crossing edges. Clearly, the edge length of the *convex* polygon spanned by the stillages in $\tilde{\omega}_k$ and the depot is a lower bound on the optimal tour length. Given the non-linear Euclidean distances, there is no easy method to calculate the respective convex polygon's edge length in a compact linear model. However, we can calculate lower bounds on the edge length in two ways: first, by using the rectilinear metric (Manhattan metric) and, second, by using the maximum metric (Chebyshev distance). Formally, for two points $P_1 = (x_1^p, y_1^p)$ and $P_2 = (x_2^p, y_2^p)$, we define the rectilinear metric distance as $D^r(P_1, P_2) = |x_1^p - x_2^p| + |y_1^p - y_2^p|$ and the maximum metric distance as $D^m(P_1, P_2) = \max\{|x_1^p - x_2^p|, |y_1^p - y_2^p|\}$.

For the first bound, we apply Proposition 4.3.

Proposition 4.3. *A convex polygon's Euclidean (i.e., actual) circumference is at least its rectilinear metric circumference divided by* $\sqrt{2}$.

Proof. The proof is based on the labeling in Figure 4.2a. The circumference of a polygon with E vertices is given by $C^{\text{poly}} = \sum_{i=1}^{E-1} D^e(P_i, P_{i+1}) + D^e(P_E, P_1)$. In two-dimensional space, the distance between two points measured in the rectilinear metric is at most $\sqrt{2}$ times greater than the distances measured in the Euclidean metric, i.e., $D^e(P_i, P_{i'}) \geq \frac{1}{\sqrt{2}} \cdot D^r(P_i, P_{i'})$ holds (cf., Proposition 4.5 in the Appendix). It follows that $C^{\text{poly}} \geq \sum_{i=1}^{E-1} \frac{1}{\sqrt{2}} \cdot D^r(P_i, P_{i+1}) + \frac{1}{\sqrt{2}} \cdot D^r(P_E, P_1) \Rightarrow C^{\text{poly}} \geq \frac{1}{\sqrt{2}} \cdot \left(\sum_{i=1}^{E-1} D^r(P_i, P_{i+1}) + D^r(P_E, P_1)\right) \Rightarrow C^{\text{poly}} \geq \frac{1}{\sqrt{2}} \cdot C^{\text{rect}}$, where C^{rect} is the polygon's circumference in the rectilinear metric. □

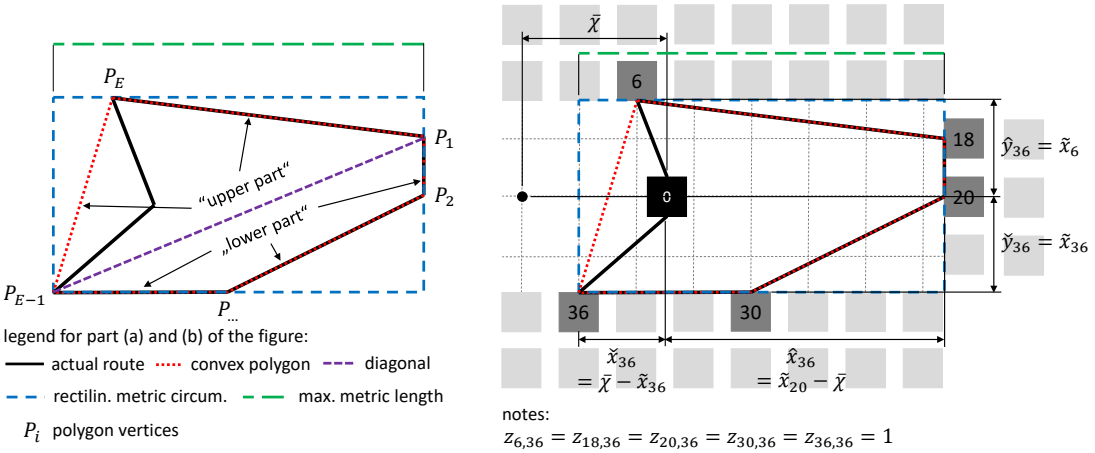
Hence, we can calculate the convex polygon's edge length in the rectilinear metric and divide it by $\sqrt{2}$ to attain a lower bound for the actual length. For the second bound, we apply Proposition 4.4.

Proposition 4.4. *A convex polygon's Euclidean (i.e., actual) circumference is at least two times its length in the maximum metric.*

Proof. The proof is based on the labeling in Figure 4.2a. Assume any diagonal of the polygon connecting the two shorter sides of its rectilinear circumference. Such a diagonal always exists, since each side of the polygon's rectilinear circumference is connected to at least one of the polygon's vertices. The diagonal splits the polygon into two parts, which we label *upper part*

and *lower part* in the following. By the triangle inequality, the polygon's upper part edge length is longer than the diagonal. The same applies for the lower part's edge length. Therefore, the polygon's Euclidean circumference is at least two times the length of the diagonal. Again by the triangle inequality, the diagonal length is never less than the polygon's maximum length, which is equal to the longer side of the polygon's rectilinear circumference. Hence, two times the polygon's maximum length is never greater than its Euclidean circumference. □

Therefore, two times the convex polygon's length in the maximum metric is also a lower bound for the actual (Euclidean) circumference.



(a) Concept of using the rectilinear and maximum metric to formulate lower bounds on a convex polygon's circumference.

(b) Illustration of the variables $\check{x}_j$, $\check{y}_j$, $\hat{x}_j$, and $\hat{y}_j$ being used to calculate lower bounds on a route's length.

Figure 4.2.: Lower bounds on a picking route's length.

To calculate the bounds, we introduce four sets of auxiliary continuous variables to the master model. Variables $\check{x}_j$ ($\check{y}_j$) denote the distance from the depot to the leftmost (bottommost) stillage on the tour that contains item j. Analogously, variables $\hat{x}_j$ ($\hat{y}_j$) denote the distance from the depot to the rightmost (topmost) stillage on the tour that contains item j. Moreover, we introduce auxiliary variable $\bar{\chi}$ to denote the position of the movable depot. An illustrative example is given in Figure 4.2b. We consider the bounds by adding the following objective and valid inequalities to the master model:

$$\text{[MP] Minimize } G = \sum_{j \in \Omega} \hat{d}_j + \frac{1}{v} \cdot \bar{\chi} \tag{4.8}$$

subject to (4.4) - (4.7) and

$$\hat{x}_j \geq \tilde{x}_{j'} \cdot z_{j',j} - \bar{\chi} \qquad \forall j, j' \in \Omega : j' \leq j \quad (4.9)$$

$$\check{x}_j \geq \bar{\chi} - \left(\left(\tilde{x}_{j'} - l\right) \cdot z_{j',j} + l\right) \qquad \forall j, j' \in \Omega : j' \leq j \quad (4.10)$$

$$\hat{y}_j \geq \tilde{y}_{j'} \cdot z_{j',j} \qquad \forall j, j' \in \Omega : j' \leq j \quad (4.11)$$

$$\check{y}_j \geq -\tilde{y}_{j'} \cdot z_{j',j} \qquad \forall j, j' \in \Omega : j' \leq j \quad (4.12)$$

$$\hat{d}_j \geq 2 \cdot \frac{1}{\sqrt{2}} \cdot (\hat{x}_j + \check{x}_j + \hat{y}_j + \check{y}_j) \qquad \forall j \in \Omega \quad (4.13)$$

$$\hat{d}_j \geq 2 \cdot (\hat{x}_j + \check{x}_j) \qquad \forall j \in \Omega \quad (4.14)$$

$$\hat{d}_j \geq 2 \cdot (\hat{y}_j + \check{y}_j) \qquad \forall j \in \Omega \quad (4.15)$$

$$\hat{x}_j, \check{x}_j \in \left[0, \max_{i \in I} \{x_i\}\right] \qquad \forall j \in \Omega \quad (4.16)$$

$$\hat{y}_j, \check{y}_j \in \left[0, \max_{i \in I} \{|y_i|\}\right] \qquad \forall j \in \Omega \quad (4.17)$$

$$\hat{d}_j \in [0, 2 \cdot l + 4 \cdot b] \qquad \forall j \in \Omega \quad (4.18)$$

$$\bar{\chi} \in \left[0, l - \frac{w}{2}\right] \quad (4.19)$$

Objective (4.8) minimizes the sum of the distance approximations of the routes plus the cost to move the depot. For each route, Inequalities (4.9) determine the distance between the depot and the polygon's rightmost vertex. If the depot is to the right of the polygon's rightmost vertex, $\hat{x}_j$ assumes 0. Similarly, Inequalities (4.10) calculate the distance between the depot and the polygon's leftmost vertex. If the depot is to the left of the polygon's leftmost vertex, $\check{x}_j$ assumes 0. Inequalities (4.11) and (4.12) calculate the respective distances on the y-axis for every tour. Constraints (4.13) calculate lower bounds on the tour length based on the rectilinear metric. Constraints (4.14) and (4.15) determine respective lower bounds based on the maximum metric. Finally, Constraints (4.16) through (4.19) define the domain of the auxiliary variables.

4.4.1.2. Slave problem

Solving the master model generates feasible item batches, which may, however, not be optimal. Let $\bar{\boldsymbol{z}}$ be a candidate integer solution of the master model, defining $r = \sum_{j \in \Omega} \bar{z}_{j,j}$ pick tours. Each pick tour $k = 1, \ldots, r$ consists of picking the items in set $\bar{\omega}_k = \{j \in \Omega \mid \bar{z}_{j,j'} = 1\}$, where $j' \in \Omega$ is the k-th item for which $\bar{z}_{j'j'} = 1$. We refer to the set of pick tours derived from the

current master solution as $\bar{\Phi} = \{\bar{\omega}_1, \ldots, \bar{\omega}_r\}$, where we use bars to indicate that the sets $\bar{\omega}_k$ and $\bar{\Phi}$ are fixed within the slave problem.

Given an item batching $\bar{\boldsymbol{z}}$, two problems remain to be solved: First, we have to decide on the location χ of the depot, and second, we need to determine the optimal picker route for each $\bar{\omega}_k$. As soon as the first problem is solved, the second becomes trivial. Hence, we begin with the former. Note that auxiliary variable $\bar{\chi}$ in the master model does not necessarily correspond to the optimal depot location because the distance values $\hat{d}_j$ are only lower bounds. Instead, the best depot location χ for a given set $\bar{\Phi}$ of tours can be found by minimizing $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ (cf., Equation (4.3)) for the given candidate integer solution $\bar{\boldsymbol{z}}$, which can be done as follows.

The term $\frac{\partial G(\bar{\omega}_1,\ldots,\bar{\omega}_r,\chi)}{\partial \chi}$ is not continuous because of the minimum term in each $g(\bar{\omega}_k, \chi)$ (cf., Equation (4.2)). However, for a fixed set $\bar{\Phi}$, it is piece-wise continuous in every interval where $\arg\min_{(j,j')\in\eta_k} \left\{\gamma_{j,j'}(\chi)\right\}$ is constant, $\forall k \in \{1, \ldots, r\}$. Hence, to minimize $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$, three steps are necessary. First, we need to determine every interval, in which $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ has a constant derivative. Second, for each interval, we need to determine the minimum of $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ separately. Third, out of all intervals' minima, we need to select the one that is minimal overall. The last step is trivial. In the following, we discuss the first two steps, where we label the minimal value for $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ as $G^*(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ and the respective depot location $\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$.

Starting from the second step, the minimum within an interval can be determined by finding the root of $\frac{\partial G(\bar{\omega}_1,\ldots,\bar{\omega}_r,\chi)}{\partial \chi}$ within that interval. The root within a continuous interval can be determined with arbitrary precision using a gradient descent method (e.g., Newton's method). Note that finding the root of the derivative within each interval is sufficient, since $\frac{\partial^2\left(-d_{j,j'}+\gamma_{j,j'}(\chi)\right)}{\partial \chi^2} = \tilde{y}_j^2 \cdot \left(\left(\tilde{x}_j - \chi\right)^2 + \tilde{y}_j^2\right)^{-\frac{3}{2}} + \tilde{y}_{j'}^2 \cdot \left(\left(\tilde{x}_{j'} - \chi\right)^2 + \tilde{y}_{j'}^2\right)^{-\frac{3}{2}} > 0$, hence, in a given interval, $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ is convex and only has a single minimum. For the same reason, if the root lies outside of the interval, the interval's minimum is equal to one of its boundaries.

It remains to discuss the first step, i.e., to determine the borders of the intervals in which $\arg\min_{(j,j')\in\eta_k} \left\{\gamma_{j,j'}(\chi)\right\}$ is constant. The edge $\arg\min_{(j,j')\in\eta_k} \left\{\gamma_{j,j'}(\chi)\right\}$, which we call *dominant* edge in the following, can only change at values χ where the functions $\gamma_{j,j'}(\chi)$ and $\gamma_{j'',j'''}(\chi)$ intersect, for two edges $(j, j'), (j'', j''') \in \eta_k : (j, j') \neq (j'', j''')$. This leads to the following iterative procedure. Initially, we determine $(\hat{j}, \hat{j}') = \arg\min_{(j,j')\in\eta_k} \left\{\gamma_{j,j'}(0)\right\}$ as the dominant edge for $\chi = 0$ and we initialize $\hat{\chi} := 0$. In each iteration, we determine the

consecutive dominant edge. The dominant edge $(\hat{j}, \hat{j}')$ changes, when

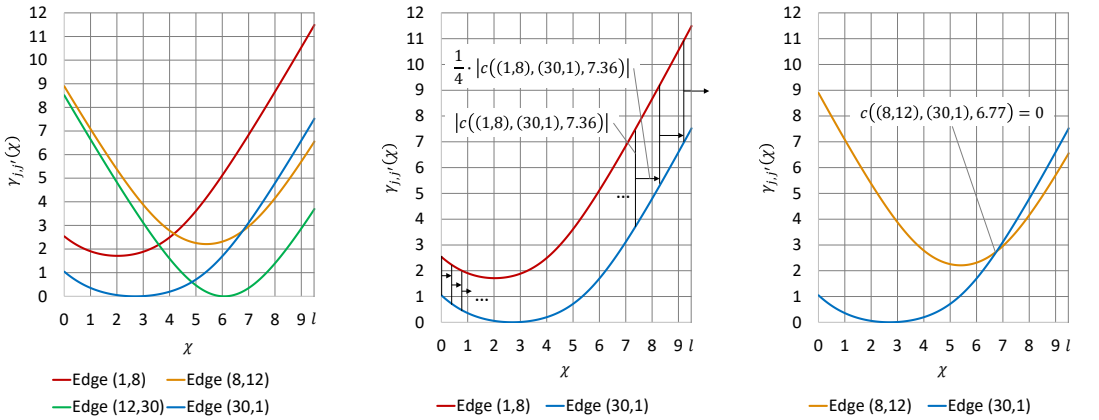
$$c((\hat{j}, \hat{j}'), (j, j'), \chi) = \gamma_{\hat{j},\hat{j}'}(\chi) - \gamma_{j,j'}(\chi)$$

assumes zero, approaching from a negative value with increasing χ, for any $(j, j') \in \eta_k : (j, j') \neq (\hat{j}, \hat{j}')$ and $l - \frac{w}{2} \geq \chi > \hat{\chi}$. Let χ' be the smallest value of χ for which $c((\hat{j}, \hat{j}'), (j, j'), \chi)$ assumes zero approaching from a negative value, $\forall (j, j') \in \eta_k : (j, j') \neq (\hat{j}, \hat{j}')$, and let (j'', j''') be the respective edge. Then we update $\hat{\chi} := \chi'$ and $(\hat{j}, \hat{j}') = (j'', j''')$ and start the next iteration. The values of $\hat{\chi}$ found during the iterations mark the intervals' borders. The procedure terminates as soon as no $\chi' \leq l - \frac{w}{2}$ can be found anymore.

During the iterative procedure, we solve $c((\hat{j}, \hat{j}'), (j, j'), \chi) = 0$ as follows. At first glance, we cannot rule out that function $c((\hat{j}, \hat{j}'), (j, j'), \chi)$ has none, one or multiple roots, such that we cannot use a gradient descent approach, since it may overshoot if there are multiple roots or not terminate if there are none. However, we can conclude that $\left|\frac{\partial c((j,j'),(j'',j'''),\chi)}{\partial \chi}\right| \leq 4$ because $0 \leq \left|\frac{\partial \sqrt{(\tilde{x}_j - \chi)^2 + \tilde{y}_j^2}}{\partial \chi}\right| = \left|-\frac{\tilde{x}_j - \chi}{\sqrt{(\tilde{x}_j - \chi)^2 + \tilde{y}_j^2}}\right| \leq 1, \forall j \in \Omega$ (cf. Equation (4.2)). This means that if χ changes by a step size of μ, $c((j, j'), (j'', j'''), \chi)$ changes by no more than $4 \cdot \mu$, which leads to the following iterative sub-routine. Starting from $\tilde{\chi} := \hat{\chi}$, we determine a step size $\mu := \max\left\{\frac{1}{4} \cdot |c((j, j'), (j'', j'''), \tilde{\chi})|, \mu^{min}\right\}$, where μ^{min} is the arbitrarily small numerical precision. We update $\tilde{\chi} := \tilde{\chi} + \mu$ and evaluate $c((j, j'), (j'', j'''), \tilde{\chi})$. We repeat this process until $c((j, j'), (j'', j'''), \tilde{\chi})$ changes from a negative value to a positive value between two iterations or until we reach $\tilde{\chi} > l - \frac{w}{2}$ and return $\tilde{\chi}$ as the solution. The complete procedure to solve the slave problem is summarized in pseudo-code in Algorithm 4.1.

Example. For additional clarification, in the following, we exemplary demonstrate how we determine the borders of the intervals in which $g(\bar{\omega}_k, \chi)$ has a continuous derivative. Consider a U-zone as depicted in Figure 4.1a, where the items $j \in \omega_k = \{1, 8, 12, 30\}$ should be picked on the same tour. To keep the example simple, we assume every item is stored in stillage i with the same respective index and set $w = 1.3$ m and $s = 0.05$ m (cf., Section 4.5). The set of edges is then given as $\eta_k = \{(1, 8), (8, 12), (12, 30), (30, 1)\}$. Figure 4.3a depicts the functions $\gamma_{j,j'}(\chi)$, $\forall (j, j') \in \eta_k$.

The procedure starts by determining the dominant edge at $\chi = 0$, which is $(\hat{j}, \hat{j}') = \arg\min_{(j,j') \in \eta_k} \left\{\gamma_{j,j'}(0)\right\} = (30, 1)$. Starting with edge $(1, 8)$ and with $\chi = 0$, we iteratively increase χ to determine a possible intersection between $\gamma_{30,1}(\chi)$ and $\gamma_{1,8}(\chi)$, which is



(a) $\gamma_{j,j'}(\chi), \forall (j,j') \in \eta_k$.

(b) Iterative determination of intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{1,8}(\chi)$.

(c) Intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{8,12}(\chi)$.

Figure 4.3.: Exemplary determination of intervals where $g(\omega_k, \chi)$ has a continuous derivative.

schematically shown in Figure 4.3b. Since there is no intersection, for $\chi \leq l - \frac{w}{2}$, we continue with the next edge, i.e., $(8, 12)$. Here, we find an intersection at $\chi = 6.77$ (cf., Figure 4.3c). We save $\hat{\chi} := 6.77$ and select the next edge $(12, 30)$ to look for intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{12,30}(\chi)$. We find an intersection at $\chi = 4.85$ and overwrite $\hat{\chi} := 4.85$. At this point, we have checked for intersections between $\gamma_{30,1}(\chi)$ and $\gamma_{j,j'}(\chi)$ for all other edges $(j, j') \in \eta_k : (j, j') \neq (30, 1)$ and found the intersection closest to 0 at $\hat{\chi} := 4.85$. Hence, the fist interval where $g(\bar{\omega}_k, \chi)$ has a constant derivative is $[0, 4.85)$. To determine the next interval, we set the dominant edge to $(\hat{j}, \hat{j}') = (12, 30)$ and try to find intersections for $\chi > 4.85$ in the same way as before. Since there are no more intersections, we determine the second (and in this case final) interval as $[4.85, l - \frac{w}{2}]$.

Determining these intervals not only for $\bar{\omega}_k$ but for all $\bar{\omega}_1, \ldots, \bar{\omega}_r$ gives the intervals in which $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ has a continuous derivative. To solve the slave problem, the final step is then to determine the root of $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi)$ within every such interval and to select the one that is minimal overall.

Once the optimal depot location $\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ has been determined, the optimal route for each set $\bar{\omega}_k$, $\forall k = \{1, \ldots, r\}$, follows immediately from the continuous interval in which $\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ lies. We define $(j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r), j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)) = \arg\min_{(j,j') \in \eta_k} \{\gamma_{j,j'}(\chi^*(\bar{\omega}_1, \ldots, \bar{\omega}_r))\}$. Then the optimal route for the set $\bar{\omega}_k$ visits items $j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ and $j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)$ directly before returning to and after leaving from the depot, respectively. All other items

$j \in \bar{\omega}_k \setminus \{j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r), j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r)\}$ are visited in clockwise order in-between (cf., Proposition 4.1).

4.4.1.3. Combinatorial cuts

Let UB be the objective value of the current best known solution (i.e., the best currently known upper bound on the optimal objective value). If no feasible solution is known yet, let $UB = \infty$. If $G^* < UB$, a new best solution has been found, which is stored, and UB is updated to G^*. If UB is updated, we add the following cut, which we call an *optimality cut*, to the constraint set of the master model:

$$G \leq G^*(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi^*(\bar{z})) - \epsilon, \tag{4.20}$$

where ϵ is a sufficiently small positive number and G is the objective function of the master model. This equation cuts off all solutions whose lower bound is not better than the current upper bound.

Regardless of whether a new upper bound has been found, we make use of the following observation to generate further cuts. To find a better solution, the duration of at least one of the r tours must be made shorter, or one of the tours must be dissolved altogether. A tour can only become shorter if the polygon constituting the tour changes its form. The polygon can only change its form if a stillage lying at a vertex changes, but the polygon stays unchanged if a stillage lying on (the middle of) an edge changes. There are two general possibilities where the former is the case.

The first possibility is that either the first or the last stillage to be visited per face of the U-shaped picking area changes. Adding or removing stillages that lie in-between two other stillages on the same face (and that are not visited directly before or after the depot) cannot reduce the duration of the tour because the picker passes by that stillage in any case. The second possibility is that a stillage changes that is visited either directly after leaving or before returning to the depot. For additional clarification, Figure 4.4 provides an example.

Let $T = \{j \in \Omega \mid \tilde{y}_j = \max_{i \in I} \{y_i\}\}$ be the set of items on the top face of the U-shaped picking area. Analogously, let B be the set of items on the bottom face, and R be the set of items on the perpendicular face. For notational convenience, we define the set $\underline{T}_k = \{j \in T \cap \omega_k \mid \tilde{x}_j = \min_{j \in T \cap \omega_k} \{\tilde{x}_j\}\}$ and $\overline{T}_k = \{j \in T \cap \omega_k \mid \tilde{x}_j = \max_{j \in T \cap \omega_k} \{\tilde{x}_j\}\}$ as well as, analogously, $\underline{B}_k$, $\overline{B}_k$, $\underline{R}_k$ and $\overline{R}_k$ as the extreme items on each face. Then the set of all items that could be

Algorithm 4.1: Algorithm for solving the slave problem.

Input: $\bar{\Phi} = \{\bar{\omega}_1, \ldots, \bar{\omega}_r\}$

$B := \left\{l - \frac{w}{2}\right\}$; // borders of the intervals, in which $\frac{\partial G(\bar{\omega}_1,\ldots,\bar{\omega}_r,\chi)}{\partial \chi}$ is continuous

for $k \in \{1, \ldots, r\}$ **do**

  $b := \{0\}$; // borders of the intervals, in which $\frac{\partial g(\bar{\omega},\chi)}{\partial \chi}$ is continuous

  $\hat{\chi}^{\text{previous}} := 0$; // previous interval border

  $(\hat{j}, \hat{j}') = \arg\min_{(j,j')\in\eta_k} \{\gamma_{j,j'}(0)\}$; // dominant edge

  $\hat{\chi} := 0$; // interval border

  while $\hat{\chi} < l - \frac{w}{2}$ **do**

    $\hat{\chi} := l - \frac{w}{2}$;

    foreach $(j, j') \in \eta_k : \gamma_{\hat{j},\hat{j}'}(\chi) \neq \gamma_{j,j'}(\chi)$ **do**

      $\chi := \max\{b\}$;

      while $c((\hat{j}, \hat{j}'), (j, j'), \chi) < 0 \wedge \chi < l$ **do**

        $\chi := \chi + \max\left\{\frac{1}{4} \cdot |c((j, j'), (j'', j'''), \tilde{\chi})|, \mu^{min}\right\}$;

      if $\chi < \hat{\chi}$ **then**

        $\hat{\chi} := \chi$;

        $(\hat{j}, \hat{j}') = (j, j')$

    $b := b \cup \{\hat{\chi}\}$

  $B := B \cup b$;

$\chi^* := 0$; // optimal depot position for the given $\bar{\Phi}$

$G^* := \infty$; // optimal objective for the given $\bar{\Phi}$

foreach *pair of numerically consecutive elements* β_1 *and* β_2 *(with* $\beta_1 < \beta_2$*) in* B **do**

  $\chi :=$ the solution of $\frac{\partial G(\bar{\omega}_1,\ldots,\bar{\omega}_r,\chi)}{\partial \chi} = 0$ using Newton's method;

  $\chi := \min\{\chi, \beta_2\}$;

  $\chi := \max\{\chi, \beta_1\}$;

  if $G(\bar{\omega}_1, \ldots, \bar{\omega}_r, \chi) < G^*$ **then**

    $G* := G(\chi)$;

    $\chi^* := \chi$;

Output: G^* and χ^*

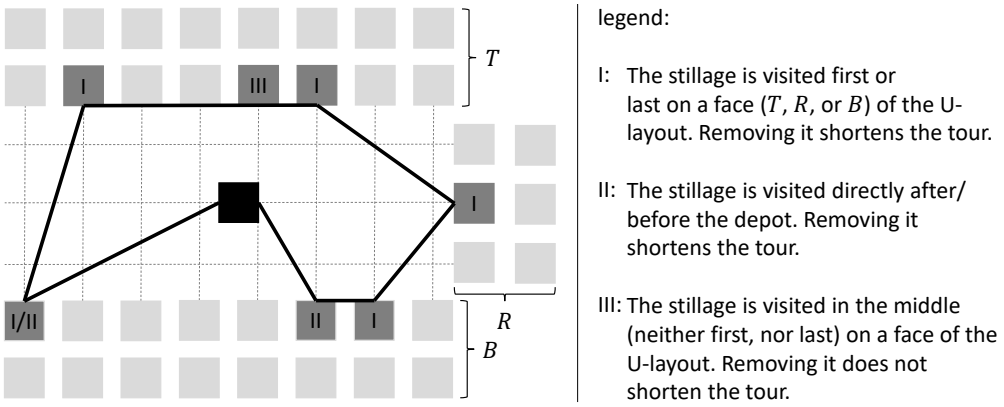


Figure 4.4.: Example of which stillages are considered in the *progression cut*.

changed to alter the route of tour k is given as $\Lambda_k = \underline{T}_k \cup \overline{T}_k \cup \underline{B}_k \cup \overline{B}_k \cup \underline{R}_k \cup \overline{R}_k \cup \left\{ j_k^*(\bar{\omega}_1, \ldots, \bar{\omega}_r), j_k'^*(\bar{\omega}_1, \ldots, \bar{\omega}_r) \right\}$.

Using these definitions, we add the following cut, which we call a *progression cut*, to the master model:

$$1 \leq \sum_{k=1}^{r} \sum_{j \in \Lambda_k} \left(1 - z_{j,\max\{\bar{\omega}_k\}}\right) \tag{4.21}$$

Inequality (4.21) enforces that at least one stillage at a polygon's vertex of one of the r tours is reassigned. Note that this always makes the current solution infeasible. The solver thus continues solving the master model with the newly added cut(s), intermittently calling the slave problem whenever a new candidate integer solution is found until the search space is empty.

4.4.2. Heuristic solution approaches

While the proposed CBD is able to solve problem instances of smaller sizes in acceptable runtime, its runtime gets excessively long when working on larger sized instances (cf., Section 4.5.2). Therefore, we also consider heuristic solution approaches in the following.

Glock and Grosse (2012) propose a heuristic sweep algorithm (SA) for the picker routing problem in U-shaped pick areas. In our computational experiments (cf., Section 4.5), the SA produces good results on average. However, for some instances, the optimality gaps can be substantial. Inspired by these findings, we propose a heuristic solution procedure using the concept of dynamic programming (DP) that expands on the idea of the SA. In the following, we briefly

review the SA of Glock and Grosse (2012) before discussing our DP approach.

4.4.2.1. Sweep algorithm

The SA starts by assuming a fixed depot position $\bar{\chi}$. Its basic idea is to assign items to sets ω_k in clockwise order. Starting from an initial item, the SA assigns items to the same set ω_k, until the next item would exceed the picker's capacity Q. In that case, the next item is assigned to a new set ω_{k+1} and the previous set is closed. For each closed set of items ω_k, $g(\omega_k, \bar{\chi})$ is derived (cf. Equation 4.2). Since $\bar{\chi}$ is given, $g(\omega_k, \bar{\chi})$ can be easily calculated. Once all items have been assigned to sets, the objective value is given by $G(\omega_1, \ldots, \omega_r, \bar{\chi})$ (cf. Equation 4.3). The fixed depot position $\bar{\chi}$ and the starting item are varied over multiple iterations of the algorithm. Using $\chi^{\text{step-size}}$ as an arbitrarily small step-size to increment $\bar{\chi}$, the algorithm can be formally described as follows.

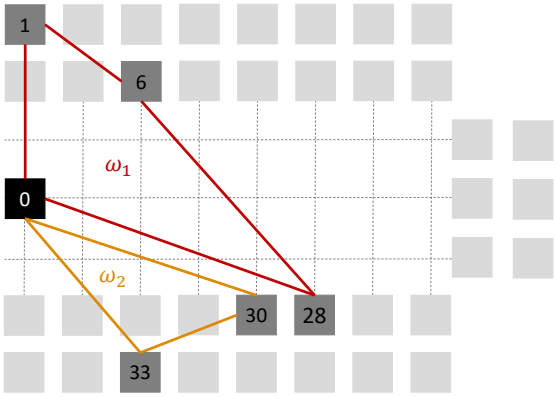
1. Set $\bar{\chi} := 0$, $G^* = \infty$ and $\chi^* = 0$.
2. Set $j^{\text{start}} := 1$.
3. Set $j := j^{\text{start}}$ and $k := 1$.
4. Set $\omega_k = \{j\}$.
5. Increment $j := j + 1$. If $j > |\Omega|$ set $j := 1$, i.e., after the U-zone's final item has been reached, proceed with the first item to continue the clockwise sweeping. If $j = j^{\text{start}}$, i.e., if item j has already been considered in the beginning, go to Step 7.
6. If $\sum_{j' \in \omega_k} q_{j'} + q_j \leq Q$, add $\omega_k := \omega_k \cup \{j\}$ and go to Step 5. Else, increment $k := k + 1$ and go to Step 4.
7. Calculate $G(\omega_1, \ldots, \omega_k, \bar{\chi})$. If $G(\omega_1, \ldots, \omega_r, \bar{\chi}) < G^*$, update $G^* := G(\omega_1, \ldots, \omega_r, \bar{\chi})$ and $\chi^* := \bar{\chi}$. Increment $j^{\text{start}} := j^{\text{start}} + 1$. If $j^{\text{start}} \in \Omega$, go to Step 3.
8. Increment $\bar{\chi} := \bar{\chi} + \chi^{\text{step-size}}$. If $\bar{\chi} \leq l - \frac{w}{2}$, go to Step 2. Else, terminate the procedure.

Example. Consider a U-zone as depicted in Figure 4.1a, where the items $j \in \{1, 6, 28, 30, 33\}$ should be picked. To keep the example simple, we assume every item is stored in stillage i with the same respective index and set $w = 1.3$ m and $s = 0.05$ m (cf., Section 4.5). The weights of the items are given in Figure 4.5a and the picker has a capacity of $Q = 5$. For the given initial

item $j^{\text{start}} = 1$ and the depot location $\bar{\chi} := 0$, the solution as determined by the SA is given in Figure 4.5b with an objective value of 29.43.

j	q_j
1	1
6	2
28	2
30	1
33	2

(a) Example item weights.



(b) Example solved by the the SA.

Figure 4.5.: Example solved by the the SA.

4.4.2.2. Heuristic dynamic programming algorithm

In the SA, items are always added to the currently "open" set ω_k until adding another item would exceed the capacity (cf., Step 6). However, we notice in our computational experiments that this is not always a good choice (see Section 4.5). Instead, sometimes it is better to "close" a set ω_k before the capacity is reached, and add the next item to the consecutive set ω_{k+1}. The following DP approach takes this observation into consideration.

Similar to the SA, the DP procedure assumes a fixed depot position $\bar{\chi}$ and an initial start item j^{start} at the beginning of each iteration and considers items in a clockwise order. The solution is then constructed piece-wise using a DP scheme, based on the general idea formulated by Bellman (1954).

For given values of $\bar{\chi}$ and j^{start}, the DP consists of $|\Omega|+1$ stages $p = 0, \ldots, |\Omega|$, each containing one state $\Theta_p = \{j \in J : j^{\text{start}} \leq j < j^{\text{start}} + p \vee 1 \leq j < p + j^{\text{start}} - |\Omega|\}$, denoting the set of items that have already been considered in the partial solution. Starting from initial stage $p = 0$ with state $\Theta_p = \emptyset$, a successor stage $p' > p$ is reached by adding the set $\Theta_{p'} \setminus \Theta_p$ to Θ_p, indicating that items $j \in \Theta_{p'} \setminus \Theta_p$ are picked in the same tour $\omega_{p'}$. A transition is only feasible if $\sum_{j \in \Theta_{p'} \setminus \Theta_p} q_j \leq Q$, i.e., if the capacity is not exceeded.

Let $V(\Theta_p)$ be the set of states from which a feasible transition to state Θ_p exists. The optimal

objective value $h^*(\Theta_p)$ of the partial solution in state Θ_p can then be calculated recursively as

$$h^*(\Theta_p) = \min_{\Theta' \in V(\Theta_p)} \left\{ h^*(\Theta') + g\left(\Theta_p \setminus \Theta', \bar{\chi}\right) \right\},$$

with $h^*(\emptyset) = 0$. The objective value of a complete solution in final state $\Theta_{|\Omega|}$ is also the best objective value for the given values of $\bar{\chi}$ and j^{start}. We can obtain the corresponding assignment by backward recovery along the best path. To complete the proposed DP, we increment $\bar{\chi}$ and j^{start} in the same manner as for the SA and save the overall best obtained solution.

The way the DP is set up, it is guaranteed to always find a solution that is at least as good as the sweep algorithm's solution. Concerning the time complexity, for given values of $\bar{\chi}$ and j^{start}, there are $O(n^2)$ transitions. Note that, by careful implementation, it is possible to calculate the objective contribution of all $O(n)$ successors of a state in $O(n)$ time. Let $\sigma = \frac{l - \frac{w}{2}}{\chi^{\text{step-size}}}$ be the number of increments considered for $\bar{\chi}$. Then the asymptotic runtime of the complete procedure (including varying $\bar{\chi}$ and j^{start}) is bounded by $O(\sigma \cdot n^3)$.

Example. Consider the same example as for the SA in Section 4.4.2.1. For the given initial item $j^{\text{start}} = 1$ and the depot location $\bar{\chi} := 0$, Figure 4.6 depicts the dynamic programming graph including the optimal path recovered from backtracking. The objective of the DP's solution is 22.63, which is 23.10% below the one of the SA's solution.

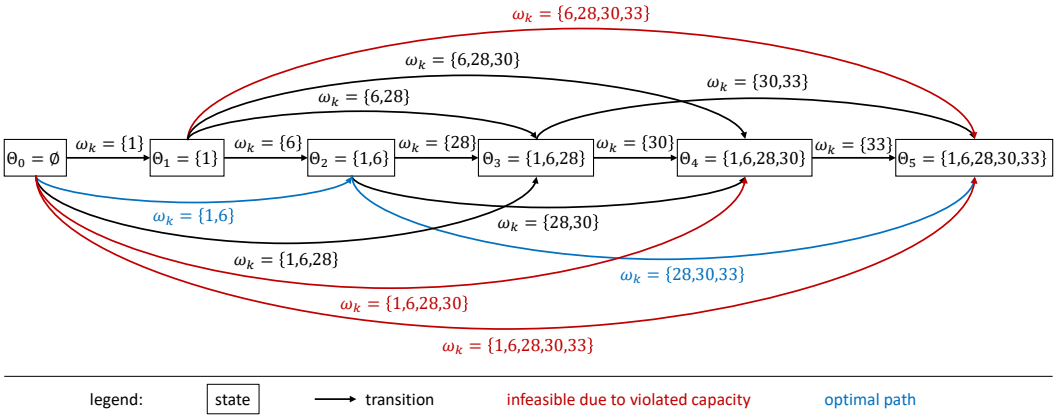


Figure 4.6.: Dynamic programming graph for the example of Section 4.4.2.1.

4.5. Numerical experiments and analysis

4.5.1. Generating instances for the computational tests

To evaluate our proposed solution procedures and gain some managerial insights, we generate problem instances based on our observations in practice and the assumptions presented by Glock and Grosse (2012). The following section describes how the instances are generated.

We consider U-layouts with two different capacities, either 44 stillages (cf., Glock and Grosse, 2012) or 88 stillages. The layout of an instance is defined by setting n and m, the number of stillages in one horizontal and in one vertical row. Since both Glock and Grosse (2012) and Diefenbach and Glock (2019) found that narrow U-shapes are advantageous, we consider the layouts of $(n, m) = (10, 2)$, $(n, m) = (9, 4)$, and $(n, m) = (8, 6)$ if the U contains 44 stillages, and the layouts of $(n, m) = (21, 2)$, $(n, m) = (20, 4)$, and $(n, m) = (19, 6)$ if the U contains 88 stillages. In accordance with Glock and Grosse (2012), we set the measurements of the stillages to $w = 1.3$ m and the gap between the stillages to $s = 0.05$ m.

For the item demands, we assume a $20/60$-Pareto distribution (cf., Bender, 1981), i.e., 20% of the items are responsible for 60% of demand. Based on the distribution, we randomly draw $|\Omega|$ items to be picked for each instance. For the instances with 44 stillages, we set either $|\Omega| = 10$ or $|\Omega| = 15$; for the instances with 88 stillages, we set either $|\Omega| = 30$ or $|\Omega| = 60$. Items weights q_j are drawn randomly from the set $\{1, \ldots, 5\}$ and the picker capacity is set to $Q = 15$ (cf., Glock and Grosse, 2012). Finally, we assign the items to stillages in a random manner, and set $v = 3$ (cf., Glock and Grosse, 2012) as the default value for the penalty to move the depot. Note that we will deviate from the default settings for v and the default item assignment in later experiments.

We generate ten instances for each setting. All instances are labeled as follows: "number of stillages"-(n, m)-$|\Omega|$-*running index*. All instances are available for download at `https://doi.org/10.5281/zenodo.4671870`.

4.5.2. Computational performance

This section investigates the performance of the three proposed solution procedures, namely CBD, SA, and DP. All testing is performed on an Intel Core i7-3631QM CPU @ 2.20 GHz and with 8 GB of RAM. All algorithms are implemented in C#, and CPLEX (version 12.10) is used

as a default solver for the master model of the proposed CBD at default settings. The maximum runtime is set to 3600 s (1 h). The numerical precision is set to $\mu = \epsilon = \chi^{\text{step-size}} = 0.01$ for all procedures and respective parameters.

In a preliminary test we found that the influence of the exact layout (i.e., the values of n and m for a given number of stillages) on the algorithmic performance is negligible. We therefore restrict the performance tests to the instances with $m = 4$. The results of the performance test are summarized in Table 4.2.

The computational experiments show that the exact CBD is able to solve all instances with $|\Omega| \leq 15$ in acceptable runtime. However, not surprisingly given the NP-hard, non-linear nature of the problem, for instances with $|\Omega| \geq 30$ the procedure does not terminate within the maximum runtime of 3600 s and the gaps to the lower bounds of the MP (as reported by CPLEX) remain high at the point of termination. Furthermore, both heuristic approaches yield better solutions for $|\Omega| \geq 30$. Especially for larger instances, the time share spent solving the MP (which is above 97% for $|\Omega| \geq 30$) indicates that the MP is the procedure's bottleneck; the SP appears to be sufficiently fast to solve.

The SA runs very fast with a runtime of below 1 s even for $|\Omega| = 60$. For $|\Omega| \geq 15$, where we know the optimum thanks to CBD, the optimality gaps are generally low. For most instances, the optimality gaps are even zero. However, for five instances they are above 2% and can be as high as about 5%. Nonetheless, we can draw the conclusion that the relatively simple sweep algorithm works quite well for U-shaped picking areas. This is certainly good news for practitioners, who may find it easy to implement such a routing scheme.

To further improve the SA, our approach based on dynamic programming yields even better solutions. It completely closes the optimality gap in all small and most medium size instances. Even though the DP is a little slower than the SA, it sill runs fast, with a runtime of at most 10 s for $|\Omega| = 60$, which is clearly sufficient for practical application.

4.5.3. Effects of having a movable depot

In our default settings, we assume a movable depot that can be positioned at any position $0 \leq \chi \leq l$ and set $v = 3$, which means that moving the depot is six times faster than travel during order picking (cf., Section 4.3.1). However, in practice, companies may find it easier to define a fixed position for the depot. Furthermore, the depot could be set up and moved in a different way. For example, the depot could be moved by a forklift or a manual hand lift truck. Depending

Instances	Benders Decomposition								Sweep Algorithm				Dynamic Programming			
label	value	LB CPLEX	chi	runtime (in s)	time MP (in %)	time SP (in %)	opt cuts	prog cuts	value	chi	gap (in %)	runtime (in s)	value	chi	gap (in %)	runtime (in s)
44-(9x4)-10-01	34.34	34.34	3.87	0.58	75.22	24.78	3	34	34.34	3.87	0.00	0.01	34.34	3.87	0.00	0.76
44-(9x4)-10-02	37.40	37.40	9.13	0.30	80.20	19.80	10	23	37.40	9.13	0.00	0.01	37.40	9.13	0.00	0.29
44-(9x4)-10-03	44.55	44.55	10.83	1.40	60.73	39.27	12	105	45.50	10.10	2.13	0.01	44.55	10.83	0.00	0.19
44-(9x4)-10-04	37.26	37.26	8.75	0.45	90.22	9.78	5	18	37.26	8.75	0.00	0.01	37.26	8.75	0.00	0.22
44-(9x4)-10-05	32.72	32.72	1.74	0.30	79.87	20.13	6	25	32.72	1.74	0.00	0.01	32.72	1.74	0.00	0.66
44-(9x4)-10-06	38.99	38.99	3.07	0.68	57.46	42.54	7	66	38.99	3.07	0.00	0.01	38.99	3.07	0.00	0.58
44-(9x4)-10-07	43.11	43.11	5.47	1.10	58.00	42.00	11	130	43.99	6.54	2.04	0.01	43.11	5.47	0.00	0.35
44-(9x4)-10-08	37.45	37.45	9.80	0.58	75.56	24.44	6	55	37.91	9.56	1.23	0.01	37.45	9.80	0.00	0.42
44-(9x4)-10-09	34.40	34.40	11.69	0.42	81.67	18.33	5	24	35.63	11.55	3.58	0.01	34.40	11.69	0.00	0.28
44-(9x4)-10-10	34.35	34.35	8.19	0.72	61.42	38.58	8	79	34.35	8.19	0.00	0.01	34.35	8.19	0.00	0.19
mean	**37.46**	**37.46**	**7.25**	**0.65**	**72.04**	**27.97**	**7.3**	**55.9**	**37.81**	**7.25**	**0.90**	**0.01**	**37.46**	**7.25**	**0.00**	**0.39**
44-(9x4)-15-01	52.42	52.42	9.67	39.27	83.84	16.16	12	1239	52.42	9.67	0.00	0.02	52.42	9.67	0.00	0.64
44-(9x4)-15-02	57.43	57.43	5.12	54.75	85.42	14.58	19	1163	57.43	5.12	0.00	0.03	57.43	5.12	0.00	0.35
44-(9x4)-15-03	53.81	53.81	6.84	99.48	87.21	12.79	12	2096	56.41	6.98	4.83	0.02	56.41	6.98	4.83	0.57
44-(9x4)-15-04	47.42	47.42	5.75	12.66	84.28	15.72	9	381	47.42	5.75	0.00	0.02	47.42	5.75	0.00	0.88
44-(9x4)-15-05	42.47	42.47	3.85	8.07	72.59	27.41	7	472	42.47	3.85	0.00	0.02	42.47	3.85	0.00	1.38
44-(9x4)-15-06	49.35	49.35	6.00	19.14	94.25	5.75	10	220	49.35	6.00	0.00	0.03	49.35	6.00	0.00	0.44
44-(9x4)-15-07	43.67	43.67	3.75	13.14	82.69	17.31	10	412	43.67	3.75	0.00	0.02	43.67	3.75	0.00	1.03
44-(9x4)-15-08	46.98	46.98	6.59	7.54	95.08	4.92	7	78	46.98	6.59	0.00	0.02	46.98	6.59	0.00	0.58
44-(9x4)-15-09	53.01	53.01	7.96	79.91	91.13	8.87	15	1080	53.01	7.96	0.00	0.02	53.01	7.96	0.00	0.40
44-(9x4)-15-10	45.75	45.75	10.06	16.82	69.38	30.62	9	829	46.85	10.03	2.40	0.02	45.75	10.06	0.00	0.99
mean	**49.23**	**49.23**	**6.56**	**35.08**	**84.59**	**15.41**	**11.0**	**797.0**	**49.60**	**6.57**	**0.72**	**0.02**	**49.49**	**6.57**	**0.48**	**0.73**
88-(20x4)-30-01	148.87	84.97	16.23	–	96.79	3.21	19	2732	145.65	16.32	-2.16	0.17	144.03	16.31	-3.25	4.73
88-(20x4)-30-02	148.01	84.01	17.24	–	97.67	2.33	21	1750	140.87	17.42	-4.82	0.20	140.87	17.42	-4.82	2.94
88-(20x4)-30-03	146.74	82.44	14.07	–	98.27	1.73	14	1703	139.27	14.10	-5.09	0.16	139.27	14.10	-5.09	3.54
88-(20x4)-30-04	177.22	93.24	16.84	–	97.17	2.83	28	214	167.76	15.78	-5.34	0.22	166.42	16.58	-6.09	2.18
88-(20x4)-30-05	145.45	79.07	11.90	–	98.21	1.79	20	1440	142.00	10.90	-2.37	0.20	138.11	11.80	-5.05	2.49
88-(20x4)-30-06	171.98	86.04	12.46	–	98.40	1.60	15	1261	159.75	12.94	-7.11	0.20	159.75	12.94	-7.11	2.70
88-(20x4)-30-07	143.08	81.31	12.78	–	98.10	1.90	32	1699	134.48	12.97	-6.01	0.17	134.48	12.97	-6.01	6.72
88-(20x4)-30-08	151.92	97.99	13.44	–	99.23	0.77	9	665	147.22	13.53	-3.09	0.17	147.22	13.53	-3.09	4.02
88-(20x4)-30-09	154.80	93.21	15.48	–	97.85	2.15	26	1704	148.02	14.35	-4.38	0.19	148.02	14.35	-4.38	2.83
88-(20x4)-30-10	144.04	77.70	13.81	–	99.42	0.58	18	457	137.94	13.54	-4.23	0.19	137.94	13.54	-4.23	3.52
mean	**153.21**	**86.00**	**14.43**	**–**	**98.11**	**1.89**	**20.2**	**1555.8**	**146.30**	**14.19**	**-4.46**	**0.19**	**145.61**	**14.35**	**-4.91**	**3.57**
88-(20x4)-60-01	297.23	82.28	14.86	–	98.31	1.69	11	388	255.57	15.84	-14.02	0.74	255.55	15.83	-14.02	6.24
88-(20x4)-60-02	305.78	74.83	15.12	–	99.33	0.67	9	132	237.18	15.20	-22.43	0.72	234.57	15.21	-23.29	8.32
88-(20x4)-60-03	288.54	69.71	10.56	–	96.20	3.80	14	345	234.99	11.67	-18.56	0.72	231.58	11.97	-19.74	10.43
88-(20x4)-60-04	376.85	89.00	12.97	–	96.56	3.44	16	645	285.32	15.20	-24.29	0.76	283.06	14.81	-24.89	6.31
88-(20x4)-60-05	309.40	83.28	9.66	–	93.53	6.47	14	1124	239.07	12.25	-22.73	0.68	237.95	12.65	-23.09	6.20
88-(20x4)-60-06	341.57	94.20	13.41	–	99.71	0.29	17	67	266.08	15.12	-22.10	0.76	263.65	15.58	-22.81	5.77
88-(20x4)-60-07	339.56	98.19	12.69	–	98.29	1.71	10	106	270.38	14.15	-20.37	0.75	268.30	14.28	-20.99	5.05
88-(20x4)-60-08	334.89	94.83	13.41	–	97.60	2.40	15	533	275.25	15.14	-17.81	0.84	274.08	14.86	-18.16	5.17
88-(20x4)-60-09	310.37	79.73	12.35	–	98.15	1.85	16	414	240.26	13.27	-22.59	0.78	239.94	13.24	-22.69	5.98
88-(20x4)-60-10	293.21	77.76	13.19	–	98.73	1.27	25	236	252.90	13.95	-13.75	0.74	251.66	13.69	-14.17	9.40
mean	**319.74**	**84.38**	**12.82**	**–**	**97.64**	**2.36**	**14.7**	**399.0**	**255.70**	**14.18**	**-19.86**	**0.75**	**254.03**	**14.21**	**-20.39**	**6.89**

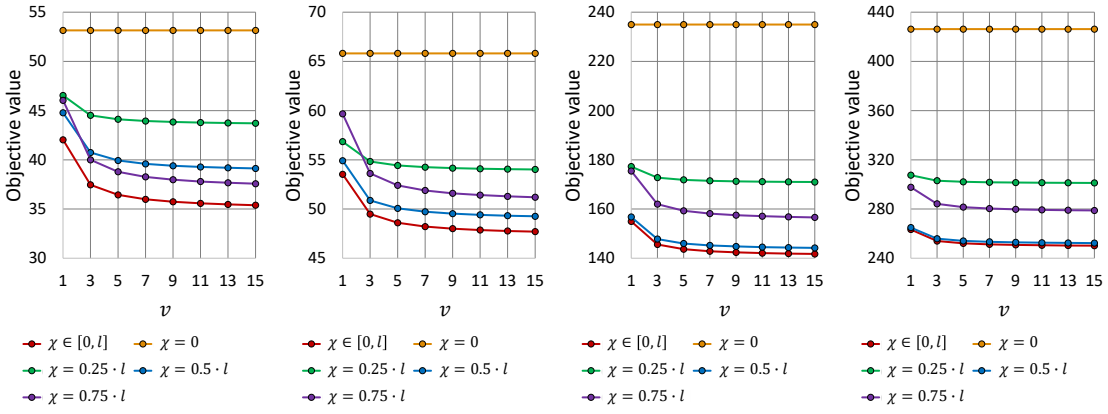
value = objective value; LB CPLEX = lower bound as reportet by CPLEX at the point of termination; chi = depot position of the best solution; runtime = runtime until termination; time MP =share of the runtime spent solving the master problem; time SP share of the runtime spent solving the slave problem; opt cuts = number of added optimality cuts; prog cuts = number of added progression cuts; gap = relative gap to the Benders Decomposition's best upper bound
– = the procedure was terminated because the runtime limit of 3600 s had been reached

Table 4.2.: Computational performance tests.

on this, the factor v may vary.

This section investigates the effects of having a movable or fixed depot as well as the effects of different values for v. For the experiment, we use the same instances as in the performance evaluation (cf., Table 4.2). We solve these instances using the DP approach assuming either a movable or a fixed depot. We test four different fixed depot positions at $\chi \in \{0, 0.25 \cdot l, 0.5 \cdot l, 0.75 \cdot l\}$. Furthermore, we test eight different values $v \in \{1, 3, 5, 7, 9, 11, 13, 15\}$. Figure 4.7. shows the mean results summarized for each instance size.

Obviously, having a stationary depot can never be better than having a movable depot, since the movable depot can always assume the position of the stationary depot. However, the benefits of having a movable depot strongly depend on the size of the U-zone and the length of the picklist, as the results in Figure 4.7 show. For larger U-zones and picklists, a stationary depot at $\chi = 0.5 \cdot l$ is almost as efficient as a movable one. For smaller U-zones and picklists, the benefits of the movable depot are more significant. For the 44-(n, m)-10-instances, the gap in the objective value between the movable depot case and the best fixed depot case was on average (over all



(a) 44-(n, m)-10-instances. (b) 44-(n, m)-15-instances. (c) 88-(n, m)-30-instances. (d) 88-(n, m)-60-instances.

Figure 4.7.: Best found objective values for different values of v.

different values of v) 6.37 %. For the 44-(n, m)-15-instances, the 88-(n, m)-30-instances, and 88-(n, m)-60-instances, the gaps were 3.05 %, 1.60 %, and 0.78 %, respectively.

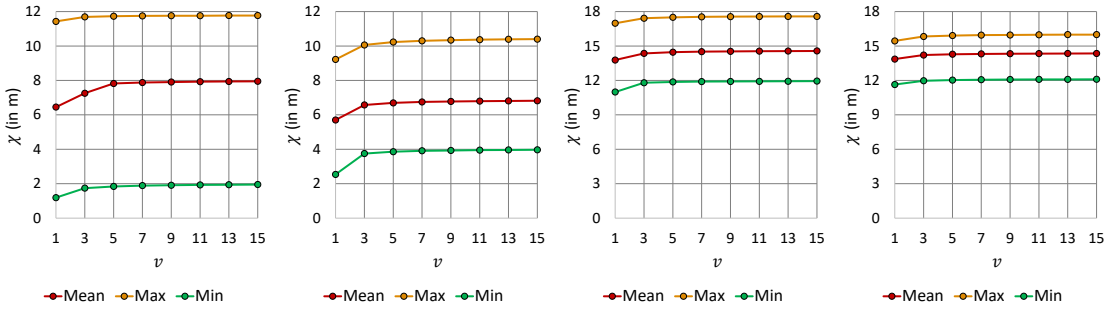
Given these finding, a practically relevant approach might be to limit possible depot positions to a few discrete locations. As indicated in Section 4.4.1, this would make PRP-UA non-exact and turn it into a type of capacitated vehicle routing problem. While the problem remains NP-hard, the capacitated vehicle routing problem is well researched with various exact and heuristic solution procedures being readily available (cf., Ralphs et al., 2003). Such an approach is beyond the scope of this paper, but may be interesting to investigate for future research.

Overall, $\chi = 0.5 \cdot l$, i.e., placing the depot in the middle of the U, performs best if a stationary position is enforced; only for the 44-(n, m)-10-instances, $\chi = 0.75 \cdot l$ is better. Fixing the depot at the entrance of the U-zone, i.e., at $\chi = 0$, is the worst option according to our experiments and results in objective values that are significantly higher than for a movable depot or any of the other fixed positions.

Except for the case where the depot is fixed at $\chi = 0$, the objective values decrease with increasing values of v. They do, however, not decrease linearly but asymptotically approach a constant value due to the fact that v influences the objective anti-proportionally. Hence, there are diminishing returns for increasing v.

For the movable depot, not only does v affect the objective value, but also the optimized depot position. Figure 4.8 shows how the mean, maximum and minimum depot positions vary with the

value of v. Comparing the sub-figures of the different instance sizes shows that the ideal depot position varies less for bigger U-zones and larger picklists (i.e., the maximum and minimum positions are closer to each other). The effect of v on the spread is negligible. Furthermore, with increasing values of v, the ideal depot position increases, asymptotically approaching a constant value. Again, this can be explained by the anti-proportional effect of v in the objective function. For increasing values of v, the effort to move the depot approaches zero, such that χ approaches the value that would be ideal to solely minimize the tour length during order picking.



(a) 44-(n, m)-10-instances.

(b) 44-(n, m)-15-instances.

(c) 88-(n, m)-30-instances.

(d) 88-(n, m)-60-instances.

Figure 4.8.: Best found depot positions for different values of v.

4.5.4. Effects of storage assignment policies

So far, we have only considered instances with a random assignment of items to stillages. However, the allocation of items to stillages significantly influences the performance of a warehouse with regard to the order picking process. Furthermore, in practice, the (time-)effort associated with exchanging stillages on different levels (upper and lower) needs to be considered in addition to travel distances. Picking items from stillages requires replenishments of empty stillages over time. In the industry case described in Glock and Grosse (2012), stillages are stacked one atop the other, and exchanging stillages in the upper row is less time-consuming than exchanging stillages on the lower level.

Typically, this problem is addressed with simple rules-of-thumb in both practice and research (cf., Section 4.2). Among various storage assignment methods that have been discussed in the literature, Glock and Grosse (2012) use a dedicated storage assignment policy where each item has a dedicated location in the U-zone, which is kept constant for a set of orders. In this paper, we develop a new assignment policy that we term “radial assignment” and compare this new

policy to the four storage assignment policies developed by Glock and Grosse (2012). The storage assignment policies studied in this paper are briefly explained in the following:

1. *Random assignment*: Items are assigned randomly to the stillages. Previous work used this policy as a benchmark for evaluating other assignments, which is also done in this paper. It was found that the random storage assignment usually leads to the highest average travel distance.

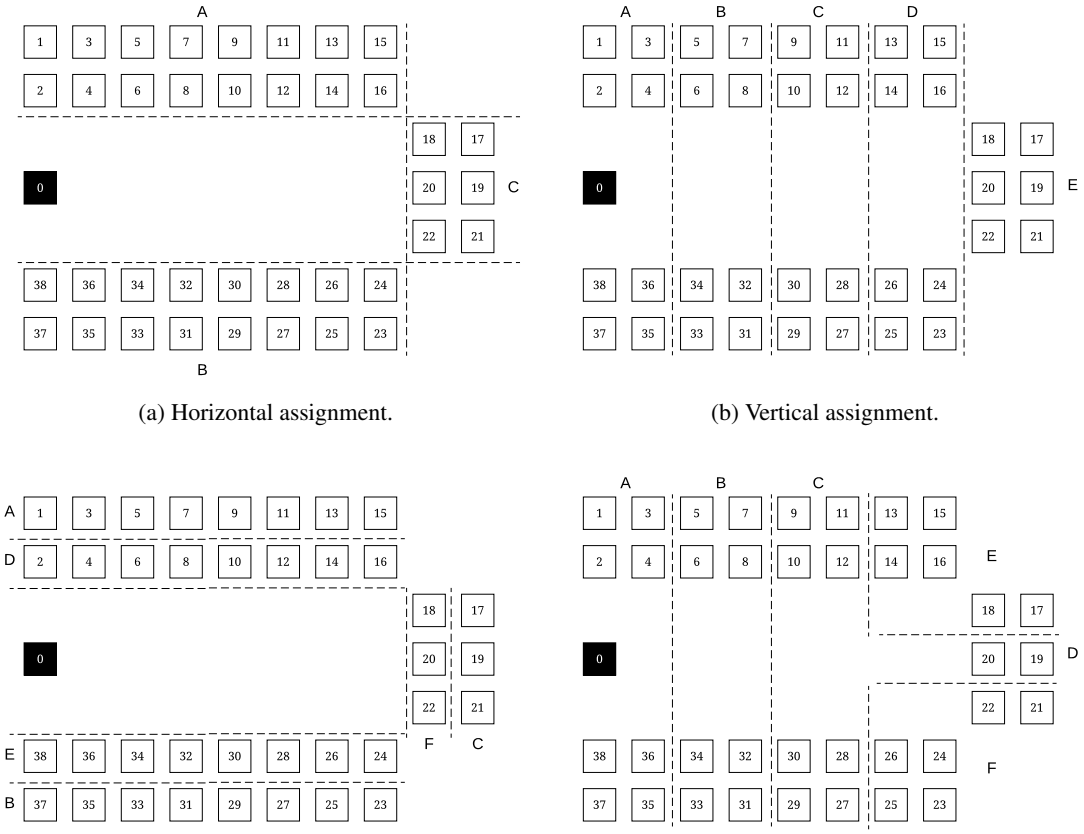
2. *Horizontal assignment*: First, items are sorted in descending order of their pick frequency. More frequently requested items are assigned to locations along the stillages on the left side of the zone referred to as zone A in Figure 4.9a. Once the stillages in zone A have been filled, items are assigned to zone B, then to zone C etc. Glock and Grosse (2012) showed that a horizontal assignment is especially beneficial for wide aisles, as it helps to avoid unproductive crossings of the aisle.

3. *Vertical assignment*: Items are again sorted in descending order of their pick frequency. Items are then assigned to both parallel aisles from left to right, as can be seen in Figure 4.9b. First, stillages in zone A are filled, followed by zones B, C etc. This type of assignment was found to produce better results for longer and narrower zones in earlier research. In such cases, the picker saves more travel distance by crossing the zone to pick items, instead of continuing along the same row of stillages.

4. *Upper/Lower assignment*: This storage assignment policy exploits the difference in the effort associated with exchanging upper and lower level stillages by assigning frequently-required products (that are assumed to result in frequent exchanges of stillages) to upper level stillages (Section 4.5.4). Therefore, items are again sorted in descending order of their pick frequency and then assigned to the upper level first before the lower level stillages are filled. Figure 4.9c illustrates the preference of item allocation following the order: A,B,..,E,F. Glock and Grosse (2012) reported that this policy is especially beneficial in case the effort of exchanging stillages differs strongly between upper and lower level stillages.

5. *Radial assignment*: This new storage assignment policy is introduced in this paper, where again items are sorted in decreasing order of pick frequency. Frequently required items are allocated radially closer to the depot location. However, the optimal depot location is usually unknown when items are assigned. Therefore, we must assume a given depot location for the radial assignment policy. In this paper, we will investigate four alternative assumed depot locations $\chi \in \{0, 0.25 \cdot l, 0.5 \cdot l, 0.75 \cdot l\}$. We emphasize that the depot

location is only assumed to determine the assignment; for the optimization, χ is still freely adjustable. Figure 4.9d depicts the alphabetical sequence of filling the stillages for the assumed depot location $\chi = 0$. It is expected that this will reduce the distance covered at the beginning and end of each tour, and should lead to good results for larger zones.



(a) Horizontal assignment.

(b) Vertical assignment.

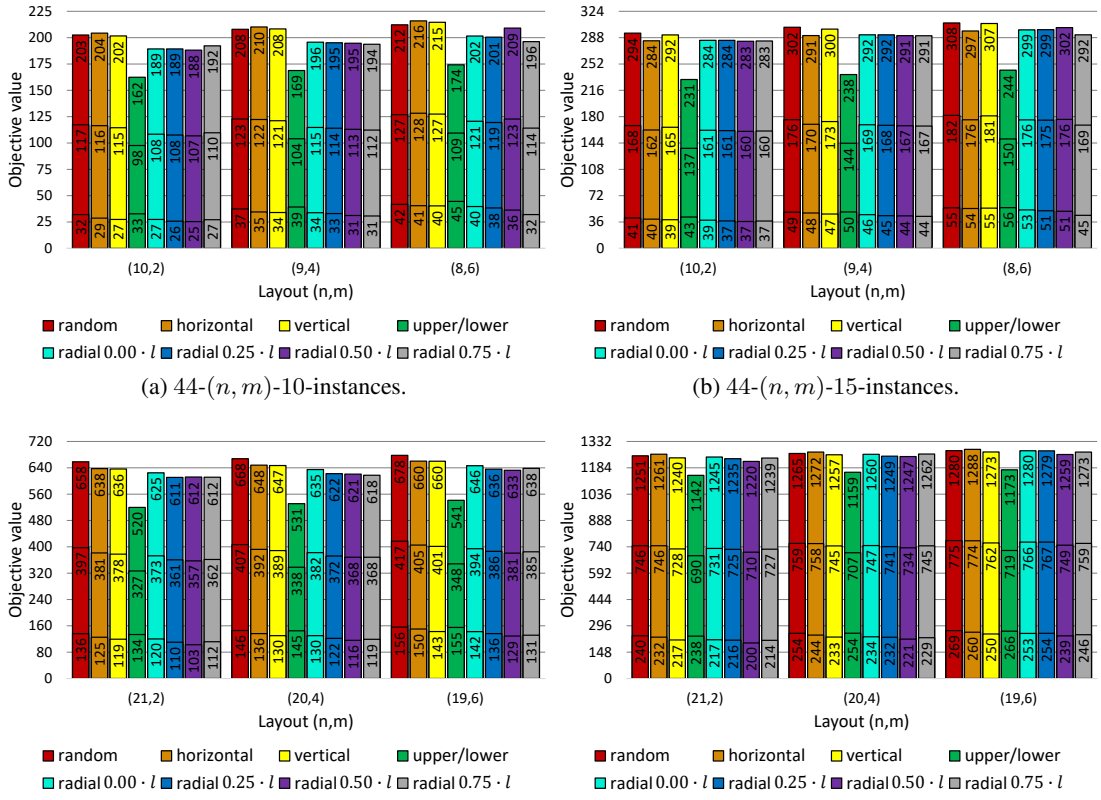
(c) Upper/Lower assignment.

(d) Radial assignment for $\chi = 0$.

Figure 4.9.: Storage assignment policies for U-shaped order picking zones

In the following, we investigate the effects of these storage assignment policies. We generate new instances based on the default instances of the performance evaluation (cf., Table 4.2), by assigning the items according to the presented storage assignment policies instead of randomly. I.e., we neither change the picklists nor the item weights, but only the assignment of items to stillages. Furthermore, we investigate the effect of different layouts, namely $(n, m) \in \{(10, 2), (9, 4), (8, 6)\}$ for the instances with 44 stillages and $(n, m) \in \{(21, 2), (20, 4), (19, 6)\}$ for the instances with 88 stillages.

To evaluate the effort for exchanging stillages, we assume exchanging an upper stillage takes 3 min and exchanging a lower stillage takes 5 min. Assuming a picker has a travel speed of 1.2 $\frac{\text{m}}{\text{s}}$, this means that exchanging an upper (a lower) stillage causes an increase in the objective of $3 \cdot 60 \cdot 1.2 = 216$ ($5 \cdot 60 \cdot 1.2 = 360$), since the objective value is measured in the normalized time the picker requires to travel 1 m. However, stillages only need to be exchanged once they are empty. To account for this, we assume all stillages have an equal capacity of Q^{stillage} and need to be exchanged once the capacity is depleted. We can then calculate the time-share per picklist to exchange a stillage. For example, assume a given picklist where item j is required with q_j and item j is stored in an upper stillage. Based on our assumptions, this would result in an additional objective value of $\frac{q_j}{Q^{\text{stillage}}} \cdot 216$ for the respective item. We evaluate the exchange effort for two different stillage capacities $Q^{\text{stillage}} \in \{50, 100\}$ and solve all instances with the proposed DP approach. The results of the experiment are summarized in Figure 4.10.



(a) 44-(n, m)-10-instances.

(b) 44-(n, m)-15-instances.

(c) 88-(n, m)-30-instances.

(d) 88-(n, m)-60-instances.

Figure 4.10.: Effects of different storage and layouts on the efficiency of U-shaped order picking zones.

Figure 4.10 shows the mean objective values for (a) the 44-(n, m)-10-, (b) the 44-(n, m)-15-

, (c) the 88-(n, m)-30-, and (d) the 88-(n, m)-60-instances in separate bar charts. Each bar consists of three stacked bars. The lowest bar shows the objective value if the exchange effort is not considered, the middle bar shows the objective value if $Q^{\text{stillage}} = 100$, and the top bar if $Q^{\text{stillage}} = 50$. Note that all bars are measured from the bottom line (i.e., 0).

Concerning the efficiency of different layouts, our results confirm the findings of Glock and Grosse (2012) and Diefenbach and Glock (2019) in that narrow layouts are beneficial. This is the case for all tested instances. For the most efficient storage assignment policies, the results show that the upper/lower assignment is best if the exchange effort is taken into account. This is already true for $Q^{\text{stillage}} = 100$ but especially evident for $Q^{\text{stillage}} = 50$, the reason being that upper/lower assignment is especially designed to minimize the exchange effort. Interestingly, upper/lower assignment is the worst assignment policy (except for random storage) if solely the pick effort is considered. In that case, the newly proposed radial assignment with $\chi \in \{0.5 \cdot l, 0.75 \cdot l\}$ performs best.

Concerning the efficiency of the upper/lower assignment if the exchange effort is considered, we note that the objective value is strongly dependent on the effort for exchanging stillages and the stillage capacity Q^{stillage}, where higher capacities cause lower exchange efforts. While we only considered $Q^{\text{stillage}} \in \{50, 100\}$ in our experiment, higher capacities are also possible. By extrapolating our results, we can calculate the theoretical break-even stillage capacities, from where on the upper/lower assignment is outperformed by the (former) second best assignment policy, namely radial assignment with $\chi = 0.5 \cdot l$ or $\chi = 0.75 \cdot l$. These break-even stillage capacities are given in Table 4.3.

instances	44-(n,m)-10-instances			44-(n,m)-15-instances			88-(n,m)-30-instances			88-(n,m)-60-instances		
layout	(10,2)	(9,4)	(8,6)	(10,2)	(9,4)	(8,6)	(21,2)	(20,4)	(19,6)	(21,2)	(20,4)	(19,6)
break-even at	223.60	197.03	137.96	475.29	487.81	281.00	199.37	217.00	225.37	151.62	183.70	211.81
break-even with	radial $0.50 \cdot l$	radial $0.75 \cdot l$	radial $0.75 \cdot l$	radial $0.50 \cdot l$	radial $0.75 \cdot l$	radial $0.75 \cdot l$	radial $0.50 \cdot l$	radial $0.75 \cdot l$	radial $0.75 \cdot l$	radial $0.50 \cdot l$	radial $0.50 \cdot l$	radial $0.50 \cdot l$

Table 4.3.: Stillage capacity from where on the radial assignment is more efficient than the upper/lower assignment.

Finally, we investigate the influence of the items' demand skewness on the storage assignment policies' efficiency. For the previous experiments, we assumed that item demand follows a $20/60$-Pareto distribution (cf., Section 4.5.1). However, more and less skewed demand distributions are also common in practice. Therefore, we also consider a $20/80$- and a $20/40$-Pareto distribution in the following. For each demand distribution and instance size, we generated ten new instances, which we solved for all layout options and storage assignment policies (except for random storage, where the item demand distribution is insignificant). The results are given

in Table 4.4, which shows the relative change (in %) of the mean objective value (including the exchange effort for stillages, where we set $Q^{\text{stillage}} = 100$) compared to the default $20/60$-Pareto distribution. The results are summarized over all layout options.

instances		44-(n,m)-10-instances			44-(n,m)-15-instances			88-(n,m)-30-instances			88-(n,m)-60-instances		
demand skewness		80/20	60/20	40/20	80/20	60/20	40/20	80/20	60/20	40/20	80/20	60/20	40/20
storage assignment policy	horizontal	-4.6	0.0	4.8	1.7	0.0	2.6	-3.2	0.0	2.5	-0.1	0.0	0.2
	vertical	-3.3	0.0	8.9	-0.2	0.0	2.0	-3.1	0.0	1.8	0.1	0.0	1.4
	upper/lower	-2.5	0.0	1.7	2.6	0.0	0.2	-2.2	0.0	0.4	0.7	0.0	0.0
	radial 0	-4.9	0.0	10.1	0.4	0.0	1.3	-4.3	0.0	2.0	-0.5	0.0	0.9
	radial 0.25	-2.4	0.0	9.5	0.4	0.0	1.3	-3.3	0.0	4.8	-0.5	0.0	1.1
	radial 0.5	-2.5	0.0	8.1	-0.5	0.0	1.2	-4.4	0.0	3.6	0.1	0.0	1.6
	radial 0.75	-4.0	0.0	8.7	0.6	0.0	2.3	-4.5	0.0	3.7	-0.2	0.0	0.3

Table 4.4.: Influence of the demand skewness on the storage assignment policies' efficiency.

Table 4.4 generally indicates that more skewed demand distributions result in lower objective values for all storage assignment policies. This was to be expected due to all considered storage assignment policies being demand-based, meaning they are specifically designed to make use of a skewed item demand distribution. The benefits of a more skewed demand distribution are primarily relevant if the U-zone's capacity is large compared to the order size $|\Omega|$. Otherwise, the effect is marginal. Moreover, if we compare the best storage assignment policies from the previous tests, namely upper/lower assignment and radial assignment with $\chi \in \{0.5 \cdot l, 0.75 \cdot l\}$, we gain an interesting insight. The demand skewness has a much lower influence for the former than for the latter. This indicates that for highly skewed demand, the radial assignment with $\chi \in \{0.5 \cdot l, 0.75 \cdot l\}$ becomes more beneficial, while for lower demand skewness, the upper/lower assignment is superior.

4.6. Conclusion

This paper considers the order picker routing problem in U-shaped order picking zones. The assumed order picking zones are built from stillages stacked one atop another and arranged in a U-shape with a movable depot at its center-line, where items are dropped off during order picking. We show that the problem is NP-hard and develop the first exact solution procedure, which is based on combinatorial Benders decomposition. Furthermore, we develop a new heuristic solution procedure based on dynamic programming by expanding the concept of a heuristic sweep algorithm from the literature, such that the new heuristic is guaranteed to find solutions that are at least as good as the ones of the sweep algorithm.

In a computational study, we compare the performance and runtime of our two newly proposed algorithms and the sweep algorithm. We find that the exact procedure is sufficiently fast to

solve small problem instances in acceptable runtime but struggles with larger ones. Both the sweep algorithm and the dynamic programming approach run very fast with at most ten seconds runtime even for large instances. Comparing the results of the two heuristics to the exact solutions, we find the optimality gaps are very small at below 1% on average and zero for many instances. Comparing both heuristics, we find that the results of the newly developed dynamic programming approach are on average 0.55% better than the ones of the sweep algorithm.

Beyond that, we investigate the effects of having a movable depot compared to a static depot and the influence of different storage assignment policies, where we suggest a new policy, called radial assignment, and compare it to various policies from the literature. We derive the following managerial insights:

- A movable depot is always favorable compared to a stationary depot. However, for larger U-zones and longer picklists, a stationary depot in the middle of the U-zone is almost as beneficial as having a movable one. In our experiments, objective values were between 0.78% (for large U-zones) and 6.37% (for small U-zones) higher if the depot was fixed compared to the movable depot case.

- Having a stationary depot directly at the entrance of the U-zone is the worst option by a large margin in all of our experiments. It is therefore not advisable.

- In our experiments, the newly proposed radial assignment policy minimizes the effort for order picking, while the upper/lower assignment policy minimizes the combined effort for order picking and exchanging empty stillages. However, the latter is strongly dependent on the assumed stillage capacities. For high stillage capacities, radial assignment remains the best policy even if the effort for exchanging empty stillages is considered. Moreover, radial assignment appears superior for highly skewed item demand distributions, while upper/lower assignment is beneficial for lower demand skewness.

- In our experiments, narrow U-zones are advantageous, which validates the results from the literature.

- The relatively simple sweep algorithm, adapted to U-shaped picking zones with a movable depot, performs quite well. This may be good news for practitioners who do not wish to implement complicated optimization logic.

We base our problem definition on some assumptions. Among the more critical ones are the assumption that the depot is point-like and that pickers travel in Euclidean paths, while actual human walk paths often resemble an elongated S-shape (cf., Diefenbach and Glock, 2019). We

regard the resulting errors to be comparatively small. Nevertheless, future research may aim to improve or refine our assumptions.

Furthermore, we consider storage assignment only in a rudimentary way by comparing various assignment policies. However, as our computational study shows, storage assignment can greatly influence a U-zone's efficiency. Future research may investigate the possibility to store multiple kinds of items per stillage, which is sometimes found in practice. Our solution procedures are already suited for such scenarios, but we did not consider suitable storage assignment policies. Moreover, it could be advisable to consider a combined storage assignment and routing problem in the future. As the performance of our exact solution approach suggests, this is most likely only possible with heuristic solution approaches, although investigating stronger cuts might also present an interesting and promising opportunity to improve our BD's performance further in the future.

In this paper, we considered a fully manual system, as currently automation plays a subordinate role for U-shaped order picking zones. However, automation becomes increasingly important for order picking in general, where it has achieved significant performance increases in recent years (Jaghbeer et al., 2020). Looking into future developments for U-shaped picking areas, a logical step would be to automate depots, enabling them to (autonomously) relocate while the picker processes pick tours. Future research may investigate the benefits of such (semi-)automated systems and thereby encourage the development of suitable technologies.

With this paper being the most recent addition, U-shaped order picking zones have been increasingly studied in recent years, since they were first introduction by Glock and Grosse (2012). However, there has not yet been a comprehensive comparison between U-shaped layouts and conventional layouts with parallel shelves. Especially from a practical point of view, it might be desirable to have some guidelines about when which layout poses which benefits. The insight gathered in previous works and in this paper may spark future research into this topic.

4.A. Appendix

Proposition 4.5. *Given two points $P_1 = (x_1^p, y_1^p)$ and $P_2 = (x_2^p, y_2^p)$ with Cartesian coordinates, the inequality $\sqrt{2} \cdot D^e(P_1, P_2) - D^r(P_1, P_2) \geq 0$ always holds true.*

Proof. To simplify notation, we define $l_x^p = |x_1^p - x_2^p|$ and $l_y^p = |y_1^p - y_2^p|$ and $\frac{l_y^p}{l_x^p} = \phi \Leftrightarrow l_y^p = \phi \cdot l_x^p$. Using these definitions and the definitions of $D^e(P_1, P_2)$ and $D^r(P_1, P_2)$, it follows that $\sqrt{2} \cdot \sqrt{(1+\phi^2) \cdot (l_x^p)^2} - (1+\phi) \cdot l_x^p \geq 0$ must hold in order for Property 4.5 to be true. Rearranging yields $\sqrt{2} \cdot \sqrt{(1+\phi^2)} - (1+\phi) \geq 0 \Rightarrow \sqrt{2} \cdot \sqrt{(1+\phi^2)} \geq (1+\phi) \Rightarrow 2 \cdot (1+\phi^2) \geq (1+\phi)^2 \Rightarrow \phi^2 - 2 \cdot \phi + 1 \geq 0$. Applying the binomial theorem finally yields the inequality $(\phi - 1)^2 \geq 0$, which is always true, since the left side is a quadratic term that cannot be smaller than zero. □

4.B. Bibliography

Barachet, L. (1957). Letter to the editor – graphic solution of the traveling-salesman problem. *Operations Research*, 5(6):841–845.

Battini, D., Glock, C. H., Grosse, E., H., Persona, A., and Sgarbossa, F. (2016). Human energy expenditure in order picking storage assignment: A bi-objective method. *Computers & Industrial Engineering*, 94:147–157.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60:503–515.

Bender, P. S. (1981). Mathematical modeling of the 20/80 rule: theory and practice. *Journal of Business Logistics*, 2(2):139–157.

Bureau of Labor Statistics (2016). Occupational outlook handbook, hand laborers and material movers. U.S. Department of Labor. Online: `http://www.bls.gov/ooh/transportation-and-material-moving/hand-laborers-and-material-movers.htm` [2021-02-03].

Calzavara, M., Glock, C. H., Grosse, E. H., Persona, A., and Sgarbossa, F. (2017). Analysis of economic and ergonomic performance measures of different rack layouts in an order picking warehouse. *Computers & Industrial Engineering*, 111:527–536.

Calzavara, M., Glock, C. H., Grosse, E. H., and Sgarbossa, F. (2019). An integrated storage assignment method for manual order picking warehouses considering cost, workload and posture. *International Journal of Production Research*, 57(8):2392–2408.

Çelik, M. and Süral, H. (2019). Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. *International Journal of Production Research*, 57(3):888–906.

Çelk, M. and Süral, H. (2014). Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE transactions*, 46(3):283–300.

Cergibozan, Ç. and Tasan, A. S. (2019). Order batching operations: an overview of classification, solution techniques, and future research. *Journal of Intelligent Manufacturing*, 30(1):335–349.

Chabot, T., Lahyani, R., Coelho, L. C., and Renaud, J. (2017). Order picking problems under weight, fragility and category constraints. *International Journal of Production Research*, 55(21):6361–6379.

Chen, F., Xu, G., and Wei, Y. (2019). Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction. *International Journal of Production Research*, 57(1):228–249.

Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

De Koster, R., Le-Duc, T., and Roodbergen, K. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

Diefenbach, H. and Glock, C. H. (2019). Ergonomic and economic optimization of layout and item assignment of a U-shaped order picking zone. *Computers & Industrial Engineering*, 138:106094.

Eurostat (2016). Warehousing and transport support services statistics – NACE Rev. 2. Online: `http://ec.europa.eu/eurostat/statistics-explained/index.php/Business_economy_by_sector_-_NACE_Rev._2` [2020-02-07].

Fang, K., Wang, S., Pinedo, M. L., Chen, L., and Chu, F. (2021). A combinatorial Benders decomposition algorithm for parallel machine scheduling with working-time restrictions. *European Journal of Operational Research*, 291(1):128–146.

Füßler, D., Boysen, N., and Stephan, K. (2019). Trolley line picking: storage assignment and order sequencing to increase picking performance. *OR Spectrum*, 41(4):1087–1121.

Gademann, N. andVelde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75.

Garey, M. and Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-hardness*. San Fransisco: WH Freeman.

Glock, C. and Grosse, E. (2012). Storage policies and order picking strategies in U-shaped order-picking systems with a movable base. *International Journal of Production Research*, 50(16):4344–4357.

Glock, C. H., Grosse, E. H., Abedinnia, H., and Emde, S. (2019). An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. *European Journal of Operational Research*, 273(2):516–534.

Glock, C. H., Grosse, E. H., Elbert, R. M., and Franzke, T. (2017). Maverick picking: the impact of modifications in work schedules on manual order picking processes. *International Journal of Production Research*, 55(21):6344–6360.

Grosse, E., Glock, C., and Ballester-Ripoll, R. (2014). A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. *International Journal of Operations and Quantitative Management*, 20(2):65–83.

Grosse, E., Glock, C., Jaber, M., and Neumann, W. (2015). Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, 53(3):695–717.

Grosse, E. H., Glock, C. H., and Neumann, W. P. (2017). Human factors in order picking: a content analysis of the literature. *International Journal of Production Research*, 55(5):1260–1276.

Gu, J., Goetschalckx, M., and McGinnis, L. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1):1–21.

Gue, K. and Meller, R. (2009). Aisle configurations for unit-load warehouses. *IIE Transactions*, 41(3):171–0182.

Hanson, R., Falkenström, W., and Miettinen, M. (2017). Augmented reality as a means of conveying picking information in kit preparation for mixed-model assembly. *Computers & Industrial Engineering*, 113:570–575.

Henn, S., Koch, S., Gerking, H., and Wäscher, G. (2013). A U-shaped layout for manual order-picking systems. *Logistics Research*, 6(4):245–261.

Henn, S. and Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, 222(3):484–494.

Hong, S., Johnson, A., L., and Peters, B. A. (2012). Large-scale order batching in parallel-aisle picking systems. *IIE Transactions*, 44(2):88–106.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55(3):588–602.

Huang, D., Mao, Z., Fang, K., and Yuan, B. (2021). Combinatorial Benders decomposition for mixed-model two-sided assembly line balancing problem. *International Journal of Production Research*, pages 1–27.

Jaghbeer, Y., Hanson, R., and Johansson, M. I. (2020). Automated order picking systems and the links between design and performance: a systematic literature review. *International Journal of Production Research*, 58(15):4489–4505.

Kress, D., Müller, D., and Nossack, J. (2019). A worker constrained flexible job shop scheduling problem with sequence-dependent setup times. *OR Spectrum*, 41(1):179–217.

Kulak, O., Sahin, Y., and Taner, M. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, 24(1):52–80.

Lu, W., McFarlane, D., Giannikas, V., and Zhang, Q. (2016). An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, 248(1):107–122.

Masae, M., Glock, C. H., and Grosse, E. H. (2020a). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, 224:107564.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2020b). Optimal order picker routing in the chevron warehouse. *IISE Transactions*, 52(6):665–687.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2021). A method for efficiently routing order pickers in the leaf warehouse. *International Journal of Production Economics*, 234:108069.

Matusiak, M., de Koster, R., Kroon, L., and Saarinen, J. (2014). A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *European Journal of Operational Research*, 236(3):968–977.

Mowrey, C. and Parikh, P. (2014). Mixed-width aisle configurations for order picking in distribution centers. *European Journal of Operational Research*, 232(1):87–97.

Muppani, V. and Adil, G. (2008). A branch and bound algorithm for class based storage location assignment. *European Journal of Operational Research*, 189(2):492–507.

Otto, A., Boysen, N., Scholl, A., and Walter, R. (2017). Ergonomic workplace design in the fast pick area. *OR Spectrum*, 39(4):945–975.

Öztürkoğlu, Ö., Gue, K., and Meller, R. (2014). A constructive aisle design model for unit-load warehouses with multiple pickup and deposit points. *European Journal of Operational Research*, 236(1):382–394.

Pan, J. C.-H., Shih, P.-H., and Wu, M.-H. (2015). Order batching in a pick-and-pass warehousing system with group genetic algorithm. *Omega*, 57:238–248.

Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127.

Petersen, C. and Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92(1):11–19.

Petersen, C., Aase, G., and Heiser, D. R. (2004). Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, 34(7):534–544.

Petersen, C., Siu, C., and Heiser, D. R. (2005). Improving order picking performance utilizing slotting and golden zone storage. *International Journal of Operations & Production Management*, 25(10):997–1012.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., and Trotter, L. E. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2-3):343–359.

Ratliff, H. and Rosenthal, A. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521.

Reyes, J., Solano-Charris, E., and Montoya-Torres, J. (2019). The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, 10(2):199–224.

Roodbergen, K. and Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883.

Roodbergen, K. and Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43.

Roodbergen, K., Sharp, G., and Vis, I. (2008). Designing the layout structure of manual order picking areas in warehouses. *IIE Transactions*, 40(11):1032–1045.

Roodbergen, K. and Vis, I. (2006). A model for warehouse layout. *IIE Transactions*, 38(10):799–811.

Roodbergen, K., Vis, I., and Taylor Jr, G. (2015). Simultaneous determination of warehouse layout and control policies. *International Journal of Production Research*, 53(11):3306–3326.

Rushton, A., Croucher, P., and Baker, P. (2014). *The handbook of logistics and distribution management: Understanding the supply chain*. Kogan Page Publishers.

Scholz, A., Henn, S., Stuhlmann, M., and Wäscher, G. (2016). A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, 253(1):68–84.

Tadumadze, G., Emde, S., and Diefenbach, H. (2020). Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. *OR Spectrum*, 42(2):461–497.

Theys, C., Bräysy, O., Dullaert, W., and Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763.

Tompkins, J., White, J., Bozer, Y., and Tanchoco, J. (2010). *Facilities planning*. John Wiley & Sons.

Van Gils, T., Caris, A., Ramaekers, K., and Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, 277(3):814–830.

Van Gils, T., Ramaekers, K., Caris, A., and de Koster, R. B. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, 267(1):1–15.

Vaughan, T. (1999). The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37(4):881–897.

Žulj, I., Glock, C. H., Grosse, E. H., and Schneider, M. (2018a). Picker routing and storage-assignment strategies for precedence-constrained order picking. *Computers & Industrial Engineering*, 123:338–347.

Žulj, I., Kramer, S., and Schneider, M. (2018b). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, 264(2):653–664.

Paper 5: Multi-depot electric vehicle scheduling in in-plant production logistics considering non-linear charging models[10]

Abstract: Electric vehicle scheduling is concerned with assigning a fleet of electrically powered vehicles to a set of timetabled trips. Since the range of these vehicles is limited, charging breaks need to be scheduled in-between trips, which require detours and time. This paper presents a novel electric vehicle scheduling problem with multiple charging stations in an in-plant logistics setting with the objective of minimizing the required fleet size. Contrary to previous works, we consider constant, linear and non-linear battery charging functions, which, among other things, allows to model realistic non-linear lithium-ion battery charging. We present an integer programming model and an exact branch-and-check solution procedure, which is based on decomposing the problem into a master and a subproblem. The former is concerned with assigning vehicles to trips while relaxing the battery constraints. The latter schedules charging breaks and checks if the master problem's solution is feasible with regard to the non-relaxed battery constraints. Our computational tests show that solving the IP model with a standard solver (CPLEX) is inferior to the branch-and-check approach, which generally performs well even for practically relevant instance sizes. Furthermore, we derive some insights into the influence of the charging mode and maximum battery capacity on the required fleet size. Lastly, we investigate the effects of the number of warehouses (with respective charging stations).

Keywords: Scheduling; Non-linear charging; Tow train; In-plant logistics; Electric vehicle scheduling

[10]This chapter has been published as: Diefenbach, H., Emde, S., and Glock, C. H. (2022). Multi-depot electric vehicle scheduling in in-plant production logistics considering non-linear charging models. *European Journal of Operational Research*, in press. DOI: `https://doi.org/10.1016/j.ejor.2022.06.050`

5.1. Introduction

Vehicle scheduling is concerned with assigning a set of timetabled trips to a set of vehicles to satisfy some objective, typically the minimization of the size of the vehicle fleet, or of the total deadheading time, or of a combination of both. While vehicle scheduling problems have received a lot of attention from academia, interest in scheduling electric vehicles, whose range is severely limited by their battery, has only recently increased.

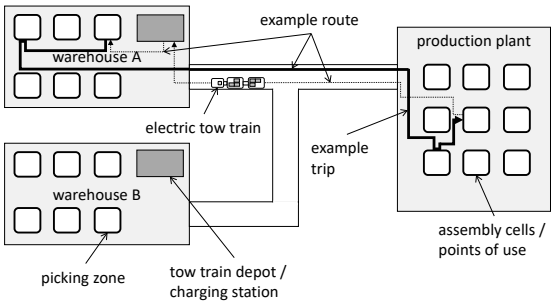
This paper considers the multi-depot electric vehicle scheduling problem with the objective of minimizing the fleet size (EVSP-MD-FS). It can be briefly described as follows. Given are a set of timetabled service trips with predetermined start and end times, a fleet of homogeneous electric vehicles with a limited battery capacity, a set of depots for the vehicles, and a set of charging stations (which can correspond to the depots), where the vehicles can be charged. Which vehicle should process which service trip such that no battery ever runs empty, the timetabled processing times are met, and the total fleet size is minimal? Batteries can be (partially or fully) charged in-between service trips at the charging stations, time permitting. We consider constant, linear and non-linear battery charging functions, which allows modeling the two most common charging technologies, battery swapping and lithium-ion battery plug-in charging with a realistic, non-linear charging function. In contrast to many classic versions of vehicle scheduling problems (VSP) and electric vehicle scheduling problems (EVSP), we consider deadheading trips solely in the form of additional time and battery requirements, but not as costs in the objective function.

EVSP-MD-FS is motivated by an application in in-plant production logistics, where the typical part feeding process in mixed-model assembly plants is as follows (Boysen et al., 2015, Emde et al., 2018). Customer orders are sequenced several days before production begins, such that the exact part demand in every work cycle is known with certainty. Subsequently, service trips – trips for re-supplying the work cells –, schedules, and loads of the tow trains – electric vehicles attached to a handful of wagons carrying parts to the assembly line, see Figure 5.1b – are determined. Since space at the work cells on the shopfloor is scarce, it is usually not possible to store parts for more than a few work cycles there. Hence, deliveries are frequent and come in small lots. Logistics workers in the warehouses prepare the parts needed at the assembly line in stillages. A service trip – or trip for short – by the tow train then consists of the following steps: pick-up of the prepackaged stillages from the respective picking stations in the warehouse, drive to the assembly plant, and drop-off the stillages at the appropriate assembly cells. After completing a trip, the tow train may set off to the start point of a consecutive trip, to a recharging station or, at the end of the shift, to a depot, where tow trains are also stationed at the beginning

of the shift. The parts that need to be picked, the picking zones involved, and the assembly stations that need to be visited on a given trip depend on the production sequence. Since tow trains are electric (combustion engines are generally not legally allowed on shopfloors), the vehicles' batteries must be periodically charged at charging stations.

In classic VSP, the minimization of deadheading distances is often the primary objective. However, this is not the case for the application in in-plant production logistics due to the following properties of the latter. Apart from distances being generally shorter, routes are frequently circular milk-runs: a vehicle sets off from a logistics area, executes a service trip, and returns to the logistics area for the next trip. Therefore, the vehicles pass by depots and stations quite frequently. Because of this and due to the short driving distances, deadheading – while still necessary to some degree – is less of a concern in production logistics. Instead, the number of vehicles and operators are the main cost drivers, which, therefore, should be minimized (Golz et al., 2012, Emde et al., 2018). It is essential to minimize the fleet size on an operational level, since operational decisions entail the satisfaction of tactical objectives – requiring fewer vehicles for daily operations enables a company to reduce their (tactical) fleet size in the long run.

Throughout the paper, we distinguish trips from routes. Trips always refer to pre-planned service trips as explained above. A vehicle's route is a sequence of consecutively executed trips, visited stations, and depots. An exemplary tow train's route, where the charging stations correspond to the depots and the executed trip is marked in bold, is depicted in Figure 5.1a.



(a) Example of a tow train making a delivery on factory premises.

(b) Electric tow train[11].

Figure 5.1.: Electric vehicles carrying parts to final assembly.

Vehicles for in-plant production logistics, such as tow trains, for example, are usually equipped

[11] By AutoGuidedVehicles [Public domain], from Wikimedia Commons

with lithium-ion batteries. For these vehicles, two approaches are common to replenish empty batteries: either battery swapping (i.e., exchanging a depleted battery with a charged one) or plug-in charging. For the latter, either a constant current-constant voltage (CC-CV) scheme or a constant power-constant voltage scheme (CP-CV) is usually applied (Liu, 2013). Both charging schemes consist of two phases and aim to reduce battery degradation by avoiding excessive voltage levels during charging. In the first phase, the battery is charged with a constant current (CC-CV) or power (CP-CV), which results in a linear increase of the battery's state-of-charge (SOC) over time. At some point, usually when the SOC is at about 80%, the battery's voltage reaches a critical level and the second phase begins. Here, the voltage is kept constant at the maximum acceptable level resulting in a declining charging rate over time such that the battery's SOC gradually approaches the maximum charge. The resulting charging function is a concave piece-wise non-linear function (cf., Figure 5.3a).

The contribution of this paper is as follows. We present a novel application for the multi-depot EVSP in in-plant logistics, thereby extending the electric vehicle milk-run scheduling problem, first introduced by Emde et al. (2018), by considering multiple depots. Contrary to most previous works on the EVSP, we consider realistic lithium-ion battery charging functions, including battery swapping and non-linear charging functions. We develop a MIP model and propose the first exact solution method, based on branch-and-check, for this kind of problem with non-linear battery charging. The solution approach is shown to solve instances generated based on realistic specifications in good time. Moreover, we provide some insight into the effect of the charging technology as well as warehouse and depot placement on system performance.

The remainder of this paper is organized as follows. We review the literature in Section 5.2. In Section 5.3, we formally define the problem. A compact mixed-integer programming (MIP) model is provided in Section 5.4. We propose our branch-and-check solution method in Section 5.5, and test it in a computational study in Section 5.6. Finally, Section 5.7 concludes the paper.

5.2. Literature review

Since the seminal work of Saha (1970), VSP have received a lot of attention from academia. Bunte and Kliewer (2009) and Ibarra-Rojas et al. (2015) provide surveys. Scheduling of electric vehicles has been addressed considerably less often. There are some models that impose a maximum length or duration on schedules for individual vehicles (Freling and Paixão, 1995, Haghani and Banihashemi, 2002, Wang and Shen, 2007); however, this is not the same as taking into consideration battery capacities and charging intervals. Only in recent years, the electric vehi-

cle scheduling problem (EVSP) has gained increasing attention from the scientific community. The majority of research deals with the EVSP based on the application in public transportation. Although the problems disucssed in the context of public transportation have a different focus than EVSP-MD-FS in terms of objectives (mostly minimization of deadheading cost), problem structure (e.g., chaining multiple visits to charging stations in a row), and instance data (e.g., the operational area is much larger, deadheading trips are substantially shorter than service trips, etc.), from a modelling perspective they are nonetheless similar. We therefore discuss the state of the art in public transport EVSP in more detail in the following.

Wen et al. (2016) discuss a multi-depot electric VSP, where vehicles have a limited battery capacity and can be charged at any given charging station. They assume that batteries are charged at a linear rate (i.e., battery swapping is not considered) and that the energy consumption depends on the driving distance. Moreover, they allow for chains of charging, i.e., visiting a sequence of charging stations between trips. The goal is to minimize the weighted sum of the fleet size and the total deadheading distance. They present a heuristic based on adaptive large neighborhood search.

Wang et al. (2017) present a case study of the electric mass transit system in Davis, California. They develop a holistic optimization model that considers the total annual operating costs, including, among other things, charging costs and the cost for setting up the charging stations in the first place. Similarly, Chao and Xiaohong (2013) present a case study from Chinese metropolitan areas. Messaoudi and Oulamara (2019) study the operational problem of optimizing the deployment of electric buses based on a case study at an urban public transportation service. They assume that buses can only charge at a central depot. Besides scheduling charge events, they aim to optimize the assignment of buses to line services and to parking positions with varying accessibility at the depot.

Reuer et al. (2015) and Adler and Mirchandani (2017) investigate the electric VSP where vehicles must always be charged to capacity, i.e., no partial charging is allowed. The latter assume a constant time for battery charging and allow for chains of charging with the aim to minimize the fleet size and operational costs for charging. They solve the considered problem using a branch-and-price approach. Reuer et al. (2015) assume that charging times are constant and minimize the total deadheading cost. Besides only considering the scheduling of electric vehicles, Reuer et al. (2015) also consider a scheduling problem with a combined fleet of electrical and conventional vehicles, where the latter have unlimited range. They also assume that charging stations correspond to the start or end points of a subset of the trips, such that no detours are necessary to reach charging stations.

Based on the model and a heuristic solution approach developed by Adler and Mirchandani (2017), Olsen and Kliewer (2018) study the validity of the often assumed linear and constant-time battery charging compared to actual non-linear charging. They conclude that the assumption can cause significant discrepancies and suggest a more sophisticated, i.e., non-linear, modeling approach. Olsen and Kliewer (2020) follow this stream of research further, finding that constant-time charging (as an approximation of non-linear charging) overestimates the time required to charge, which results in too many vehicles being used. On the other hand, linear charging is found to underestimate charging times, which results in too tight vehicle schedules that are infeasible in practice. Moreover, Olsen and Kliewer (2020) investigate the number of vehicles charging simultaneously at the same station. They find that the maximum number of simultaneously charging vehicles ranges from about two to eight, depending on the exact instance and the assumed battery charging model.

Li (2014) develops two models, one for electric bus scheduling and one for scheduling buses with limited range that cannot be charged between tours, however. For the electric buses, either battery swapping or fast charging is assumed. Both are modeled as constant-time charging. The objective is to minimize the weighted fleet size and total operational costs. An exact branch-and-price solution approach is developed and shown to perform well on real-world instances.

Sassi and Oulamara (2017) study the scheduling of a given fleet of both electric and conventional vehicles with the objective of minimizing the total charging and traveling cost, where they only consider plug-in charging. Similarly, Zhou et al. (2020) study the scheduling of a fleet of electric and conventionally powered vehicles with the aim to minimize the total operational costs as well as carbon emissions. Moreover, Yao et al. (2020) consider an EVSP with multiple types of electric buses, where certain trips can only be performed by certain types of buses. They further assume that vehicles can only charge at their nearest depot or base depot, where charging is modeled as constant-time charging.

Janovec and Kohání (2019a) study the effects of degrading battery capacities on an EVSP with the objective of minimizing the fleet size. They assume linear battery charging, model the problem as a MIP and solve it using commercial software. The same model and solution approach is applied to solve real-world instances in Janovec and Kohání (2019b).

While most of the previously considered works assume the trip timetables as given, Teng et al. (2020) investigate the integrated optimization of trip timetables and vehicle schedules with the objective of minimizing the fleet size and charging costs. They assume that vehicles can only be charged at a single depot and must always be charged to the maximum battery capacity. They solve the integrated problem heuristically using particle swarm optimization. Somewhat

related, Schneider et al. (2014) and Goeke and Schneider (2015) are concerned with the routing of electric vehicles. Their aim is to route a fleet of electric vehicles such that, for a set of given locations, each location is visited within a respective time window. The electric vehicles have a limited range and must visit stations to charge. While the former consider a fleet of pure electric vehicles, the latter assume a mixed fleet of electric and conventionally powered vehicles.

In contrast to the work presented above, where battery charging is assumed to be linear or executed in constant time, van Kooten Niekerk et al. (2017) develop a set of EVSP models with more sophisticated battery charging. Their objective is to minimize the total cost, which depends on the fleet size, the total energy cost and the battery depreciation. They allow for non-linear charging and daytime-dependent energy costs when charging. Furthermore, they track the batteries' SOC over all dis-/charging events to assess the costs of battery depreciation. The authors formulate a MIP with simplified, linear battery charging and solve it using a commercial standard solver. In addition, they formulate a model based on column generation, where the non-linear battery charging is incorporated into the models via discretization. Based on this model, they are able to solve smaller problem instances to optimality and larger instances heuristically. However, contrary to most of the previously mentioned literature and the problem considered in this paper, van Kooten Niekerk et al. (2017) assume that charging can take place only at the end/start of certain trips, such that no detours are necessary to visit charging stations.

Zhang et al. (2021) consider an electric bus scheduling problem with multiple types of buses and multiple depots. They assume that buses can charge in-between any two service trips. However, in contrast to the majority of EVSP including EVSP-MD-FS, they assume that each vehicle can only charge at its respective home depot, which limits the flexibility of schedules significantly. Their considered objective is to minimize the total operational costs of the fleet, which includes vehicle purchasing costs, energy consumption costs and time requirement costs. Moreover, they discuss two possibilities to approximate non-linear battery charging – either by a single linear function or by a piece-wise linear function. To solve their problem, Zhang et al. (2021) present a MIP as well as a large neighborhood search heuristic. In their computational study, the authors find that the piece-wise linear approximation works well. On the other hand, the linear approximation is shown to frequently cause too tight schedules that are infeasible in practice.

Sweda et al. (2016) examine a somewhat different problem: given a single vehicle following a given route, where should the vehicle stop to recover how much charge such that the total charging cost is minimal?

To power electric vehicles, various types of batteries and battery replenishment technologies have been developed. However, in recent years, lithium-ion batteries have been established as

the dominant battery technology for electric vehicles (Den Boer et al., 2013). Their main advantage is a high power density compared to other battery technologies such as, for example, Pb-acid, NiCd, or NiMH batteries (Den Boer et al., 2013). On the downside, lithium-ion batteries are expensive and have a limited lifespan, which is further influenced by various handling and employment conditions (Barré et al., 2013). Among other factors, a greater depth of discharge, higher (dis-)charging rates and a higher SOC during storage can accelerate battery degradation (Vetter et al., 2005).

Considering the high costs of lithium-ion batteries and the influence of employment conditions on their lifetime, Pelletier et al. (2017) suggest to integrate battery degradation into decision support models for electric vehicle operations and discuss respective modeling approaches. However, according to our observations, such considerations are currently only of minor concern for intra-logistical planners and only taken into account in a rudimentary manner, for example, by limiting the depth of discharge of battery powered vehicles. The consideration of battery degradation is therefore beyond the scope of this paper.

Finally, from a strategic point of view, research has been concerned with planning optimal locations of charging stations in the context of public transportation. Liu et al. (2018), for example, consider the location planning of fast-charging stations for electric bus systems. They develop both an exact and a robust stochastic optimization model with the objective of minimizing the total costs for charging stations and required batteries. He et al. (2019) consider a similar problem, where they additionally account for costs of energy storage systems and electricity demand. Lin et al. (2019) present a large-scale charging station planning approach that not only considers charging demands but also the available power grid infrastructure. The strategic planning of charging locations is, however, beyond the scope of this paper.

To summarize the previous review and to emphasize the contribution of this paper, we conclude the overview of the EVSP with Table 5.1, which compares the versions of EVSP that are most closely related to the problem considered in this paper. The overview demonstrates the novelty of this paper by considering realistic, non-linear battery charging in a situation where charging requires detours and vehicles are free to charge at any charging station. Moreover, most VSP papers consider the problem in the context of public transportation, which requires a different focus on things like, e.g., deadheading costs and multiple visits of charging stations in a row due to the sparse charging infrastructure relative to the large service area. To the best of our knowledge, the only previous paper dealing with electric VSP in a production logistics context is Emde et al. (2018), who present a tabu search heuristic for the single depot case. Unlike this previous paper, we consider both constant-time and non-linear charging as well as multiple

depots and develop an exact algorithm.

paper	battery charging				charge tours		objective			depots		bat. per distance		other constraints	
	constant t. charging	linear t. charging	non-linear t. charging	partial charging possible	charging requires detour	free choice of charging st.	miscellaneous	w. FS + w. total TC/RC	FS	multiple	single	proportional	arbitrary	start de. equals end de.	de. have max. ve. cap.
Chao and Xiaohong (2013)	✓						✓[8]		✓	✓	✓	✓		✓	
Li (2014)[1]	✓				✓	✓	✓[9]	✓	✓		✓	✓		✓	
Reuer et al. (2015)[2]	✓					✓		✓	✓		✓	✓			
Wen et al. (2016)		✓		✓	✓	✓		✓	✓	✓	✓	✓		✓	
Adler and Mirchandani (2017)	✓				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Sassi and Oulamara (2017)[2]		✓		✓		✓	✓[10]		✓		✓	✓	✓		
Wang et al. (2017)	✓			✓	✓	✓	✓[11]				✓	✓			
van Kooten Niekerk et al. (2017)	✓	✓	✓	✓		✓	✓[12]	✓	✓	✓	✓	✓	✓	✓	
Emde et al. (2018)	✓						✓[13]		✓		✓	✓	✓		
Olsen and Kliewer (2018)[3]	✓		(✓)[6]		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Janovec and Kohání (2019a)[4]		✓		✓	✓	✓			✓		✓	✓	✓		
Messaoudi and Oulamara (2019)		✓		✓				✓	✓		✓	✓	✓		
Olsen and Kliewer (2020)	✓	✓	(✓)[6]	✓	✓	✓	✓[14]	✓	✓		✓	✓	✓		
Yao et al. (2020)	✓				✓		✓[15]		✓	✓	✓	✓	✓	✓	
Zhou et al. (2020)[5]		✓		✓	✓		✓[16]	✓	✓		✓	✓			
Zhang et al. (2021)		✓	(✓)[7]	✓	✓		✓[17]	✓	✓	✓	✓	✓	✓	✓	✓
this paper	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓

Abbreviations: bat. = battery; cap. = capacity; de. = depot; FS = fleet size; max. = maximum; TC/RC = travel cost and/or charging cost; ve. = vehicle; w. = weighted. Annotations: [1] This refers to the considered EVSP with electric buses.; [2] This refers to the considered EVSP with electric vehicles only.; [3] Their model is identical to the one of Adler and Mirchandani (2017).; [4] The problem is identical to Janovec and Kohání (2019b).; [5] This refers to the considered electric buses within their mixed-fleet model.; [6] The solution process only considers constant-time/linear charging – non-linear charging is only considered to examine solutions afterwards.; [7] Non-linear charging is approximated using a piece-wise linear function; [8] The objective is to minimize the cost of the fleet size, standby batteries, and total charge demand.; [9] The considered objective is to minimize the fleet size, the total travel distance, and total battery charging service cost.; [10] The objective is to maximize the distance covered with electric buses (instead of conventional ones), and minimize their energy consumption.; [11] The objective is to minimize the annual total electric bus recharging system operating costs.; [12] The considered objective is to minimize the fleet size, total energy cost, and total battery depreciation costs.; [13] The objective is to minimize the fleet size and the workload of the busiest vehicle.; [14] The objective is to minimize the fleet size plus the operational costs.; [15] The objective is to minimize the total annual scheduling costs.; [16] The objective is to minimize the operative costs and carbon emissions; [17] The objective is to minimize the operational costs including vehicle purchasing costs, energy consumption costs, and travel time costs.

Table 5.1.: Comparison of different electric vehicle scheduling problems.

Given that tow trains are widely employed in many production systems, it is not surprising that several publications deal with the routing and scheduling of such vehicles in a just-in-time context (e.g., Emde and Boysen, 2012, Emde and Gendreau, 2017, Emde and Schneider, 2018, Fathi et al., 2014, 2016, Zhou and Peng, 2017). Despite the fact that in-plant delivery vehicles are almost without exception electric ones, very few papers on automotive part feeding have so far considered this aspect. Hu et al. (2017) address the problem of drawing up a schedule for a single electric tow train serving a given route, where the speed of the vehicle can be adjusted to modify its energy consumption, which is to be minimized. Zhou and Tan (2018) optimize both the location of charging stations and the routes of the tow trains in a holistic approach. Finally,

Briand et al. (2018) aim to schedule tow trains such that their energy consumption is minimal. They also propose a method to derive realistic energy expenditures, showing that minimizing travel distance and minimizing energy expenditure are not always the same. All of these papers consider the superordinate problems of deciding on routes and schedules for the vehicles, not assigning individual vehicles to already timetabled trips. An overview of the whole part logistics process in the automotive industry is given by Boysen et al. (2015).

From a scheduling perspective, the EVSP-MD-FS is somewhat reminiscent of scheduling point-to-point deliveries with time windows (Emde and Zehtabian, 2019, Gschwind et al., 2019), except that trips do not only have time windows, but that they are already timetabled. Scheduling jobs with fixed starting times on a set of machines is the subject of interval scheduling (surveyed by Kovalyov et al., 2007, Kolen et al., 2007), which does not take battery charging into account, however.

5.3. Problem description

We present a formal description of the EVSP-MD-FS in Section 5.3.1 and a MIP model in Section 5.4. Like all mathematical models, our problem definition is based on some assumptions.

Specifically, we assume that all parameters, especially all trips, are known and deterministic, which is a realistic assumption in just-in-time industries, where production sequences are typically fixed several days in advance, and the exact transport demands are consequently known (Emde et al., 2012). Moreover, we consider homogeneous vehicles with fully charged batteries at the beginning of the planning horizon, which are initially all stationed at depots.

All charging stations always have sufficient space and capacity to service any docked tow trains. Clearly, this may not always be the case in practice, where charging capacities can be limited by space or the charging infrastructure's capacity. Nevertheless, it presents a common assumption in the literature (e.g., Reuer et al., 2015, Wen et al., 2016, Zhang et al., 2021). Moreover, Olsen and Kliewer (2020) found that even without restricting the stations' capacities in their model, solutions rarely had an excessive number of vehicles charging simultaneously, which is in line with our observations in practice. Finally, we consider deadheading implicitly in the constraints, but not in the objective function, because vehicle and operator costs tend to dominate the detour costs due to the relatively short driving distances (Golz et al., 2012, Emde et al., 2018).

5.3.1. Formal problem description

We base the definition of the EVSP-MD-FS on the notation summarized in Table 5.2. Let $D = \{1, \ldots, d\}$ be the set of depots, let $J = \{d+1, \ldots, d+n\}$ be the set of trips to be processed during the planning horizon and let $S = \{d+n+1, \ldots, d+n+s\}$ be the set of charging stations (called "stations" in the following). Note that we introduce offset indices to ease notation in the following definitions and models. Generally speaking, the locations of stations do not have to coincide with the depots or the start or end points of the trips, but can of course correspond to these locations. In fact, the former is the case for most of the in-plant milk-run logistic systems we observed. Each trip consists of a vehicle visiting multiple locations (i.e., assembly cells) in a given path. The exact path the vehicle takes is immaterial for the EVSP-MD-FS; we assume that this issue has been decided in a previous step. Let s_j be the start time of trip j, i.e., the time when the vehicle arrives at the trip's first location, and let e_j be the end time of trip j, i.e., the time when the vehicle departs from the trip's last location. For notational convenience, we also define $s_j = \infty$ and $e_j = 0$, $\forall j \in D \cup S$, which implies that depots and stations are always available. Without loss of generality, we assume that $s_j \leq s_{j'}$ holds, $\forall j, j' \in J : j < j'$, i.e., the trips are indexed according to their starting time in ascending order. Moreover, let $t_{j,j'}$, $\forall j, j' \in D \cup S \cup J : j \neq j'$, be the transit time of a vehicle from depot, station, or the end of trip j to depot, station, or the starting point of trip j'. For transit times, we assume that the triangle inequality holds in the sense that $t_{j,j'} \leq t_{j,j''} + \max\{e_{j''} - s_{j''}, 0\} + t_{j'',j'}$, $\forall j, j', j'' \in J \cup D \cup S$, where $\max\{e_{j''} - s_{j''}, 0\}$ is the trip's execution time if $j'' \in J$, and zero if $j'' \in D \cup S$. Let the battery capacity, i.e., maximum SOC, of the vehicle be $\hat{c}$, and the charge required to complete trip $j \in J$ be $\tilde{c}_j$, where we assume $\tilde{c}_j \leq \hat{c}$ applies, $\forall j \in J$. Note that the required charge $\tilde{c}_j$ may depend on the total distance covered, the number of stopovers, the carried load, the terrain (e.g., steep slopes), and other factors. We assume that these values have been determined beforehand and are given. To ease notation, we further define $\tilde{c}_j = 0$ and $\bar{c}_{j,j} = 0$, $\forall j \in D \cup S$. Let $\bar{c}_{j,j'}$ be the charge required to drive from depot, station or the end of trip j to depot, station, or the starting point of trip j'. As for travel times, we assume that the triangle inequality also holds for battery requirements in the sense that $\bar{c}_{j,j'} \leq \bar{c}_{j,j''} + \tilde{c}_{j''} + \bar{c}_{j'',j'}$, $\forall j, j', j'' \in J \cup D \cup S$. We assume that charging is only possible at stations. If vehicles should be able to charge at depots or at the start or end points of trips, we can add stations that correspond to the respective locations.

c	state of charge (SOC)
$\hat{c}$	maximum SOC
D	set of depots; $D = \{1, \dots, d\}$
J	set of trips; $J = \{d+1, \dots, d+n\}$
j	depot, trip or station index
S	set of charging stations; $S = \{d+n+1, \dots, d+n+s\}$
$\tilde{c}_j$	required charge to complete trip j
$\bar{c}_{j,j'}$	required charge for the transit from depot, station, or (the end of) trip j to depot, station, or (the start of) trip j'
e_j	end time of trip j
$\bar{m}_j$	maximum vehicle capacity of depot j
m	number of vehicles
r	constant time required to attach a vehicle to a charger (or to exchange the battery, depending on the context)
s_j	start time of trip j
$t_{j,j'}$	transit time from depot, station, or (the end of) trip j to depot, station, or (the start of) trip j'
$\delta_k(l)$	mapping signifying that vehicle k visits station $\delta_k(l) \in S$ or no station at all ($\delta_k(l) = 0$) after the l-th trip of its schedule
$\zeta(j, j', j'', c)$	function giving the maximum SOC at the end of trip (or depot) j' for a vehicle coming from j with SOC c visiting station $j'' \in S$ or no station at all ($j'' = 0$) in-between
$\bar{\zeta}(k, l)$	SOC at the end of the l-th trip of vehicle k
$\eta(\tau)$	battery charging function dependent on time τ; $\eta(\tau)$ is the SOC if an empty battery is charged for time τ
τ	amount of time
$\bar{\tau}(j, j', j'', c)$	time a vehicles spends charging at station j'' while transiting from j with SOC c to j'
ω_k	ordered set denoting which depots are to be visited and trips are to be performed by vehicle k in what order

Table 5.2.: Notation.

Let $\eta(\tau)$ denote the charging function over time τ, i.e, the SOC an empty battery would have after being charged for a time of τ. We assume that $\eta(\tau)$ is either a constant function, a linear function or a non-linear function, which, in the latter two cases, is capped at $\hat{c}$, i.e., when the battery is full. Moreover, we assume that a vehicle arriving at a station requires a time of r before charging can start, where, depending on the assumed charging technology, r is the time required to swap the battery or to connect the battery to the charger. This enables us to account for the

two most commonly encountered charging technologies in practice, namely battery swapping and charging via power cable with a realistic non-linear increase of the SOC. Furthermore, we assume the charging function is equal for all vehicles and stations and that vehicles are always charged whenever they spend sufficient time at a charging station. For a vehicle charging for a time of τ beginning from a SOC $c \geq 0$, the resulting SOC is given as $\eta\left(\eta^{-1}(c) + \tau\right)$, where $\eta^{-1}(c)$ is the inverse function of $\eta(\tau)$ and states the time τ required to charge an empty battery to SOC c. For the case of $\eta(\tau)$ being a constant function, we define $\eta^{-1}(c) = 0$. To further ease notation, we define $\eta^{-1}(c) = 0$, for $c < 0$, and $\eta^{-1}(c) = \infty$, for $c > \hat{c}$, as well as $\eta(\tau) = 0$, for $\tau < 0$. This makes it easy to accommodate non-linear charging.

A solution for EVSP-MD-FS then consists of

- a collection $\{\omega_1, \ldots, \omega_m\}$ of ordered sets ω_k, denoting which depots are to be visited and which trips are to be performed by each vehicle $k = 1, \ldots, m$ in what order,
- and m mappings $\delta_k : \{1, \ldots, |\omega_k| - 1\} \to j$, for $k = 1, \ldots, m$, where $\delta_k(l) = j$ specifies the deadheading route of vehicle k between the l-th trip (or depot for $l = 1$) and the $(l+1)$-th trip (or depot for $l = |\omega_k| - 1$). Two general types of deadheading routes are possible: If $j = 0$, the vehicle drives directly form trip l to $l + 1$ without any charging break. Otherwise, if $j \in S$, the vehicle visits station j after the l-th trip (or depot) to recharge its battery before heading for the $(l+1)$-th trip (or depot).

Let $\omega_k(l)$ refer to the l-th element in the ordered set. For all vehicles $k = 1, \ldots, m$, it must hold that $\omega_k(1) \in D$ and $\omega_k(|\omega_k|) \in D$, since each vehicle departs from a depot initially and returns to one at the end. Furthermore, all trips need to be processed exactly once, i.e., $\bigcup_{k=1}^{m} \bigcup_{l=2}^{|\omega_k|-1} \{\omega_k(l)\} = J$ and $\omega_k(l) \neq \omega_{k'}(l'), \forall k, k' = 1, \ldots, m; l, l' = 2, \ldots, |\omega_k| - 1 : k \neq k' \vee l \neq l'$ must hold.

Note that by these definitions, vehicles are also allowed (but not required) to visit a charging station right after leaving and before returning to a depot. Moreover, our definition of δ_k imposes that vehicles visit at most one station between consecutively executed service trips. This forecloses the possibility of chains of charging, which would be especially useful if distances between service trips are large compared to the vehicles' battery reach, such that multiple charges are required to travel between consecutive service trips. Since distances are comparatively small in intra-logistic transportation systems, visiting multiple charging stations in direct succession is all but unheard of, however.

We say a solution to an instance of EVSP-MD-FS is feasible if it is *feasibly executable*. I.e., for

every vehicle, the battery's SOC must never be negative and the executed route must not violate the time constraints. For a formal definition, see Section 5.3.2.

Moreover, while somewhat uncommon in the in-plant transportation systems that motivated this paper, there are additional constraints that planners may reasonably want to consider. First, due to space constraints, each depot $j \in D$ could be required to host no more than $\bar{m}_j$ vehicles, where $\bar{m}_j = n$ if no limit exists. Second, vehicles could be required to return to their depot of initial departure at the end of their schedule. In the following, we discuss them as optional extensions of the basic EVSP-MD-FS problem. Specifically, $|\{k \in \{1,\dots,m\} : \omega_k(1) = j\}| \leq \bar{m}_j \wedge |\{k \in \{1,\dots,m\} : \omega_k(|\omega_k|) = j\}| \leq \bar{m}_j, \forall j \in D$, must hold to account for the limited capacity of the depots. Finally, if vehicles are required to return to their depot of initial departure, it must hold that $\omega_k(1) = \omega_k(|\omega_k|), \forall k \in \{1,\dots,m\}$.

The objective of EVSP-MD-FS is to find a feasible schedule that minimizes the number of vehicles m. Hence, we minimize the number of vehicles m to which trips are assigned.

5.3.2. Feasibly executable solutions

In order to determine whether a solution is *feasibly executable*, we must determine the recharged battery during charging breaks and, hence, their duration. Each vehicle can charge its battery whenever it visits a station, i.e., whenever $\delta_k(l) \in S$. We can preempt the decision on the charging breaks' duration by taking the following proposition into account.

Proposition 5.6. *Given the trips (and depots) ω_k to be executed (or visited) by vehicle k, the policy of charging the vehicle at every charging station it visits for as long as the trip sequence ω_k permits creates a feasible solution, if any exists. I.e., using this charging policy, the vehicle reaches each trip (and depot) $j \in \omega_k$ with the maximum SOC possible given ω_k.*

Proof. The proof is by an interchange argument. Assume that in some feasible solution, vehicle k reaches trip $j \in \omega_k : j \neq \omega_k(|\omega_k|)$ with SOC c' and the consecutive trip (or depot) j' with c'', where it executes the deadheading route j'' in-between. We refer to this as case one. Alternatively, assume a second case, where j is reached with SOC $c''' = c' + \Delta c$ and j' with SOC c'''', where Δc is a positive amount of charge. In the following, we prove that $c'''' \geq c''$ is guaranteed to hold true.

In the second case, the vehicle can execute the exact same deadheading route as in the first case. If $j'' = 0$, it follows that $c'' = c' - \bar{c}_{j,j'}$ and $c'''' = c''' - \bar{c}_{j,j'} = c' + \Delta c - \bar{c}_{j,j'}$, hence $c'''' > c''$

applies. Else, if $j'' \in S$, in case one, the vehicle reaches j'' with SOC $c' - \bar{c}_{j,j''}$. Let τ' denote the time the vehicle charges at station j'', such that it has SOC $\eta\left(\eta^{-1}\left(c' - \bar{c}_{j,j''}\right) + \tau'\right)$ when leaving j''. Consequently, $c'' = \eta\left(\eta^{-1}1\left(c' - \bar{c}_{j,j''}\right) + \tau'\right) - \bar{c}_{j'',j'}$ follows. In the second case, the vehicle reaches station j'' with $c' + \Delta c - \bar{c}_{j,j''}$ and can charge for the exact same duration τ' as in the first case. Consequently, it follows that $c'''' = \eta\left(\eta^{-1}\left(c' + \Delta c - \bar{c}_{j,j''}\right) + \tau'\right) - \bar{c}_{j'',j'}$. Since $\eta^{-1}(c)$ is an increasing function of c and $\eta(\tau)$ is an increasing function of τ, $c'''' \geq c''$ is guaranteed to hold true.

The same reasoning applies for all consecutive trips in ω_k until the final depot $\omega_k(|\omega_k|)$ is reached. Hence, if there exists a feasible solution, there also exists a feasible solution where the remaining charge when reaching each $j \in \omega_k$ is maximized, that is, where charging breaks are never cut short before the vehicle must depart for its next trip. □

Therefore, at each visited station, the vehicle should charge for the maximum possible duration that still allows to timely reach the consecutive trip or depot. Given $\delta_k(l) = j''$ with $j'' \in S$, let $\omega_k(l) = j$ and $\omega_k(l+1) = j'$ be two consecutively executed trips or one of them be a depot. Furthermore, let c be the SOC when the vehicle leaves from trip (or depot) j. The optimal time $\bar{\tau}(j, j', j'', c)$ the vehicle spends charging at station j'' then follows as

$$\bar{\tau}\left(j, j', j'', c\right) = s_{j'} - e_j - t_{j,j''} - t_{j'',j'} - r, \tag{5.1}$$

which is comprised of the available time between the start of trip j' and the end of trip j minus the travel time to and from station j'' minus the plug-in/battery swap time r.

Given two consecutively executed trips $\omega_k(l) = j$ and $\omega_k(l+1) = j'$ (where either might be a depot instead) and $\delta_k(l) = j''$ (with $j'' \in S \cup \{0\}$) indicating the deadheading route in-between, the maximal SOC at the end of trip (or depot) j' can be calculated as

$$\zeta\left(j, j', j'', c\right) = \begin{cases} c - \bar{c}_{j,j'} - \tilde{c}_{j'} & \text{if } j'' = 0 \wedge e_j + t_{j,j'} \leq s_{j'} \\ \eta\left(\eta^{-1}\left(c - \bar{c}_{j,j''}\right) + \bar{\tau}\left(j, j', j'', c\right)\right) & \text{if } j'' \in S \wedge \bar{c}_{j,j''} \leq c \\ \quad -\bar{c}_{j'',j'} - \tilde{c}_{j'} & \quad \wedge \bar{\tau}\left(j, j', j'', c\right) \geq 0 \\ -1 & \text{otherwise} \end{cases}. \tag{5.2}$$

If no station is visited at the deadheading route, the first case applies. Here, the SOC at the end of trip j' is the SOC at the end of trip j minus the battery charge consumed for the transit from j to j' and for the execution of j'. If a station is visited in-between j and j', the second case applies. Here, the vehicle reaches station j'' with SOC $c - \bar{c}_{j,j''}$ and charges for a duration of

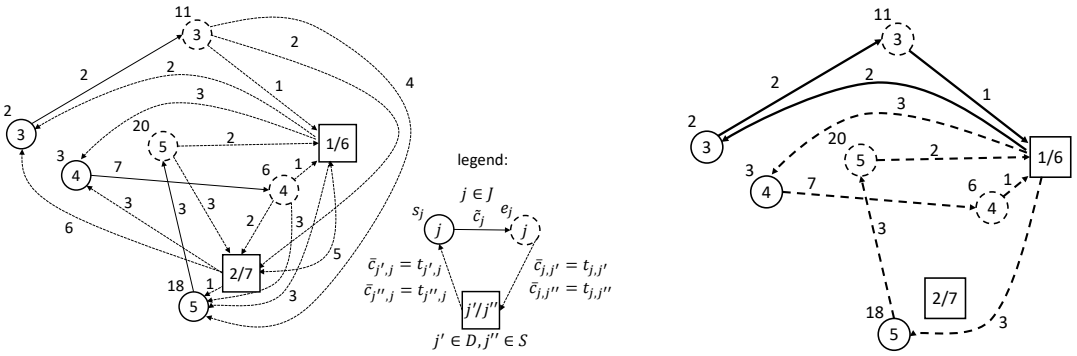
$\bar{\tau}\left(j, j', j'', c\right)$, which results in a SOC of $\eta\left(\eta^{-1}\left(c - \bar{c}_{j,j''}\right) + \bar{\tau}\left(j, j', j'', c\right)\right)$ at the end of the charging procedure. Afterwards, trip j' must be reached and executed, which reduces the SOC at the end of trip j' by $\bar{c}_{j'',j'} + \tilde{c}_{j'}$. If either the charge c is not sufficient to reach the station j'' or there is not enough time, $\zeta\left(j, j', j'', c\right)$ assumes a value < 0 to indicate infeasibility.

We define $\bar{\zeta}(k, l)$ as the SOC of vehicle k at the end of the l-th trip or depot, where $\bar{\zeta}\left(k, |\omega_k|\right)$ denotes the SOC when vehicle k reaches the final depot. For $l \geq 2$, it can be recursively calculated as

$$\bar{\zeta}(k, l) = \zeta\left(\omega_k(l-1), \omega_k(l), \delta_k(l-1), \bar{\zeta}(k, l-1)\right),$$

with $\bar{\zeta}(k, 1) = \hat{c}$ for $l = 1$.

Based on these definitions, a solution to EVSP-MD-FS is *feasibly executable* if and only if $\zeta\left(k, l\right) \geq 0, \forall k \in \{1, \ldots, m\}, l \in \{1, \ldots, |\omega_k|\}$. I.e., for every vehicle, the battery's SOC must never be negative and the executed route must not violate the time constraints (since, otherwise, $\zeta\left(j, j', j'', c\right)$ and, hence, $\bar{\zeta}(k, l)$, would assume the value -1).



(a) Example problem data.

(b) Optimal solution in the example.

Figure 5.2.: Example data and solution.

Example: Consider the example problem depicted in Figure 5.2a, consisting of $n = 3$ trips and $d = s = 2$ depots that are also charging stations. Let the battery capacity of the vehicles be $\hat{c} = 11$, the charge function be linear $\eta(\tau) = \tau$ (with the cap $\eta(\tau) = 11$, for $\tau \geq 11$) and the constant charge time be $r = 0$. Furthermore, let $\bar{m}_j = n, \forall j \in D$, and let the vehicles be not required to return to their initial depot. Note that in this example, we assume that all driving times ($t_{j,j'}$) are equal to the energy expenditures ($\bar{c}_{j,j'}$), which generally does not need to be the case. The optimal solution is depicted in Figure 5.2b, corresponding to $m = 2$ vehicles being

used. Solid arrows denote the route of vehicle 1, dotted arrows the route of vehicle 2. Vehicle 1 processes only trip 3 (i.e., $\omega_1 = \{1, 3, 1\}$, $\delta_1(1) = 0$, $\delta_1(2) = 0$, $\bar{\zeta}(1, 1) = 11$, $\bar{\zeta}(1, 2) = 7$, $\bar{\zeta}(1, 3) = 6$). Vehicle 2 processes trips 4 and 5 (in that order), returning to, first, station 6 – where it charges its battery for a time of $\bar{\tau}(4, 5, 6, 1) = 18 - 6 - 1 - 3 - 0 = 8$ – and, second, depot 1 after the trips (i.e., $\omega_2 = \{1, 4, 5, 1\}$, $\delta_2(1) = 0$, $\delta_2(2) = 6$, $\delta_2(3) = 0$, $\bar{\zeta}(2, 1) = 11$, $\bar{\zeta}(2, 2) = 1$, $\bar{\zeta}(2, 3) = 2$, $\bar{\zeta}(2, 4) = 0$).

5.3.3. Complexity

Concerning the time complexity of EVSP-MD-FS, classic (single-depot) non-electric vehicle scheduling is well-known to be tractable (Saha, 1970), as it can be reduced to finding a minimum cost perfect matching in a bipartite graph (Bertossi et al., 1987). This result also holds for the special single-depot case of EVSP-MD-FS where the battery capacity is not a limiting factor ($\hat{c} = \infty$). In this case, charging breaks need not be considered and the driving time t to/from the depot is immaterial. The problem then is to assign the given trips to as few vehicles as possible such that no trips overlap, which is equivalent to classic vehicle scheduling with the objective of minimizing the fleet size.

However, as soon as the battery capacity becomes a limiting factor, single-depot electric vehicle scheduling with finite battery capacity and the objective of minimizing the fleet size becomes intractable, as is proven by Emde et al. (2018, Proposition 3) by a reduction from bin packing. Obviously, this result extends to the multi-depot case as considered in EVSP-MD-FS. Finally, the result also holds if additional constraints are considered, since they merely generalize the basic EVSP-MD-FS problem.

5.3.4. Feasible deadheading routes

By the definitions in Section 5.3.1, a solution to EVSP-MD-FS consists of two distinctive parts. The ordered subsets $\omega_1, \ldots, \omega_m$ denote which depots are to be visited and which trips to be executed by which vehicle in which sequence. The mappings $\delta_1, \ldots, \delta_m$ indicate the deadheading route in-between the trips and depots for all vehicles $1, \ldots, m$. Given only ω_k, the respective feasible deadheading routes δ_k can be determined efficiently as follows.

Let $\omega_k(l) = j$ and $\omega_k(l + 1) = j'$ be two consecutively executed trips or either be a depot. Furthermore, let c be the SOC when the vehicle leaves from trip (or depot) j. By Proposition 5.6, provided that any feasible solution exists at all for a given ω_k, there is always a feasible

solution that reaches j' with the highest possible SOC and, consequently, completes j' with the maximum possible SOC (since $\tilde{c}_{j'}$ is constant). The latter SOC is given by $\zeta(j, j', j'', c)$ (cf., Equation 5.2), which beyond j, j' and c is solely dependent on $\delta_k(l) = j''$, the deadheading route between j and j'. Hence, it follows that

$$\delta_k^*(l) = \underset{j'' \in S \cup \{0\}}{\arg\max} \left\{ \zeta \left(j, j', j'', c \right) \right\}$$

is guaranteed to be feasible (if feasibility is attainable at all). Consequently, the maximum and therefore feasible charge at the end of trip j' is given as

$$\zeta^* \left(j, j', c \right) = \max_{j'' \in S \cup \{0\}} \left\{ \zeta \left(j, j', j'', c \right) \right\}.$$

5.4. MIP model

To enable the use of default solvers, we propose a MIP model based on the notation given in Table 5.3. We incorporate non-linear battery charging by discretizing the SOC. I.e., the SOC can only assume discrete values $c \in C$. We note that the discretization is only required for the MIP model; the solution procedure presented in Section 5.5 does not rely on discretization. For linear and constant-time charging, the discrete SOC is exact, as long as we assume that all problem parameters, i.e., travel times and required battery charges, are integer. For non-linear charging, using discrete SOC is an approximation. By scaling the problem parameters accordingly, we can, however, approximate the actual SOC to any desired precision.

C	set of discrete SOC values; index c; $C = \{0, \ldots, \hat{c}\}$; a value of c' refers to $\frac{c'}{\hat{c}}$ of the maximum SOC
$v_{j,j'}$	binary variable: 1, if trip j is associated with depot j', else 0
$x_{j,j'}$	binary variable: 1, if a vehicle transits from trip or depot j to trip or depot j' (with possible detours for charging in-between), else 0
$\beta_{j,c}$	binary variable: 1, if trip j is completed (or depot j is reached) with SOC c

Table 5.3.: Notation for the MIP.

We introduce binary variables $\beta_{j,c}$ to track the discrete SOC $c \in C$ at the end e_j of each trip $j \in J$ and when a vehicle reaches the final depot $j \in D$. Note that the number of variables $\beta_{j,c}$ grows pseudo-polynomially with $O(\hat{c} \cdot n)$. Moreover, we define set $N = \{(j, c) \in (J \cup D) \times C \mid c \neq \lfloor \zeta^* (j', j, c') \rfloor, \forall j' \in J \cup D, c' \in C\}$ as the set of pairs (j, c) indicating that trip j is never completed with SOC c in the optimal solution. Note that set N can be determined in

$O((n+d)^2 \cdot s \cdot \hat{c})$, by simply finding all combinations $j, j' \in J \cup D : j < j'$, $c \in C$, where $\max_{j'' \in S \cup \{0\}} \{\zeta(j, j', j'', c)\} = \zeta^*(j, j', c) < 0$.

Using these definitions, Equations (5.3) to (5.13) define a MIP model for the case that vehicles are not required to return to their initial depot at the end of their schedules.

$$\text{[EVSP-MD-FS] Minimize } F(\boldsymbol{x}, \boldsymbol{\beta}) = \sum_{j \in D} \sum_{j' \in J} x_{j,j'} \tag{5.3}$$

subject to

$$\sum_{\substack{j' \in J \cup D: \\ j' > j \vee j' \in D}} x_{j,j'} = 1 \qquad \forall j \in J \tag{5.4}$$

$$\sum_{\substack{j \in J \cup D: \\ j < j'}} x_{j,j'} = 1 \qquad \forall j' \in J \tag{5.5}$$

$$\sum_{\substack{c \in C: \\ (j,c) \notin N}} \beta_{j,c} = 1 \qquad \forall j \in J \tag{5.6}$$

$$0 \geq x_{j,j'} + \beta_{j,c} - 1 \qquad \forall j \in J, j' \in J \cup D : j < j' \vee j' \in D, \; c \in C : (j,c) \notin N \wedge \lfloor \zeta^*(j, j', c) \rfloor < 0 \tag{5.7}$$

$$\beta_{j', \lfloor \zeta^*(j,j',c) \rfloor} \geq x_{j,j'} + \beta_{j,c} - 1 \qquad \forall j \in J, j' \in J \cup D : j < j' \vee j' \in D, \; c \in C : (j,c) \notin N \wedge \lfloor \zeta^*(j, j', c) \rfloor \geq 0 \tag{5.8}$$

$$\beta_{j', \lfloor \zeta^*(j,j',\hat{c}) \rfloor} \geq x_{j,j'} \qquad \forall j \in D, j' \in J : \lfloor \zeta^*(j, j', \hat{c}) \rfloor \geq 0 \tag{5.9}$$

$$\sum_{j \in J} x_{j',j} \leq \bar{m}_{j'} \qquad \forall j' \in D \tag{5.10}$$

$$\sum_{j \in J'} x_{j,j'} \leq \bar{m}_{j'} \qquad \forall j' \in D \tag{5.11}$$

$$x_{j,j'} \in \{0,1\} \qquad \forall j, j' \in J \cup D : (j, j' \in J \wedge j < j') \vee (j \in J \wedge j' \in D) \vee (j \in D \wedge j' \in J \wedge \lfloor \zeta^*(j, j', \hat{c}) \rfloor \geq 0) \tag{5.12}$$

$$\beta_{j,c} \in \{0,1\} \qquad \forall j \in J \cup D, c \in C : (j,c) \notin N \tag{5.13}$$

Objective function (5.3) minimizes the number of required vehicles. Constraints (5.4) and (5.5) enforce that every trip has exactly one preceding and one succeeding trip or depot. Constraints (5.6) make sure that each trip is finished with exactly one non-negative SOC.

Constraints (5.7) to (5.9) ensure consistent SOC values. Inequalities (5.7) prohibit the execution of trip j' after trip j has been completed with SOC c if the SOC would be insufficient for completing the trip or time constraints would be violated (i.e., $\lfloor \zeta^* (j, j', c) \rfloor < 0$). Likewise, if the SOC is sufficient and no time constraints are violated (i.e., $\lfloor \zeta^* (j, j', c) \rfloor \geq 0$), Inequalities (5.8) set the SOC at the end of trip j' to the appropriate value. Constraints (5.9) determine the SOC of the trips that are initially executed after the vehicles leave the depots.

Constraints (5.10) and (5.11) enforce limits on the number of vehicles per depot. Finally, Constraints (5.12) and (5.13) define the domains of the decision variables. Note that – in order to reduce the search space – we omit $\beta_{j,c}$ for $(j, c) \in N$, since the states can never be feasibly reached.

Moreover, to formulate the MIP, we note that determining $\zeta^* (j, j', c)$ requires a pseudo-polynomial amount $O((n+d)^2 \cdot s \cdot \hat{c})$ of pre-calculations. Furthermore, the optimal deadheading routes need to be determined from the MIP's solution in a $O((n + d) \cdot s)$ post-processing procedure (cf., Section 5.3.4).

To enforce the optional constraint that each vehicle returns to the depot of its initial departure, we introduce the binary variables $v_{j,j'}$ that are 1 if trip j is associated with depot j', and 0 otherwise. Using these variables, we extend the MIP with the following equations:

$$\sum_{j' \in D} v_{j,j'} = 1 \qquad \forall j \in J \quad (5.14)$$

$$v_{j',j''} \geq x_{j,j'} + v_{j,j''} - 1 \qquad \forall j, j' \in J : j < j', j'' \in D \quad (5.15)$$

$$v_{j,j'} \geq x_{j',j} \qquad \forall j \in J, j' \in D \quad (5.16)$$

$$v_{j,j'} \geq x_{j,j'} \qquad \forall j \in J, j' \in D \quad (5.17)$$

$$v_{j,j'} \in \{0, 1\} \qquad \forall j \in J, j' \in D \quad (5.18)$$

Constraints (5.14) make sure that every trip $j \in J$ is associated with exactly one depot. Inequalities (5.15) force each pair of consecutively executed trips, j and j', to be associated with the same depot. Constraints (5.16) force the trip executed directly after leaving a depot to be associated with the respective depot. Likewise, Constraints (5.17) associate each final trip before returning to a depot with the respective depot. In conjunction, these constraints enforce each tour to start from and end at the same depot.

5.5. Branch-and-check

Our computational tests reveal that a default solver using the MIP model from Section 5.4 performs subpar for instances with $n \geq 100$ (see Section 5.6.2). We therefore propose a decomposition scheme based on branch-and-check (BCH, Thorsteinsson, 2001, Beck, 2010).

The general idea of BCH is to split a complicated problem into two parts, which are individually easier to solve: a mixed-integer programming master model, which is a relaxation of the original problem, and a subproblem, which encodes the remaining variables and/or constraints, albeit not necessarily in the form of a MIP. The master model is solved using classic branch-and-bound methods. Periodically, when it is advantageous, the subproblem is solved at a node of the branch-and-bound tree to check the feasibility of the current master solution and/or determine its exact objective value. This information is then communicated to the overarching branch-and-bound process, usually via cuts. The search terminates when there are no more unfathomed nodes in the enumeration tree left to explore. The best found solution is optimal.

For the EVSP-MD-FS, the flow is as follows. The master MIP model is concerned with assigning trips to vehicles, while relaxing the limited battery capacity. The feasibility and charging detours are determined by solving the subproblem. The master model is solved by a black-box default solver (CPLEX). Whenever the solver finds an integer solution during its branch-and-bound process, the solution is passed to the subproblem, which is solved by a polynomial-time algorithm and – in the case of limited depot capacities – by solving a (small) additional MIP. Information about the feasibility is injected into the branch-and-bound tree by way of combinatorial cuts, similar in form to logic-based (Hooker, 2011) or combinatorial (Codato and Fischetti, 2006) Benders cuts.

The master model is described in more detail in Section 5.5.1. The subproblem and its solution are presented in Section 5.5.2. Finally, the cuts derived from the subproblem are explained in Section 5.5.3.

5.5.1. Master problem

The purpose of the master model is assigning trips to vehicles. Charging breaks (and deadheading routes) are not scheduled at this stage. As in model [EVSP-MD-FS], we use variables $x_{j,j'}$ to denote whether two jobs, j and j', are processed consecutively by the same vehicle. Moreover $x_{0,j}$ and $x_{j,0}$ denote that a vehicle departs from any depot or returns to any depot, respectively.

The exact depots are determined later in the subproblem. Unlike before, we omit the binary variables $\beta_{j,c}$ and use continuous variables $\tilde{\beta}_j$, $\forall j \in J$, to calculate upper and lower bounds on the remaining SOC at the end of each trip.

Some trips can clearly not be assigned to the same vehicle due to either overlapping execution times or insufficient battery capacity. Therefore, we define the set of incompatible pairs of trips as

$$I = \left\{ (j, j') \in J \times J \mid \zeta^* \left(j, j', \max \left\{ \tilde{C}_j \right\} \right) < \min \left\{ \tilde{C}_{j'} \right\} \right\}, \tag{5.19}$$

where $\tilde{C}_j = \left\{ z \in \mathbb{R} \mid \min_{j' \in D \cup S} \left\{ \bar{c}_{j,j'} \right\} \leq z \leq \hat{c} - \tilde{c}_j - \min_{j' \in D \cup S} \left\{ \bar{c}_{j',j} \right\} \right\}$ is the set of feasible SOC values at the end of trip j, which takes into consideration that a vehicle must be able to feasibly reach and leave j. Note that it is possible that the vehicle executes another trip after trip j' before returning to a station or depot. However, the minimum required charge is guaranteed to be at least $\min_{j'' \in D \cup S} \{ \bar{c}_{j',j''} \}$ due to the triangle inequality holding true. In the case of insufficient time, function ζ^* will always assume -1. Otherwise, the maximum SOC that can remain at the end of trip j is $\max \left\{ \tilde{C}_j \right\}$. Hence, the maximum SOC at the end of trip j' is $\zeta^* \left(j, j', \max \left\{ \tilde{C}_j \right\} \right)$, which must be at least $\min \left\{ \tilde{C}_{j'} \right\}$ for the vehicle to feasibly continue its tour.

The master model is then given as follows.

$$\text{[MP] Minimize } \sum_{j \in J} x_{0,j} \tag{5.20}$$

subject to

$$\sum_{\substack{j' \in J \cup \{0\}: \\ (j,j') \notin I}} x_{j,j'} = 1 \qquad \forall j \in J \tag{5.21}$$

$$\sum_{\substack{j \in J \cup \{0\}: \\ (j,j') \notin I}} x_{j,j'} = 1 \qquad \forall j' \in J \tag{5.22}$$

$$\tilde{\beta}_{j'} \leq \tilde{\beta}_j + \bar{c}^{\max}_{j,j'} + M \cdot \left(1 - x_{j,j'}\right) \qquad \forall \in j, j' \in J : (j, j') \notin I \wedge \min \left\{ \tilde{C}_j \right\} + \bar{c}^{\max}_{j,j'} < \hat{c}^{\max}_{j,j'} \tag{5.23}$$

$$\tilde{\beta}_{j'} \leq \hat{c}^{\max}_{j,j'} + M \cdot \left(1 - x_{j,j'}\right) \qquad \forall j \in J \cup \{0\}, j' \in J : (j, j') \notin I \tag{5.24}$$

$$\tilde{\beta}_j \geq \sum_{\substack{j' \in J \cup \{0\}: \\ (j,j') \notin I \wedge \min\left(\tilde{C}_j\right) = \bar{c}^{\min}_{j,j'}}} \bar{c}^{\min}_{j,j'} \cdot x_{j,j'} \qquad \forall j \in J \quad (5.25)$$

$$x_{j,j'} \in \{0,1\} \qquad \forall j, j' \in J \cup \{0\} : (j,j') \notin \{0\}^2 \cup I \quad (5.26)$$

$$\tilde{\beta}_j \in \tilde{C}_j \qquad \forall j \in J \quad (5.27)$$

Objective function (5.20) minimizes the number of vehicles in use, i.e., the vehicles that initially depart from the depots. Constraints (5.21) and (5.22) enforce that each trip $j \in J$ has a predecessor and a successor, which can also be depots.

Constraints (5.23) and (5.24) calculate upper bounds on the SOC after trip j, $\forall j \in J$. For Constraints (5.23), we define the parameter $\bar{c}^{\max}_{j,j'}$ as the maximum increase in SOC between the end of trip j and the end of trip j'. If more charge is consumed than charged, $\bar{c}^{\max}_{j,j'}$ is negative. Formally, we define $\bar{c}^{\max}_{j,j'} = \max_{c \in \tilde{C}_j} \{\zeta^*(j,j',c) - c\}$. How to calculate $\bar{c}^{\max}_{j,j'}$ is explained in Appendix 5.7. By Constraints (5.23), an upper bound on the remaining SOC $\tilde{\beta}_{j'}$ after trip j' is $\bar{c}^{\max}_{j,j'}$ plus the upper bound of the SOC $\tilde{\beta}_j$ after the preceding trip j.

For Constraints (5.24), we define the parameter $\hat{c}^{\max}_{j,j'}$ as the maximum SOC at the end of trip j' if j and j' are processed consecutively. Formally, we define $\hat{c}^{\max}_{j,j'} = \zeta^*\left(j, j', \max\left\{\tilde{C}_j\right\}\right)$ and $\hat{c}^{\max}_{0,j'} = \max_{j \in D}\left\{\hat{c}^{\max}_{j,j'}\right\}$. Constraints (5.24) provide an additional upper bound by enforcing $\hat{c}^{\max}_{j,j'}$ as the maximum remaining charge at the end of trip j' if it is executed after trip j. Furthermore, if $\min\left\{\tilde{C}_j\right\} + \bar{c}^{\max}_{j,j'} \geq \hat{c}^{\max}_{j,j'}$ applies, the upper bound from Constraints (5.23) is guaranteed to be weaker than the respective one from Constraints (5.24), which is why we omit these in Constraints (5.23). Additionally, Constraints (5.24) provide upper bounds for the trips executed directly after the vehicles leave the initial depots. Finally, note that if two trips j and j' are not executed consecutively, $M \cdot \left(1 - x_{j,j'}\right)$ makes sure that no bound is forced upon $\tilde{\beta}_{j'}$. Since $0 \leq \tilde{\beta}_{j'} \leq \hat{c}$ and $\hat{c}^{\max}_{j,j'} \geq \bar{c}^{\max}_{j,j'} \geq -\hat{c}$, it is sufficient to set $M = 2 \cdot \hat{c}$.

To formulate lower bounds on the SOC, we define $\bar{c}^{\min}_{j,j'}$ as the charge that must remain at the end of trip j if trip j' is processed consecutively. Formally, we define $\bar{c}^{\min}_{j,j'} = \min\left\{c \in \tilde{C}_j \mid \zeta^*(j,j',c) \geq \min\left\{\tilde{C}_{j'}\right\}\right\}$ and $\bar{c}^{\min}_{j,0} = \min_{j' \in D}\left\{\bar{c}^{\min}_{j,j'}\right\}$. How to determine $\hat{c}^{\min}_{j,j'}$ is explained in Appendix 5.7. Note that the SOC must not only be sufficient to reach and process trip j', but also to reach a station or depot after that trip (cf., Equation (5.19)). Using this definition, Constraints (5.25) set a lower bound on the SOC at the end of trip j, $\forall j \in J$. Moreover, we can omit the pair j and j' in the sum, if $\min\left\{\tilde{C}_j\right\} = \bar{c}^{\min}_{j,j'}$ holds, because Constraints (5.27) already sets an equally strict bound on $\tilde{\beta}_j$.

Finally, Constraints (5.26) and (5.27) define the domains of the decision variables.

We solve model [MP] via a commercial black-box solver (CPLEX). Given a feasible integer candidate solution $\bar{\boldsymbol{x}}$, we derive the corresponding assignment of trips to vehicles with the following iterative procedure. We define $\hat{\omega}_k$ as the correspondent to ω_k (cf., Section 5.3.1) without scheduled depots at the beginning and the end, i.e., $\hat{\omega}_k = \omega_k \setminus D$. To distinguish $\hat{\omega}_k$ from ω_k, we refer to the former as a vehicle's working schedule in the following. Starting with $k = u = 1$, we calculate $\tilde{\omega}_k^u = \left\{ j \in J \mid x_{\max\{\tilde{\omega}_k^{u-1}\},j} = 1 \right\} \cup \tilde{\omega}_k^{u-1}$, where $\tilde{\omega}_k^0 = \left\{ \min_{j \in J \setminus \bigcap_{k'=1}^{k-1} \{\hat{\omega}_{k'}\}} \{j\} \right\}$. As long as $\tilde{\omega}_k^u \setminus \tilde{\omega}_k^{u-1} \neq \emptyset$, we increment u and repeat the calculation. Otherwise, we set $\hat{\omega}_k = \tilde{\omega}_k^u$, increment k, and reset $u = 1$ to start the procedure all over. If we reach $\tilde{\omega}_k^0 = \left\{ \min_{j \in J \setminus \bigcap_{k'=1}^{k-1} \{\tilde{\omega}_{k'}\}} \{j\} \right\} = \emptyset$, we terminate the procedure and set $m = k - 1$ for the fleet size.

The resulting assignments, i.e. $\hat{\omega}_k$, $\forall k = \{1, \ldots, m\}$, are, however, incomplete, because the actual depots to be visited at the beginning and in the end still need to be determined. Moreover, they may not allow a feasible solution for the original problem, because no charging breaks are scheduled and the battery of one or more vehicles may thus be insufficient. Consequently, we separate cuts and add them to model [MP] iteratively to converge on feasible and optimal solutions.

5.5.2. Subproblem

Given an integer candidate solution $\bar{\boldsymbol{x}}$ that is feasible for model [MP] and the assignments $\hat{\omega}_k$, $\forall k = 1, \ldots, m$, derived from it, we aim to answer two questions: First, which vehicle should depart from and return to which depot, and, second, when should which vehicle visit what station to charge, i.e., determine δ_k, $\forall k \in \{1, \ldots, m\}$?

The first question, i.e., determining the start and end depot for each working schedule $\hat{\omega}_k$, can be approached by simply trying all possibilities, where the start and end depot can be enforced to be identical if required. Let Ω_k be the set of all schedules ω_k for a given working schedule $\hat{\omega}_k$, where $\Omega_k(i)$ denotes the i-th schedule in Ω_k, corresponding to one specific pair of start ($\Omega_k(i)(1)$) and end ($\Omega_k(i)(|\Omega_k(i)|)$) depot. If the depots' capacities are unlimited, for a given working schedule $\hat{\omega}_k$, it is sufficient to find a single feasible start and a single feasible end depot, i.e., one feasible schedule $\omega_k \in \Omega_k$. We describe how to determine if a certain ω_k is feasible later in the section.

However, if the depots' capacities are limited, we must ensure that they are not exceeded, which we do as follows. Let $\hat{\Omega}_k \subseteq \Omega_k$ be the set of feasible schedules ω_k for a given working schedule

$\hat{\omega}_k$. Let $y_{i,k}$ be binary variables that are one if the i-th schedule in set $\hat{\Omega}_k$ is selected and zero otherwise. I.e., if $y_{i,k}$ assumes one, working schedule $\hat{\omega}_k$ is matched with depots $\hat{\Omega}_k(i)(1)$ and $\hat{\Omega}_k(i)\left(\left|\hat{\Omega}_k(i)\right|\right)$ as initial and final depot to form a proper schedule ω_k. Moreover, let z_k be binary variables that are one if no schedule in $\hat{\Omega}_k$ is selected and zero otherwise. I.e., if z_k assumes one, working schedule $\hat{\omega}_k$ has not been matched with an initial and final depot. Based on these definitions, we formulate the following integer program (IP):

$$\text{[Depot-Matching] Minimize } Z = \sum_{k \in \{1,\dots,m\}} z_k \tag{5.28}$$

subject to

$$\sum_{i \in \left\{1,\dots,|\hat{\Omega}_k|\right\}} y_{i,k} + z_k = 1 \qquad \forall k \in \{1,\dots,m\} \tag{5.29}$$

$$\sum_{k \in \{1,\dots,m\}} \sum_{\substack{i \in \left\{1,\dots,|\hat{\Omega}_k|\right\}: \\ \hat{\Omega}_k(i)(1)=j}} y_{i,k} \leq \bar{m}_j \qquad \forall j \in D \tag{5.30}$$

$$\sum_{k \in \{1,\dots,m\}} \sum_{\substack{i \in \left\{1,\dots,|\hat{\Omega}_k|\right\}: \\ \hat{\Omega}_k(i)(|\hat{\Omega}_k(i)|)=j}} y_{i,k} \leq \bar{m}_j \qquad \forall j \in D \tag{5.31}$$

$$y_{i,k} \in \{0,1\} \qquad \forall k \in \{1,\dots,m\}, i \in \left\{1,\dots,|\hat{\Omega}_k|\right\} \tag{5.32}$$

$$z_k \in \{0,1\} \qquad \forall k \in \{1,\dots,m\} \tag{5.33}$$

Objective (5.28) minimizes the number of vehicles where no feasible schedule could be selected, i.e., where no three-dimensional matching between a working schedule, the initial, and the final depot was found. If objective Z is zero, the solution is feasible for the original problem, i.e., there exists a combination of feasible schedules such that no depots' capacity is violated. Else, we need to add cuts to [MP], as we explain in Section 5.5.3.

Constraints (5.29) enforce that either exactly one or no schedule ω_k is selected for every vehicle k. Constraints (5.30) and (5.31) ensure the depots' capacities are not exceeded by the departing and returning vehicles, respectively. Finally, Constraints (5.32) and (5.33) define the domains of the decision variables. We solve [Depot-Matching] using a default solver.

Generally, [Depot-Matching] is NP-hard since it presents a three-dimensional matching problem. Nevertheless, it may still be solved quickly by a standard solver, due to the number of variables being in $O(m \cdot s^2)$ and usually not exceeding a couple of thousands in practical application. Moreover, if vehicles are required to return to their initial depots at the end of their

schedules, [Depot-Matching] reduces to a bipartite matching problem, which is known to be solvable in polynomial time.

Concerning the determination of charging breaks and detours, let ω_k be the i-th element in Ω_k. The optimal deadheading routes $\delta_k(l)$, $\forall l = 1, \ldots, |\omega_k| - 1$, can be derived in polynomial time by determining $\zeta^*(j, j', c)$ for every two consecutive elements, j and j', in ω_k and tracking the SOC c accordingly. More formally, we apply the following $|\omega_k|$-step iterative procedure. Starting from the initial state $l = 1$, for each state $l = 2, \ldots, |\omega_k|$, we calculate $\bar{\zeta}\,(k, l) = \zeta^*\big(\omega_k(l-1), \omega_k(l), \bar{\zeta}\,(k, l-1)\big)$, where $\bar{\zeta}(k, 1) = \hat{c}$ is the initial SOC. If we reach a state l, where $\bar{\zeta}(k, l) \leq \min\left\{\tilde{C}_{\omega_k(l)}\right\}$, the assignment ω_k cannot be solved in a feasible manner. In this case, we save the latest state $l^*_{k,i} := l$ to add cuts to the MP (cf., Section 5.5.3). Otherwise, if we reach state $l = |\omega_k|$ with $\bar{\zeta}(k, l) \geq 0$, a feasible tour has been found and we set $l^*_{k,i} := 0$. Furthermore, we save the solution to omit solving the subproblem again, if ω_k is part of a future solution to [MP].

The proposed procedure of determining the optimal deadheading routes for a given ω_k has at most $|\omega_k|$ states. In each state, the runtime for determining the optimal detour is asymptotically bounded by $O(s)$ (cf., Section 5.3.4), resulting in an asymptotic runtime of $O(|\omega_k| \cdot s)$ to solve the subproblem for a single schedule ω_k. This has to be repeated for all vehicles $k = 1, \ldots, m$ and all $\omega_k \in \Omega_k$. The number of vehicles m cannot reasonably be greater than the number of trips n. Furthermore, $\sum_{k \in \{1,\ldots,m\}} |\omega_k| \leq 3 \cdot n$ and $|\Omega_k| \leq \frac{d^2-d}{2}$ holds. Hence, the feasibility status, detours, and charging breaks for a given master solution can be determined in at most $O(n \cdot s \cdot d^2)$ time.

The idea of the above procedure can also be applied in a reverse manner. It is sufficient for the vehicle to return to the final depot with exactly zero SOC. Starting from the final depot, i.e. state $l = |\omega_k|$, we can derive the minimum sufficient SOC at a previous state $l - 1$ from the minimum sufficient SOC at state l. If we reach a state l where the minimum sufficient SOC exceeds $\hat{c}$, the assignment is infeasible, we save $l^{*,\,\text{rev}}_{k,i} = l$, and terminate the procedure. Otherwise, if we reach state $l = 0$, we set $l^{*,\,\text{rev}}_{k,i} := |\omega_k| + 1$. Clearly, concerning the feasibility of an assignment, the original and the reversed procedure yield the same result. However, $l^{*,\,\text{rev}}_{k,i}$ of the reverse procedure can be used for additional cuts if an assignment is infeasible. Hence, we only apply the reverse procedure, whenever the original procedure encounters an infeasible assignment.

5.5.3. Cuts

The subproblem may be infeasible in two distinctive ways, either due to the capacity limit of one or multiple depots being violated (i.e., $Z > 0$) or due to the working schedule $\hat{\omega}_k$ being impossible to execute feasibly for one or multiple vehicles $k = 1, \ldots, m$ (i.e., $\min_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^*_{i,k} \right\} > 0, \exists k = 1, \ldots, m$).

In the first case, we add the cut

$$1 \leq \sum_{j \in J} \sum_{\substack{j' \in J: \\ \bar{x}_{j,j'}=1}} \left(1 - x_{j,j'}\right)$$

to the model [MP], to which we refer as *depot feasibility* cut. The cut ensures that at least one working schedule $\hat{\omega}_k$ for one vehicle $k \in \{1, \ldots, m\}$ changes, making the current solution infeasible for model [MP] and thereby progressing the search.

In the second case, when there exists no feasible schedule ω_k for some vehicle k, we add so-called *schedule feasibility* cuts to model [MP] to exclude this infeasible solution and solutions sharing the same infeasible subset of consecutively executed trips from the search space. Formally, feasibility cuts are added for all $k \in \{1, \ldots, m \mid \min_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^*_{i,k} \right\} > 0\}$. Since $\max_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^*_{i,k} \right\}$ is the first trip that could not be feasibly served anymore for any choice of initial depot, at least one of the first $\max_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^*_{i,k} \right\}$ trips must change to attain feasibility. We ensure this by adding the cut

$$1 \leq \sum_{l=1}^{\max_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^*_{i,k} \right\} - 1} \left(1 - x_{\hat{\omega}_k(l), \hat{\omega}_k(l+1)}\right)$$

to the constraint set of model [MP]. Likewise, if it is not possible to execute the trips $\max_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^{*,\,\text{rev}}_{i,k} \right\}$ to $|\hat{\omega}_k|$ feasibly, we add the cut

$$1 \leq \sum_{l=\min_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^{*,\,\text{rev}}_{i,k} \right\} - 1}^{|\hat{\omega}_k| - 1} \left(1 - x_{\hat{\omega}_k(l), \hat{\omega}_k(l+1)}\right).$$

If $\min_{i \in \{1,\ldots,|\Omega_k|\}} \left\{ l^*_{i,k} \right\} = 0, \forall k = 1, \ldots, m$, and $Z = 0$ the solution is feasible for the original problem and is stored as the currently best known solution. We note that the black-box solver

passes every encountered integer solution that is feasible for model [MP] to be evaluated by solving the emerging subproblems. That means that even if the solution is feasible for the original problem, it is not necessarily optimal. Hence, a feasible solution presents merely an upper bound on [MP] and the original problem. Therefore, to expedite solving the master problem optimally, we add the *optimality* cut

$$\sum_{j \in J} x_{0,j} \leq m - 1 \tag{5.34}$$

to [MP]. As soon as the solution space of [MP] becomes empty, the last found solution that is feasible to the original problem is guaranteed to also be optimal and the procedure terminates.

5.6. Computational study

In this section, we test the computational performance of our proposed branch-and-check procedure and compare it to a default solver, namely CPLEX, solving the proposed [EVSP-MD-FS] MIP model. To do so, we generate randomized instances of various sizes and based on in-plant logistics. We describe the instance generation in the following section.

Furthermore, we derive some managerial insights into the influence of the battery capacity and charging mode on the fleet size. Finally, we investigate the influence of the number of warehouses and depots in the production facility.

5.6.1. Benchmark instances and computational environment

5.6.1.1. Generating instances for the EVSP-MD-FS based on an in-plant milk-run logistics setting

We model the instances for EVSP-MD-FS based on an in-plant milk-run logistics setting in accordance with the one regarded by Emde et al. (2018). As shown in Figure 5.1a, we assume a facility where the production area and the warehouses are separated. This need not always be the case, but is a reasonable assumption for a lot of practical cases. We implement this assumption by placing the warehouses at random locations within a $1000\ \text{m} \times 1000\ \text{m}$ area that represents the facility, where the production area is located at coordinates $(1000\ \text{m}, 500\ \text{m})$, i.e., centrally at the right border of the facility. All warehouses are assumed to have an internal size

of 200 m $\times$ 200 m and the production area is set to 400 m $\times$ 400 m.

We further assume that the tow train depots are located within the warehouses and that all depots, and exclusively depots, can be used as charging stations. As described in Section 5.1, a trip starts in a warehouse, where the tow train is loaded with the materials required at the production stations. The tow train then sets off to the production area and visits a set of predetermined assembly cells to deliver the required goods. Once the final assembly cell has been supplied, the tow train is ready to process the next trip or return to a depot. This is modeled by placing the depots and the trips' start points at random locations within the warehouses, where each warehouse can host at most one depot. The trips' end points are set to random locations within the production area.

The proposed branch-and-check approach can handle real-valued parameters. However, the [EVSP-MD-FS] model requires all parameters to be integer to yield an exact result, which is why we round parameters to integer values in the following. Considering time parameters, we use a precision of half-minutes. I.e., an integer value of 1 represents half a minute. The trips' start times are determined according to Emde et al. (2018). The first trip is set so start at $s_1 = 10$ half-minutes, such that the trip's start can be reached by a tow train departing from a depot no sooner than time 0. The starting times of all consecutive trips are set iteratively according to $s_j = s_{j-1} + rnd([0, 10])$ half-minutes, where $rnd([0, 10])$ is a random integer from the interval $[0, 10]$, which results in close to real-world timetables (Emde et al., 2018). Hence, on average, there are 24 tips per hour, which – for comparison – falls in the upper quarter of trips per hour reported for tow train systems in the German industry (cf., Lieb et al., 2017). The trips' execution times are set to random integers drawn from the interval $[20, 50]$ half-minutes, since those are typical trip durations in practice. A trip's end time is its start time plus its execution time.

To determine realistic time requirements for the detours, we first calculate respective distances applying the rectilinear metric. The total distance between two locations (e.g., the end point of a trip and the location of a depot) is set to be the sum of the internal distance (i.e., the distance within the warehouse and production area) and the external distance (i.e., the distance between warehouses and the production area), which is in line with our assumed facility layout. The travel times for all detours are then calculated by dividing the respective distances by an assumed travel speed of 10 kph and rounding to the nearest half-minute value, where 10 kph is a representative speed for tow trains.

We state all battery-related values as percentages of the maximum charge and use a precision of half-percentages. I.e., an integer value of 1 represents half a percentage of the maximum SOC. The trips' battery requirements may not be strictly proportional to their distance or execu-

tion time, since each trip requires a different number of stops at assembly stations and the tow train may need to carry different loads on every trip. To account for this fact, we determine the trips' battery requirements in the following way. First, we associate the minimum trip duration of 20 half-minutes with a battery requirement of 12.5% and the maximum trip duration of 50 half-minutes with 30%, which is based on the technical data of a representative tow train (cf., Still, 2019). Second, we interpolate each trip's preliminary battery requirement based on its duration. Third, we randomize the battery requirement by multiplying the preliminary battery requirement with a random factor from the interval $[0.75, 1.25]$ and round it to the nearest half-percentage value. While processing detours, the tow train is mostly empty and does not need to stop frequently. Hence, here, distance, execution time and battery requirement are proportional in good approximation. Therefore, we calculated the detours' battery requirements by multiplying their respective distances with the factor $5\frac{\%}{\text{km}}$, which, again, is derived from a representative tow train's technical data (cf., Still, 2019).

Using the proposed scheme, we generate ten instances of each size $n \in \{50, 100, 200\}$ with two, three, and five warehouses, respectively, where every warehouse also hosts a depot. Note that with $n = 200$, an instance covers roughly an eight-hour workday. Moreover, note that at this point, we do not make any assumptions regarding the way in which the batteries are charged. This is discussed in Section 5.6.1.2. All instances are named according to the scheme "I-s-d-n-##", where "##" is a continuous counting number.

5.6.1.2. Modeling (piecewise) non-linear and constant-time battery charging

Both the [EVSP-MD-FS] model and the proposed branch-and-check approach are able to handle various battery charging functions. In our computational study, we consider two ways of realistic battery charging, namely constant-time battery charging, e.g., battery swapping, and (piecewise) concave non-linear charging as common for lithium-ion batteries, which are frequently used for powering electric vehicles in intra-logistic applications.

As stated in Section 5.3.1, battery swapping (or other means of constant-time battery charging) can be modeled by setting r, the time between the arrival of the vehicle at the charging station and the start of the actual charging, to the time it takes to swap the battery. The battery charging function is $\eta(\tau) = \hat{c}$ in this case. Based on our observations, the time it takes to swap a tow train's battery can vary depending on the skills of the worker, the setup of the charging station and, most importantly, the exact model of the tow train. Nevertheless, we found three minutes to be a representative value, for a broad variety of cases. Hence, we set $r = 6$ half-minutes.

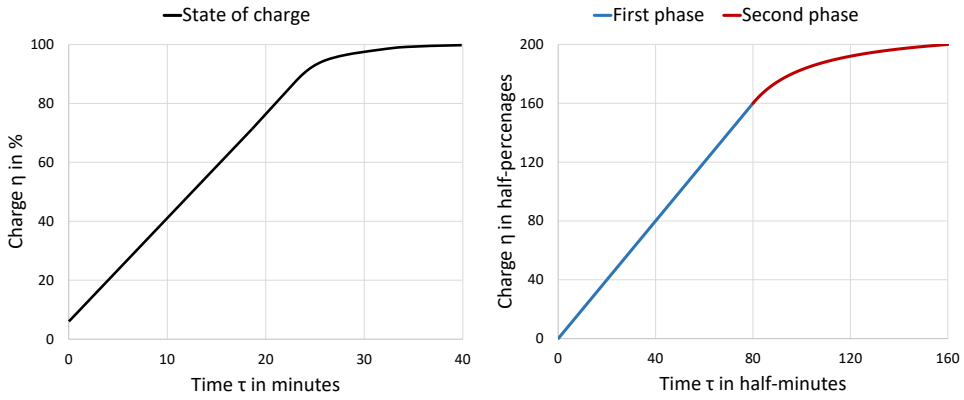
Lithium-ion batteries are a common type of batteries frequently used to power electric vehicles (Kisacikoglu et al., 2011, Pelletier et al., 2017). Their charging function is (piecewise) non-linear, concave, and usually consists of two phases, as exemplarily depicted in Figure 5.3a and explained in Section 5.1. To briefly recap, first, between 0% of total charge up to about 75% to 90%, depending on the exact battery type, charging is linear with time τ. In the second phase, the charging rate continuously depreciates, which results in a charging function that gradually approaches 100% over time. The transition between both phases is continuous. Depending on the exact type of battery, the first phase takes about 25% to 75% of the total charging time and the second phase takes between 75% to 25% of the time (Kisacikoglu et al., 2011, Pelletier et al., 2017). In the following, we assume that both phases take an equal duration and linear charging ends at 80% of total charge. Furthermore, we approximate the charging function in the second phase with a function of the general shape of $\eta^{\text{second phase}}(\tau) = k_1 + \frac{k_2}{k_3+x}$, where k_1, k_2 and k_3 are adjustable factors. Given these assumption and the battery's charging rate in the first phase, we can determine the factors k_1, k_2 and k_3 by fitting $\eta^{\text{second phase}}(\tau)$ to the end of the first phase with a continuous transition from the linear charging rate. We assume a battery charging rate in the linear phase of $2\frac{\%}{\text{min}}$, which is about an average value for most tow-trains (cf. Emde et al., 2018). This results in the battery charging function

$$\eta(\tau) = \begin{cases} 2 \cdot \tau & \text{if } \tau \leq 80 \\ \frac{640}{3} - \frac{\frac{12800}{9}}{\tau - \frac{160}{3}} & \text{if } 80 < \tau \leq 160 \\ 200 & \text{if } 160 < \tau \end{cases},$$

where τ is in the unit of half-minutes and $\eta(\tau)$ is in the unit of half-percentages. The charging function is depicted in Figure 5.3b. Furthermore, we assume the delay between the arrival at the station and the beginning of the charging procedure to be $r = 1$ half-minute.

5.6.2. Computational results

In this section, we present our computational results. All testing was performed on an Intel Core i7-6700 CPU @ 3.40 Gigahertz and with 16 Gigabyte of RAM. All algorithms were implemented in C# and CPLEX (version 12.10) was used as default solver (for the [EVSP-MD-FS] MIP as well as the [MP] master model and [Depot-Matching] subproblem model of the proposed branch-and-check procedure) at default settings. Furthermore, we limited the runtime to solve a single instance to 3600 seconds (i.e., 1 h). The results of the tests are summarized in the



(a) Exemplary lithium-ion battery charging function according to Kisacikoglu et al. (2011).

(b) Tow train battery charging function.

Figure 5.3.: Non-linear battery charging functions.

following. Detailed reports on the results as well as the original instances are provided as online supplementary material at `https://doi.org/10.5281/zenodo.6406201`.

5.6.2.1. Performance evaluation

We solved all instances with CPLEX and with the proposed BCH approach. The results are summarized in Table 5.4. Generally, CPLEX performed better for constant-time charging than for non-linear charging. It was able to solve all instances with $n \leq 100$ to optimality for both types of battery charging within a runtime of 3600 s. For instances with $n = 200$, CPLEX was able to find solutions, which were not always optimal, however. Especially for non-linear battery charging, the respective optimality gaps were rather large with 67 % on average.

The proposed BCH procedure clearly outperformed CPLEX. For constant-time charging, the BCH procedure solved all instances in at most 2.6 s. For non-linear charging, the BCH procedure solved all instances with $n = 50$ in below 0.1 s and every instance with $n = 100$ in no more than 5.1 s. While the BCH procedure could solve all T-3-3-200-## instances in 364.8 s on average, one instance required 3418.4 s to be solved while the other instances were solved much faster.

Our results clearly indicate that instances were easier to solve if constant-time charging was assumed, especially if instances were large. We attribute this to constant-time charging being generally quicker than non-linear charging, which makes it easier to find feasible solutions for the former. This becomes evident when the “s. f. cut” column in Tables 5.4 is considered. It

instance	non-linear battery recharging											constant-time battery recharging										
	CPLEX			branch-and-check								CPLEX			branch-and-check							
lable	value	LB	time (in s)	value	LB	time (in s)	time MP (in %)	time SP (in %)	s. inf. cuts	d. inf. cuts	opt. cuts	value	LB	time (in s)	value	LB	time (in s)	time MP (in %)	time SP (in %)	s. inf. cuts	d. inf. cuts	opt. cuts
I-2-2-50-01	13	13	9.5	13	13	0.1	89.1	10.9	1	0	1	13	13	7.3	13	13	0.0	100.0	0.0	0	0	1
I-2-2-50-02	12	12	11.4	12	12	0.1	86.3	13.7	11	0	1	12	12	6.9	12	12	0.0	100.0	0.0	0	0	1
I-2-2-50-03	12	12	9.4	12	12	0.0	97.6	2.4	4	0	1	12	12	8.2	12	12	0.0	100.0	0.0	0	0	1
I-2-2-50-04	17	17	8.6	17	17	0.0	93.8	6.3	0	0	1	17	17	6.1	17	17	0.0	93.8	6.3	0	0	1
I-2-2-50-05	15	15	8.2	15	15	0.0	97.1	2.9	2	0	1	15	15	6.4	15	15	0.0	94.7	5.3	1	0	1
I-2-2-50-06	13	13	9.2	13	13	0.0	94.6	5.4	4	0	1	13	13	6.2	13	13	0.0	100.0	0.0	0	0	1
I-2-2-50-07	16	16	8.8	16	16	0.0	100.0	0.0	0	0	1	16	16	6.1	16	16	0.0	100.0	0.0	0	0	1
I-2-2-50-08	12	12	9.8	12	12	0.0	95.1	4.9	8	0	1	12	12	6.3	12	12	0.0	100.0	0.0	0	0	1
I-2-2-50-09	16	16	8.3	16	16	0.0	95.0	5.0	1	0	1	16	16	6.6	16	16	0.0	93.8	6.3	0	0	1
I-2-2-50-10	13	13	11.5	13	13	0.0	95.0	5.0	0	0	1	13	13	6.2	13	13	0.0	100.0	0.0	0	0	1
mean	**13.9**	**13.9**	**9.5**	**13.9**	**13.9**	**0.0**	**94.4**	**5.6**	**3.1**	**0.0**	**1.0**	**13.9**	**13.9**	**6.6**	**13.9**	**13.9**	**0.0**	**98.2**	**1.8**	**0.1**	**0.0**	**1.0**
I-3-3-100-01	15	15	342.3	15	15	3.1	68.2	31.8	538	0	4	15	15	66.7	15	15	0.1	94.3	5.8	0	0	1
I-3-3-100-02	17	17	76.8	17	17	0.2	75.3	24.7	13	0	1	17	17	85.7	17	17	0.1	89.0	11.0	0	0	1
I-3-3-100-03	16	16	200.7	16	16	0.3	75.1	24.9	13	0	1	16	16	75.9	16	16	0.2	91.1	8.9	1	0	1
I-3-3-100-04	16	16	265.7	16	16	0.3	75.4	24.6	17	0	1	16	16	54.4	16	16	0.1	85.1	14.9	0	0	2
I-3-3-100-05	18	18	450.7	18	18	5.1	64.5	35.5	824	0	3	18	18	39.8	18	18	0.1	90.9	9.1	1	0	1
I-3-3-100-06	15	15	1464.3	15	15	4.1	68.6	31.4	824	0	4	15	15	138.8	15	15	0.1	93.1	6.9	0	0	1
I-3-3-100-07	14	14	3597.4	14	14	0.8	68.4	31.6	77	0	3	14	14	99.0	14	14	0.1	94.4	5.6	0	0	1
I-3-3-100-08	15	15	478.3	15	15	1.8	59.3	40.8	204	0	4	15	15	146.7	15	15	0.1	91.0	9.0	2	0	1
I-3-3-100-09	19	19	117.4	19	19	0.2	65.4	34.6	8	0	1	19	19	26.2	19	19	0.1	87.7	12.4	0	0	1
I-3-3-100-10	19	19	267.7	19	19	0.4	63.1	36.9	21	0	4	19	19	42.7	19	19	0.1	89.4	10.6	1	0	1
mean	**16.4**	**16.4**	**726.1**	**16.4**	**16.4**	**1.6**	**68.3**	**31.7**	**253.9**	**0.0**	**2.6**	**16.4**	**16.4**	**77.6**	**16.4**	**16.4**	**0.1**	**90.6**	**9.4**	**0.5**	**0.0**	**1.1**
I-5-5-200-01	20	18	3600.0	18	18	7.2	53.7	46.4	96	0	3	19	18	3600.0	18	18	0.4	84.1	15.9	0	0	1
I-5-5-200-02	36	17	3600.0	17	17	3418.4	86.4	13.6	38018	0	4	17	17	1039.2	17	17	0.8	87.4	12.6	1	0	1
I-5-5-200-03	37	15	3600.0	15	15	1.7	58.0	42.0	34	0	1	15	15	2913.9	15	15	0.4	87.7	12.3	0	0	1
I-5-5-200-04	17	16	3600.0	16	16	1.3	70.2	29.8	10	0	1	16	16	1750.3	16	16	2.6	42.4	57.6	10	0	3
I-5-5-200-05	18	18	1439.9	18	18	8.8	47.1	52.9	76	0	3	18	18	1750.9	18	18	0.9	82.5	17.5	1	0	1
I-5-5-200-06	16	16	1904.1	16	16	1.1	82.6	17.4	5	0	1	16	16	1037.1	16	16	0.4	84.9	15.1	0	0	1
I-5-5-200-07	31	16	3600.0	16	16	5.8	63.5	36.5	54	0	3	16	16	344.5	16	16	0.4	85.3	14.7	0	0	1
I-5-5-200-08	47	16	3600.0	16	16	2.7	56.2	43.8	34	0	2	16	16	2121.4	16	16	0.4	85.7	14.4	0	0	1
I-5-5-200-09	18	18	1365.1	18	18	191.7	45.7	54.3	4331	0	4	18	18	2250.0	18	18	0.8	91.5	8.5	0	0	1
I-5-5-200-10	41	18	3600.0	18	18	9.7	52.2	47.8	156	0	3	18	18	1486.8	18	18	0.4	85.0	15.0	0	0	1
mean	**28.1**	**16.8**	**2990.9**	**16.8**	**16.8**	**364.8**	**61.6**	**38.4**	**4281.4**	**0.0**	**2.5**	**16.9**	**16.8**	**1829.4**	**16.8**	**16.8**	**0.8**	**81.6**	**18.4**	**1.2**	**0.0**	**1.2**

LB = lower bound at the point of termination; s./d. inf. cuts = number of added schedule/depot infeasibility cuts; opt. cuts = number of added optimality cuts; time MP/SP = relative runtime of the BCH master problem/of the BCH subproblem including cut generation

Table 5.4.: Computational test on the I-$|S|$-$|D|$-n-## instances.

indicates an instance's hardness by stating the number of times the MP's solution was infeasible (due to violated battery constraints) such that one or multiple schedule feasibility cuts had to be added to the MP. The observation is also supported by the results in Section 5.6.2.4, where instances with less battery capacity were harder to solve.

5.6.2.2. Performance on instances with additional constraints

As described in Section 5.3, we also consider optional constrains for EVSP-MD-FS, namely that vehicles must return to their depot of initial departure and that the depots' capacities are limited. While these constraints are not currently relevant in the production plants we visited, they are often discussed in the VSP literature and may also play a role in intra-logistics, especially in such cases where the depots are small and vehicles are fixedly assigned to them. To evaluate the effects of these constraints, we generated constrained instances from regular instances as follows. From each regular instance in Table 5.4, we constructed a constrained instance with the same time and battery parameters. To obtain reasonable capacity limits for the depots, we took the regular instance's optimal objective value, multiplied it by a factor of 1.25, rounded the result up, and split it into integer values that we distributed randomly between the depots, where we made sure that each depot has at least a capacity of 1. Moreover, we made the instance require that vehicles must return to their depot of initial departure. Each constrained instance is labeled according to the regular instance it is based on except for an initial "C" instead of "I".

We solved all constrained instances with CPLEX and with the proposed BCH approach. The results are summarized in Table 5.5.

instance	non-linear battery recharging										constant-time battery recharging									
	CPLEX			branch-and-check							CPLEX			branch-and-check						
lable	value	LB	time (in s)	value	LB	time (in s)	time MP (in %)	time SP (in %)	s. inf. cuts	d. inf. cuts	opt. cuts	value	LB	time (in s)	value	LB	time (in s)	time MP (in %)	time SP (in %)	s. inf. cuts	d. inf. cuts	opt. cuts
C-2-2-50-01	13	13	10.2	13	13	0.0	81.6	18.4	1	0	1	13	13	7.1	13	13	0.0	82.6	17.4	0	0	1
C-2-2-50-02	12	12	13.1	12	12	0.0	83.3	16.7	11	0	1	12	12	9.0	12	12	0.0	72.0	28.0	0	0	1
C-2-2-50-03	12	12	11.0	12	12	0.0	78.4	21.6	4	0	1	12	12	7.7	12	12	0.0	73.7	26.3	0	0	1
C-2-2-50-04	17	17	8.7	17	17	0.0	79.0	21.1	0	0	1	17	17	6.3	17	17	0.0	79.0	21.1	0	0	1
C-2-2-50-05	15	15	8.8	15	15	0.0	85.7	14.3	2	0	1	15	15	6.2	15	15	0.0	77.3	22.7	1	0	1
C-2-2-50-06	13	13	9.3	13	13	0.0	83.3	16.7	4	0	1	13	13	6.6	13	13	0.0	76.2	23.8	0	0	1
C-2-2-50-07	16	16	8.7	16	16	0.0	79.0	21.1	0	0	1	16	16	6.3	16	16	0.0	76.2	23.8	0	0	1
C-2-2-50-08	12	12	10.4	12	12	0.0	84.2	15.8	8	0	1	12	12	8.4	12	12	0.0	83.3	16.7	0	0	1
C-2-2-50-09	16	16	9.4	16	16	0.0	71.4	28.6	1	0	1	16	16	6.0	16	16	0.0	75.0	25.0	0	0	1
C-2-2-50-10	13	13	12.4	13	13	0.0	77.3	22.7	0	0	1	13	13	6.5	13	13	0.0	77.3	22.7	0	0	1
mean	**13.9**	**13.9**	**10.2**	**13.9**	**13.9**	**0.0**	**80.3**	**19.7**	**3.1**	**0.0**	**1.0**	**13.9**	**13.9**	**7.0**	**13.9**	**13.9**	**0.0**	**77.2**	**22.8**	**0.1**	**0.0**	**1.0**
C-3-3-100-01	15	15	1450.1	15	15	4.8	65.1	34.9	987	2	3	15	15	48.2	15	15	0.1	88.0	12.1	0	0	1
C-3-3-100-02	17	17	818.6	17	17	0.2	68.2	31.8	13	0	1	17	17	54.4	17	17	0.1	86.6	13.4	0	0	1
C-3-3-100-03	16	16	272.6	16	16	0.3	73.4	26.6	13	0	1	16	16	48.3	16	16	0.1	86.6	13.4	1	0	1
C-3-3-100-04	16	16	1620.2	16	16	0.2	70.3	29.7	17	0	1	16	16	87.9	16	16	0.1	75.0	25.0	0	0	2
C-3-3-100-05	18	18	1008.9	18	18	4.7	63.8	36.2	824	0	3	18	18	98.5	18	18	0.1	87.5	12.5	1	0	1
C-3-3-100-06	15	15	988.0	15	15	1.9	62.2	37.8	374	2	1	15	15	65.1	15	15	0.1	88.1	11.9	0	0	1
C-3-3-100-07	14	14	1766.6	14	14	0.9	57.1	42.9	98	2	1	14	14	52.3	14	14	0.1	85.7	14.3	0	0	1
C-3-3-100-08	15	15	1738.5	15	15	8.0	61.0	39.0	1140	2	3	15	15	114.5	15	15	0.1	87.3	12.7	2	0	1
C-3-3-100-09	19	19	945.1	19	19	0.2	66.7	33.3	8	0	1	19	19	44.1	19	19	0.1	84.6	15.4	0	0	1
C-3-3-100-10	19	19	272.5	19	19	0.4	60.6	39.4	18	2	1	19	19	28.5	19	19	0.1	86.1	13.9	1	0	1
mean	**16.4**	**16.4**	**1088.1**	**16.4**	**16.4**	**2.2**	**64.8**	**35.2**	**349.2**	**1.0**	**1.6**	**16.4**	**16.4**	**64.2**	**16.4**	**16.4**	**0.1**	**85.6**	**14.4**	**0.5**	**0.0**	**1.1**
C-5-5-200-01	-	18	3600.0	18	18	13.2	49.1	50.9	163	2	1	23	18	3600.0	18	18	0.4	83.5	16.5	0	0	1
C-5-5-200-02	-	17	3600.0	18	17	3600.0	90.4	9.6	26417	2	2	23	17	3600.0	17	17	0.8	85.8	14.2	1	0	1
C-5-5-200-03	-	15	3600.0	15	15	1.8	59.7	40.4	34	0	1	-	15	3600.0	15	15	0.4	85.4	14.7	0	0	1
C-5-5-200-04	-	16	3600.0	16	16	1.3	70.5	29.5	10	0	1	-	16	3600.0	16	16	2.6	40.8	59.2	10	2	1
C-5-5-200-05	-	18	3600.0	18	18	36.3	38.1	61.9	509	2	2	22	18	3600.0	18	18	0.9	83.1	16.9	1	0	1
C-5-5-200-06	-	16	3600.0	16	16	1.1	81.9	18.1	5	0	1	-	16	3600.0	16	16	0.4	84.3	15.7	0	0	1
C-5-5-200-07	-	16	3600.0	16	16	11.2	58.5	41.5	139	2	1	21	16	3600.0	16	16	0.4	83.5	16.5	0	0	1
C-5-5-200-08	-	16	3600.0	16	16	2.8	56.5	43.5	34	0	2	19	16	3600.0	16	16	0.4	85.1	14.9	0	0	1
C-5-5-200-09	-	18	3600.0	18	18	17.5	41.0	59.0	273	2	2	23	18	3600.0	18	18	0.8	89.9	10.1	0	0	1
C-5-5-200-10	-	18	3600.0	18	18	15.6	54.3	45.7	285	2	1	24	18	3600.0	18	18	0.4	83.5	16.5	0	0	1
mean	**-**	**16.8**	**3600.0**	**16.9**	**16.8**	**370.1**	**60.0**	**40.0**	**2786.9**	**1.2**	**1.4**	**22.1**	**16.8**	**3600.0**	**16.8**	**16.8**	**0.8**	**80.5**	**19.5**	**1.2**	**0.2**	**1.0**

LB = lower bound at the point of termination; s./d. inf. cuts = number of added schedule/depot infeasibility cuts; opt. cuts = number of added optimality cuts; time MP/SP = relative runtime of the BCH master problem/of the BCH subproblem including cut generation; - = no feasible solution was found within the runtime

Table 5.5.: Computational test on the C-$|S|$-$|D|$-n-## instances.

The results indicate that the constrained instances were generally harder to solve for both CPLEX and – to a lesser degree – the proposed BCH approach. Especially for the former, required runtimes increase significantly such that for $n = 200$ and non-linear charging, CPLEX did not find any solution at all within the runtime limit. On the other hand, runtimes only increased slightly for the BCH approach, such that all instances except for instances C-5-5-200-02 at non-linear charging were solved optimally within the runtime limit. This is in line with the rather low number of applied depot feasibility cuts (cf., the "d. f. cut" columns in Table 5.5), which never exceeded two.

5.6.2.3. Performance on large instances

In Section 5.6.2.1, we considered instances with up to $n = 200$ trips, which cover roughly an eight-hour shift. To test the performance of our solution method on even larger instances and see how it scales from a theoretical perspective, we apply the scheme described in Section 5.6.1.1 to randomly generate instances with $s = d = 5$ as well as 1000, 2000, 3000, and 4000 trips, which is well beyond the number of trips we expect in practical intra-logistic applications.

We tried to solve the respective instances by applying CPLEX to the MIP. However, during initialization of the model, CPLEX always reported an out of memory error. For the BCH

approach, we set a runtime limit of 24 h (86400 s). The results are given in Table 5.6.

instance	non-linear charging								constant-time charging							
lable	value	LB	time (in s)	time MP (in %)	time SP (in %)	s. inf. cuts	d. inf. cuts	opt. cuts	value	LB	time (in s)	time MP (in %)	time SP (in %)	s. inf. cuts	d. inf. cuts	opt. cuts
I-5-5-1000-01	291	18	86400.0	47.8	52.2	61044	0	2	18	18	46.6	78.1	21.9	0	0	1
I-5-5-1000-02	291	17	86400.0	62.5	37.5	51369	0	2	17	17	86.0	61.3	38.7	5	0	1
I-5-5-1000-03	298	19	86400.0	39.3	60.7	57756	0	2	19	19	46.0	77.5	22.5	0	0	1
I-5-5-1000-04	295	18	86400.0	45.6	54.5	69618	0	2	18	18	65.2	58.2	41.8	2	0	1
I-5-5-1000-05	284	18	86400.0	40.3	59.7	69693	0	2	18	18	37.2	68.1	31.9	0	0	1
mean	**291.8**	**18.0**	**86400.0**	**47.1**	**52.9**	**61896.0**	**0.0**	**2.0**	**18.0**	**18.0**	**56.2**	**68.6**	**31.4**	**1.4**	**0.0**	**1.0**
I-5-5-2000-01	596	16	86400.0	54.9	45.1	2206	0	2	16	16	1043.0	59.1	41.0	3	0	1
I-5-5-2000-02	583	19	86400.0	45.6	54.4	4316	0	2	19	19	2777.2	33.4	66.7	10	0	2
I-5-5-2000-03	577	18	86400.0	60.6	39.5	3158	0	2	18	18	4766.5	32.3	67.7	23	0	2
I-5-5-2000-04	576	21	86400.0	28.7	71.3	4860	0	2	21	21	263.3	49.4	50.6	0	0	1
I-5-5-2000-05	578	18	86400.0	59.9	40.1	3616	0	2	18	18	665.2	84.6	15.4	0	0	1
mean	**582.0**	**18.4**	**86400.0**	**49.9**	**50.1**	**3631.2**	**0.0**	**2.0**	**18.4**	**18.4**	**1903.1**	**51.8**	**48.2**	**7.2**	**0.0**	**1.4**
I-5-5-3000-01	877	21	86400.0	9.8	90.3	46	0	2	21	21	2174.5	77.3	22.7	0	0	1
I-5-5-3000-02	872	18	86400.0	13.6	86.4	47	0	2	18	18	1714.9	77.0	23.1	0	0	1
I-5-5-3000-03	876	20	86400.0	16.9	83.1	60	0	2	20	20	2943.9	84.9	15.1	0	0	1
I-5-5-3000-04	883	19	86400.0	12.1	87.9	86	0	2	19	19	2239.4	64.9	35.1	2	0	1
I-5-5-3000-05	879	22	86400.0	11.4	88.6	54	0	2	22	22	1132.8	50.0	50.0	0	0	1
mean	**877.4**	**20.0**	**86400.0**	**12.8**	**87.2**	**58.6**	**0.0**	**2.0**	**20.0**	**20.0**	**2041.1**	**70.8**	**29.2**	**0.4**	**0**	**1**
I-5-5-4000-01	4000	18	86400.0	7.1	92.9	19	0	1	18	18	4211.9	78.2	21.8	0	0	1
I-5-5-4000-02	4000	19	86400.0	10.5	89.5	19	0	1	20	19	86400.0	54.7	45.3	348	0	2
I-5-5-4000-03	4000	17	86400.0	9.7	90.3	36	0	1	17	17	7106.2	89.1	10.9	0	0	1
I-5-5-4000-04	4000	20	86400.0	5.2	94.8	6	0	1	20	20	4857.9	53.9	46.1	1	0	1
I-5-5-4000-05	4000	19	86400.0	10.0	90.0	6	0	1	19	19	12185.7	67.0	33.0	3	0	1
mean	**4000.0**	**18.6**	**86400.0**	**8.5**	**91.5**	**17.2**	**0.0**	**1.0**	**18.8**	**18.6**	**22952.3**	**68.6**	**31.4**	**70.4**	**0.0**	**1.2**

LB = lower bound at the point of termination; s./d. inf. cuts = number of added schedule/depot infeasibility cuts; opt. cuts = number of added optimality cuts; time MP/SP = relative runtime of the BCH master problem/of the BCH subproblem including cut generation

Table 5.6.: Computational test on large instances.

For non-linear charging, only exceptionally bad solutions were found within the runtime limit. We explain this as follows. At the beginning of the procedure, CPLEX, which is used to solve the MP in the BCH approach, forwards rather bad solutions to the subproblem, which (unsurprisingly) are validated to be feasible. Afterwards, CPLEX repeatedly forwards solutions with objective values equaling the lower bound to the subproblem, which are found to be infeasible, however. As a consequence, no close-to-optimal solutions are explored within the runtime and the best found solutions remain at a bad objective value.

On the other hand, for constant-time charging, the BCH approach was able to find optimal solutions for all instances except instance I-5-5-4000-2, where the optimality gap is only 1. This is quite a remarkable result for an exact solution approach and indicates that our approach may be quite suitable in practice even for very large instances if battery-swapping technology is used.

Note that, as mentioned in Section 5.6.2, all computational tests were performed on a desktop PC with 16 GB of RAM available. However, for the larger instances, the BCH procedure required far greater memory, which sometimes even exceeded 50 GB during processing. Hence, hard drive memory was used, which slowed down processing significantly. For example, for the I-5-5-4000-## instances, reading out an integer candidate solution $\bar{x}$ from the MP required more than half an hour, even though the number of variables is bounded by $O(n^2)$. Since we attribute this time to the subproblem's runtime, it also explains the large proportion of runtime spent solving the subproblem in Table 5.6. Hence, if the BCH would be executed on a machine with

sufficient RAM, we expect further performance increases.

5.6.2.4. Influence of the battery capacity and charging mode

This section investigates the influence of the battery capacity and the charging mode on the number of required vehicles. We examined the effects of increasing and decreasing the battery capacity by 10%, 20% and 30% and compared it to the status quo. Changing the battery capacity requires adjusting the battery charging function $\eta(\tau)$, which we did by simply multiplying its output by the same factor. We performed the experiment for all instances with $n = 100$. The results of the experiment are summarized in Table 5.7 and Figure 5.4.

instance	non-linear with a capacity of							constant-time with a capacity of						
label	70%	80%	90%	100%	110%	120%	130%	70%	80%	90%	100%	110%	120%	130%
I-3-3-100-01	$^{5}20$	$^{3}18$	$^{1}16$	15	15	15	15	15	15	15	15	15	15	15
I-3-3-100-02	$^{3}20$	17	17	17	17	17	17	17	17	17	17	17	17	17
I-3-3-100-03	$^{4}20$	$^{1}17$	16	16	16	16	16	16	16	16	16	16	16	16
I-3-3-100-04	$^{3}19$	16	16	16	16	16	16	16	16	16	16	16	16	16
I-3-3-100-05	$^{4}23$	$^{3}21$	$^{1}19$	18	18	18	18	19	18	18	18	18	18	18
I-3-3-100-06	$^{6}21$	$^{3}18$	$^{1}16$	15	15	15	15	$^{1}16$	15	15	15	15	15	15
I-3-3-100-07	$^{4}18$	$^{2}16$	$^{1}15$	14	14	14	14	14	14	14	14	14	14	14
I-3-3-100-08	$^{2}18$	$^{2}17$	15	15	15	15	15	16	15	15	15	15	15	15
I-3-3-100-09	$^{1}20$	19	19	19	19	19	19	19	19	19	19	19	19	19
I-3-3-100-10	$^{3}22$	19	19	19	19	19	19	19	19	19	19	19	19	19
mean	**20.1**	**17.8**	**16.8**	**16.4**	**16.4**	**16.4**	**16.4**	**16.7**	**16.4**	**16.4**	**16.4**	**16.4**	**16.4**	**16.4**

preceding superscripts denote duality gaps > 0 after 3600 s of runtime

Table 5.7.: Computational study on the influence of the battery capacity and charging.

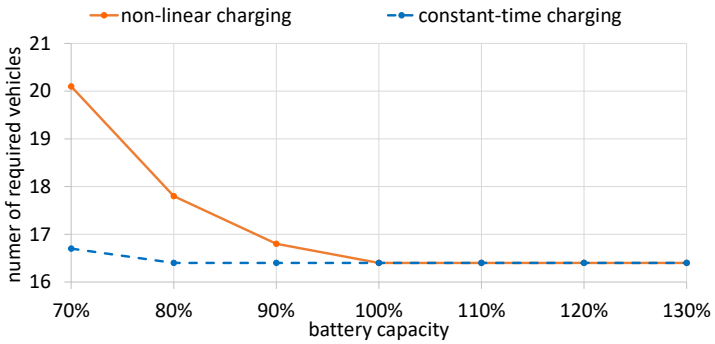


Figure 5.4.: Influence of the battery capacity and charging mode on the fleet size.

The battery charging mode did not influence the number of required vehicles for the regular or an increased battery capacity. However, for below 100 % of battery capacity, more vehicles were required for non-linear charging. For 70 % of battery capacity, the required fleet size also slightly increased for constant-time charging. Furthermore, the lower the battery capacity, the

harder it was for the BCH approach to find optimal solutions, especially for the case of non-linear charging.

Generally, non-linear battery charging required a larger fleet size than constant-time charging. The difference was only relevant for battery capacities below 100 %, however. In conclusion, our experiments show that the battery charging mode and capacity did influence the required fleet size. Above a certain threshold, which was at about 90 % to 100 % battery capacity in our experiments, the effects became marginal, however.

Despite the generally low impact of the charging mode on the fleet size, it has an important influence. As the schedule infeasibility cuts in Tables 5.4 and 5.5 show, significantly more solutions needed to be evaluated for non-linear charging than for constant-time charging before the optimal solution was found. This strongly suggests that the solution space (i.e., the number of solutions with minimal fleet size) is much narrower for the former than for the latter. Consequently, if secondary objectives, additional constraints, or robustness aspects become relevant, constant-time charging offers greater flexibility.

As a final remark, we checked whether solutions found for either of the charging modes would be feasible for the other. While all solutions found for non-linear charging were also feasible for constant-time charging, the opposite was only true for nine solutions and only when the battery capacity was at 120% or above. This is in line with the findings of Olsen and Kliewer (2018) and Olsen and Kliewer (2020) inasmuch as it demonstrates the necessity to model plug-in charging as non-linear charging to avoid schedules that are infeasible in practice.

5.6.2.5. Influence of the number of depots

Our instances consider a facility with multiple warehouses (as depicted in Figure 5.1a), where each warehouse has a separate tow train depot (which is also a charging station). In this section, we investigate how the number of warehouses and depots effects the fleet size.

We generated instances with $s \in \{1, 2, 3, 4\}$ from the regular I-5-5-200-## instances in the following way. Starting with a I-5-5-200-## instance with five warehouses, we merged the first and second warehouse into a single warehouse to attain a comparable instance with four warehouses. Likewise, to attain an instance with three warehouses, we merged warehouses one and two as well as three and four. To attain instances with two warehouses, we merged warehouses one, two, and five, as well as three and four. Finally, instances with a single warehouse were generated by merging all five warehouses. Whenever we merged two or more warehouses, we set

the location of the resulting warehouse within the facility to the average location of all merged warehouses. We did the same for the location of the depot within the merged warehouse. Furthermore, we scaled up the size of the merged warehouse – and all trip start locations within –, such that the merged warehouse's area equals the sum of the separate warehouses' areas. All other parameters were left unchanged to get fair comparisons.

We solved all I-s-d-200-## instances for non-linear and constant-time battery charging with the BCH approach. The results are summarized in Table 5.8.

In general, our experiment only found a small influence in the number of warehouses and depots on the required fleet size. On average having a medium number of warehouses – three to four in our experiments – resulted in the smallest average fleet size. Having five decentralized or only one or two centralized warehouses resulted in marginally larger fleet sizes.

While, on average, detours increased for the case of fewer and therefore larger warehouses, they increased only by a comparatively small amount. On the other hand, since (to make the comparison fair) we assumed merged warehouses to be the average distance of the individual warehouses away from the production area, very long tours that require a large amount of charge were less crucial, which increased flexibility. According to our results, the trade-off was most favorable for a medium number of warehouses. Overall, the effect of the centralization was comparatively small, however, and we expect the warehouses' locations (as opposed to their number) to be a much greater factor in practice.

instance	non-linear battery charging					constant-time battery charging				
label	$s=d=5$	$s=d=4$	$s=d=3$	$s=d=2$	$s=d=1$	$s=d=5$	$s=d=4$	$s=d=3$	$s=d=2$	$s=d=1$
I-s-d-200-01	18	18	18	18	18	18	18	18	18	18
I-s-d-200-02	17	$*$18	18	18	18	17	17	18	18	18
I-s-d-200-03	15	15	15	15	15	15	15	15	15	15
I-s-d-200-04	16	16	16	16	17	16	16	16	16	17
I-s-d-200-05	18	18	17	17	19	18	18	17	17	19
I-s-d-200-06	16	16	16	16	16	16	16	16	16	16
I-s-d-200-07	16	15	15	*16	16	16	15	15	15	16
I-s-d-200-08	16	16	17	17	17	16	16	17	17	17
I-s-d-200-09	18	18	18	18	*19	18	18	18	18	18
I-s-d-200-10	18	17	17	18	18	18	17	17	18	18
mean	**16.8**	**16.7**	**16.7**	**16.9**	**17.3**	**16.8**	**16.6**	**16.7**	**16.8**	**17.2**

* optimality was not proven within 3600 s

Table 5.8.: Computational study on the influence of the number of depots/warehouses.

5.7. Conclusion

In this paper, we consider the multi-depot electric vehicle scheduling problem with the objective of minimizing the required fleet size, which is motivated by a novel application in in-plant logistics. We consider the problem for various battery charging functions, but primarily focus on realistic concave non-linear lithium-ion battery charging and constant-time charging (i.e., battery swapping) in the course of the paper. We briefly discuss the problem's computational complexity and show that finding an optimal solution is NP-hard.

For a discrete version of the problem, we formulate a MIP model. For the real-valued (non-discrete) problem, an exact branch-and-check procedure is developed. The branch-and-check procedure decomposes the problem into two parts: first, a master problem with relaxed battery constraints that is concerned with assigning trips to vehicles and, second, a subproblem that schedules charging breaks and checks whether the master problem's solutions are feasible with regard to the battery constraints.

Computational tests show that solving the MIP model with an off-the-shelf standard solver (i.e., CPLEX) results in a subpar performance. On the other hand, the branch-and-check procedure solves instances with up to $n = 200$ trips in below 2.6 s and instances with up to $n = 4000$ in a couple of hours on average, if constant-time charging is assumed. For non-linear charging, the performance is somewhat worse. Nevertheless, for instances with $n = 200$ trips, the branch-and-check procedure is still able to find optimal solutions within a runtime of one hour. Based on our computational tests, we further derive the following take-home insights from a managerial viewpoint:

- Generally, constant-time battery charging requires a smaller fleet size than non-linear charging. This is especially relevant if the vehicles' maximum battery capacity is small. For larger battery capacities, the effect is marginal. Moreover, constant-time battery charging results in a larger solution space (i.e., more possible solutions with the minimal fleet size) and therefore generally more flexibility.

- Applying a model with constant-time charging to a situation with non-linear charging (i.e., plug-in charging) is not advisable, as solutions found by the constant-time model are very likely to be infeasible for the non-linear charging case. Hence, it is important to actually model non-linear charging as such.

- The greater the vehicles' battery capacity, the fewer vehicles are required. However, this becomes marginal once a certain threshold in battery capacity is reached, which is between

90 % and 100 % of the regular capacity in our experiments. Furthermore, the effect is more severe for non-linear battery charging.

- The required fleet size is smallest for an intermediate number of warehouses and depots. The effect is quite marginal and actual warehouse locations are likely to be much more important, however.

Future research may aim to extend our experiments on the deployment of warehouses and depots in in-plant logistics. While we only considered the effects of the number of warehouses, it stands to reason that their location has a much more severe influence, where there is a multitude of ways depots and warehouses could be arranged in practice. However, such a study would ideally include a case study on typical facility layouts, which is out of the scope of this paper.

While our solution procedure is suitable to solve instances of realistic size (with $n = 200$, an instance covers roughly an eight-hour workday), larger instances with $n \geq 1000$ could not be solved when non-linear plug-in charging was assumed. Therefore, future research may develop heuristic approaches inspired by the idea of the branch-and-check procedure. A straightforward way would be to turn the exact master problem into a heuristic beam search procedure. Alternatively, the master problem could be replaced by a meta-heuristic search approach such as tabu search or simulated annealing.

Finally, future research could consider adapting our general solution approach for electric vehicle problems with other or extended objectives as summarized in Table 5.1. While Li (2014), van Kooten Niekerk et al. (2017), and Adler and Mirchandani (2017) found that column generation performs well as an exact solution approach in that matter, the generally good performance of the BCH procedure in this paper suggests row generation may be a worthwhile consideration, too.

5.A. Appendix

Determining the parameters $\bar{c}^{\max}_{j,j'}$ and $\bar{c}^{\min}_{j,j'}$

Determining the parameter $\bar{c}^{\max}_{j,j'}$ is not straightforward from its definition in Section 5.5.1, because we do not know which value of c maximizes the term $\zeta^*(j,j,c) - c$. The same is the case for $\bar{c}^{\min}_{j,j'}$. However, we can reformulate $\bar{c}^{\max}_{j,j'} = \max_{c\in\tilde{C}_j}\left\{\max_{j''\in S\cup\{0\}}\left\{\zeta(j,j',j'',c)-c\right\}\right\} = \max_{j''\in S\cup\{0\}}\left\{\max_{c\in\tilde{C}_j}\left\{\zeta(j,j',j'',c)-c\right\}\right\}$. Likewise, we can reformulate $\bar{c}^{\min}_{j,j'} = \min\left\{c\in\tilde{C}_j \mid \max_{j''\in S\cup\{0\}}\left\{\zeta(j,j',j'',c)\right\} \geq \min\left\{\tilde{C}_{j'}\right\}\right\} = \min_{j''\in S\cup\{0\}}\left\{\min\left\{c\in\tilde{C}_j \mid \zeta(j,j',j'',c) \geq \min\left\{\tilde{C}_{j'}\right\}\right\}\right\}$. From the reformulations, it becomes evident that we need to consider every $j''\in S\cup\{0\}$ separately, for which we then calculate

$$\bar{c}^{\max,\,\mathrm{red}}_{j,j'}(j'') = \max_{c\in\tilde{C}_j}\left\{\zeta(j,j',j'',c)-c\right\} \tag{5.35}$$

and

$$\bar{c}^{\min,\,\mathrm{red}}_{j,j'}(j'') = \min\left\{c\in\tilde{C}_j \mid \zeta(j,j',j'',c) \geq \min\left\{\tilde{C}_{j'}\right\}\right\}, \tag{5.36}$$

respectively. Moreover, note that we do not need to determine $\bar{c}^{\max}_{j,j'}$ and $\bar{c}^{\min}_{j,j'}$ if $(j,j')\in I$ (cf., Constraints (5.23) to (5.25)).

For $j''=0$ and $(j,j')\notin I$, it follows that $\bar{c}^{\max,\,\mathrm{red}}_{j,j'}(j'') = -\bar{c}_{j,j'} - \tilde{c}_{j'}$ and $\bar{c}^{\min,\,\mathrm{red}}_{j,j'}(j'') = \min\left\{c\in\tilde{C}_j \mid c-\bar{c}_{j,j'}-\tilde{c}_{j'} \geq \min\left\{\tilde{C}_{j'}\right\}\right\} = \min\left\{\tilde{C}_{j'}\right\} + \bar{c}_{j,j'} + \tilde{c}_j$.

For $j''\in S$, we take the following observation into account. For all charging functions η we consider in this paper, charging is faster (or at least not slower) the lower the SOC. Therefore, the maximum increase in the SOC between j and j' occurs if station $j''\in S$ is visited with the minimum possible SOC that allows for a feasible execution. Since $\bar{c}^{\min,\,\mathrm{red}}_{j,j'}(j'')$ is the minimum SOC that must be left at j to execute the transition to j' via the deadheading route j'' feasibly, it follows that $\bar{c}^{\max,\,\mathrm{red}}_{j,j'}(j'') = \zeta(j,j',j'',\bar{c}^{\min,\,\mathrm{red}}_{j,j'}(j'')) - \bar{c}^{\min,\,\mathrm{red}}_{j,j'}(j'')$. Hence, it follows that $\bar{c}^{\min,\,\mathrm{red}}_{j,j'}(j'') = \min\left\{c\in\tilde{C}_j \mid \eta\left(\eta^{-1}\left(c-\bar{c}_{j,j''}\right)+\bar{\tau}\left(j,j',j'',c\right)\right)-\bar{c}_{j'',j'}-\tilde{c}_{j'} \geq \min\left\{\tilde{C}_{j'}\right\}\right\}$ $= \eta\Big(\eta^{-1}\Big(\min\{\tilde{C}_{j'}\}+\bar{c}_{j'',j'}+\tilde{c}_{j'}\Big)-\bar{\tau}(j,j',j'',c)\Big)+\bar{c}_{j,j''}$. Inserting the respective value for c into Expression (5.35) yields $\bar{c}^{\max,\,\mathrm{red}}_{j,j'}(j'')$ accordingly.

To foster the understanding, the procedure to determine $\bar{c}^{\max}_{j,j'}$ and $\bar{c}^{\min}_{j,j'}$ simultaneously is summarized as pseudo-code in Algorithm 5.1.

Algorithm 5.1: Algorithm for determining $\bar{c}^{\max}_{j,j'}$ and $\bar{c}^{\min}_{j,j'}$ simultaneously.

Input: $j, j', \tilde{C}_j, \tilde{C}_{j'}$

$\bar{c}^{\max}_{j,j'} = -\bar{c}_{j,j'} - \tilde{c}_{j'}$; // maximal increase in the SOC between j and j'; the initial value results from the case that $j'' = 0$

$\bar{c}^{\min}_{j,j'} = \bar{c}_{j,j'} + \tilde{c}_{j'} + \min\left\{\tilde{C}_j\right\}$; // minimum required SOC that must remain at j if j' should be reached consecutively; the initial value results from the case that $j'' = 0$

for $j'' \in S$ **do**

  $c := \eta\Big(\eta^{-1}\left(\min\{\tilde{C}_{j'}\} + \bar{c}_{j'',j'} - \tilde{c}_{j'}\right) - s_{j'} + e_j + t_{j,j''} + t_{j'',j'} + r\Big) + \bar{c}_{j,j''}$;

  $\bar{c}^{\max}_{j,j'} := \max\left\{\zeta(j, j', j'', c) - c, \bar{c}^{max}_{j,j'}\right\}$;

  $\bar{c}^{\min}_{j,j'} := \min\left\{c, \bar{c}^{\min}_{j,j'}\right\}$;

Output: $\bar{c}^{\max}_{j,j'}, \bar{c}^{\min}_{j,j'}$

5.B. Bibliography

Adler, J. D. and Mirchandani, P. B. (2017). The vehicle scheduling problem for fleets with alternative-fuel vehicles. *Transportation Science*, 51(2):441–456.

Barré, A., Deguilhem, B., Grolleau, S., Gérard, M., Suard, F., and Riu, D. (2013). A review on lithium-ion battery ageing mechanisms and estimations for automotive applications. *Journal of Power Sources*, 241:680–689.

Beck, J. C. (2010). Checking-up on branch-and-check. In *International Conference on Principles and Practice of Constraint Programming*, pages 84–98. Springer.

Bertossi, A. A., Carraresi, P., and Gallo, G. (1987). On some matching problems arising in vehicle scheduling models. *Networks*, 17(3):271–281.

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.

Briand, C., He, Y., and Ngueveu, S. (2018). Energy-efficient planning for supplying assembly lines with vehicles. *EURO Journal on Transportation and Logistics*, pages 1–28.

Bunte, S. and Kliewer, N. (2009). An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317.

Chao, Z. and Xiaohong, C. (2013). Optimizing battery electric bus transit vehicle scheduling with battery exchanging: model and case study. *Procedia-Social and Behavioral Sciences*, 96:2725–2736.

Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

Den Boer, E., Aarnink, S., Kleiner, F., and Pagenkopf, J. (2013). Zero emissions trucks. an overview of state-of-the-art technologies and their potential. Online: `https://cedelft.eu/wp-content/uploads/sites/2/2021/04/CE_Delft_4841_Zero_emissions_trucks_Def.pdf` [2021-05-12].

Emde, S., Abedinnia, H., and Glock, C. H. (2018). Scheduling electric vehicles making milk-runs for just-in-time delivery. *IISE Transactions*, 50(11):1013–1025.

Emde, S. and Boysen, N. (2012). Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines. *European Journal of Operational Research*, 217(2):287–299.

Emde, S., Fliedner, M., and Boysen, N. (2012). Optimally loading tow trains for just-in-time supply of mixed-model assembly lines. *IIE Transactions*, 44(2):121–135.

Emde, S. and Gendreau, M. (2017). Scheduling in-house transport vehicles to feed parts to automotive assembly lines. *European Journal of Operational Research*, 260(1):255–267.

Emde, S. and Schneider, M. (2018). Just-in-time vehicle routing for in-house part feeding to assembly lines. *Transportation Science*, 52(3):657–672.

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.

Fathi, M., Rodríguez, V., and Alvarez, M. J. (2014). A novel memetic ant colony optimization-based heuristic algorithm for solving the assembly line part feeding problem. *The International Journal of Advanced Manufacturing Technology*, 75(1-4):629–643.

Fathi, M., Rodríguez, V., Fontes, D. B., and Alvarez, M. J. (2016). A modified particle swarm optimisation algorithm to solve the part feeding problem at assembly lines. *International Journal of Production Research*, 54(3):878–893.

Freling, R. and Paixão, J. M. P. (1995). Vehicle scheduling with time constraint. In *Computer-Aided Transit Scheduling*, pages 130–144. Springer.

Goeke, D. and Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99.

Golz, J., Gujjula, R., Günther, H.-O., Rinderer, S., and Ziegler, M. (2012). Part feeding at high-variant mixed-model assembly lines. *Flexible Services and Manufacturing Journal*, 24(2):119–141.

Gschwind, T., Irnich, S., Tilk, C., and Emde, S. (2019). Branch-cut-and-price for scheduling deliveries with time windows in a direct shipping network. *Journal of Scheduling*, pages 1–15.

Haghani, A. and Banihashemi, M. (2002). Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints. *Transportation Research Part A: Policy and Practice*, 36(4):309–333.

He, Y., Song, Z., and Liu, Z. (2019). Fast-charging station deployment for battery electric bus systems considering electricity demand charges. *Sustainable Cities and Society*, 48:101530.

Hooker, J. (2011). *Logic-based methods for optimization: combining optimization and constraint satisfaction*, volume 2. John Wiley & Sons.

Hu, L., Zhou, B., and Li, Y. (2017). An energy saving scheduling method for just in time material handling in mixed-model assembly line. In *ASME 2017 12th International Manufacturing Science and Engineering Conference collocated with the JSME/ASME 2017 6th International Conference on Materials and Processing*. American Society of Mechanical Engineers.

Ibarra-Rojas, O., Delgado, F., Giesen, R., and Muñoz, J. (2015). Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological*, 77:38–75.

Janovec, M. and Koháni, M. (2019a). Battery degradation impact on the electric bus fleet scheduling. In *2019 International Conference on Information and Digital Technologies (IDT)*, pages 190–197. IEEE.

Janovec, M. and Koháni, M. (2019b). Exact approach to the electric bus fleet scheduling. *Transportation Research Procedia*, 40:1380–1387.

Kisacikoglu, M. C., Ozpineci, B., and Tolbert, L. M. (2011). Reactive power operation analysis of a single-phase ev/phev bidirectional battery charger. In *8th International Conference on Power Electronics-ECCE Asia*, pages 585–592. IEEE.

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.

Kovalyov, M. Y., Ng, C., and Cheng, T. E. (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, 178(2):331–342.

Li, J.-Q. (2014). Transit bus scheduling with limited energy. *Transportation Science*, 48(4):521–539.

Lieb, C., Klenk, E., Galka, S., and Keuntje, C. (2017). Einsatz von Routenzugsystemen zur Produktionsversorgung – Studie zu Planung, Steuerung und Betrieb. Technical report, Technische Universität München, Garching.

Lin, Y., Zhang, K., Shen, Z.-J. M., Ye, B., and Miao, L. (2019). Multistage large-scale charging station planning for electric buses considering transportation network and power grid. *Transportation Research Part C: Emerging Technologies*, 107:423–443.

Liu, W. (2013). *Introduction to hybrid vehicle system modeling and control*. John Wiley & Sons.

Liu, Z., Song, Z., and He, Y. (2018). Planning of fast-charging stations for a battery electric bus system under energy consumption uncertainty. *Transportation Research Record*, 2672(8):96–107.

Messaoudi, B. and Oulamara, A. (2019). Electric bus scheduling and optimal charging. In Paternina-Arboleda, C. and Voß, S., editors, *Computational Logistics*, pages 233–247, Cham. Springer International Publishing.

Olsen, N. and Kliewer, N. (2018). Electric vehicle scheduling – a study on charging modeling for electric vehicles. In *Operations Research Proceedings 2016*, pages 653–658. Springer.

Olsen, N. and Kliewer, N. (2020). Scheduling electric buses in public transport: Modeling of the charging process and analysis of assumptions. *Logist. Res.*, 13(1):4.

Pelletier, S., Jabali, O., Laporte, G., and Veneroni, M. (2017). Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. *Transportation Research Part B: Methodological*, 103:158–187.

Reuer, J., Kliewer, N., and Wolbeck, L. (2015). The electric vehicle scheduling problem: A study on time-space network based and heuristic solution approaches. In *Proceedings of the 13th Conference on Advanced Systems in Public Transport (CASPT), Rotterdam*.

Saha, J. (1970). An algorithm for bus scheduling problems. *Journal of the Operational Research Society*, 21(4):463–474.

Sassi, O. and Oulamara, A. (2017). Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–535.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520.

Still (2019). LTX technical data. Online: `https://data.still.de/assets/products/Vehicles/Platform_Trucks_and_Tractors/LTX_50-T06/pdf/LTX_20_LTX_50_EN_TD_web.pdf?mod=1530628525&download=1&s=54a8343b5761ed631c8427172f20b6fe` [2020-05-07].

Sweda, T. M., Dolinskaya, I. S., and Klabjan, D. (2016). Optimal recharging policies for electric vehicles. *Transportation Science*, 51(2):457–479.

Teng, J., Chen, T., and Fan, W. D. (2020). Integrated approach to vehicle scheduling and bus timetabling for an electric bus line. *Journal of Transportation Engineering, Part A: Systems*, 146(2):04019073.

Thorsteinsson, E. S. (2001). Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 16–30. Springer.

van Kooten Niekerk, M. E., van den Akker, J., and Hoogeveen, J. (2017). Scheduling electric vehicles. *Public Transport*, 9(1-2):155–176.

Vetter, J., Novák, P., Wagner, M. R., Veit, C., Möller, K.-C., Besenhard, J., Winter, M., Wohlfahrt-Mehrens, M., Vogler, C., and Hammouche, A. (2005). Ageing mechanisms in lithium-ion batteries. *Journal of power sources*, 147(1-2):269–281.

Wang, H. and Shen, J. (2007). Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints. *Applied Mathematics and Computation*, 190(2):1237–1249.

Wang, Y., Huang, Y., Xu, J., and Barclay, N. (2017). Optimal recharging scheduling for urban electric buses: A case study in Davis. *Transportation Research Part E: Logistics and Transportation Review*, 100:115–132.

Wen, M., Linde, E., Ropke, S., Mirchandani, P., and Larsen, A. (2016). An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, 76:73–83.

Yao, E., Liu, T., Lu, T., and Yang, Y. (2020). Optimization of electric vehicle scheduling with multiple vehicle types in public transport. *Sustainable Cities and Society*, 52:101862.

Zhang, A., Li, T., Tu, R., Dong, C., Chen, H., Gao, J., and Liu, Y. (2021). The effect of nonlinear charging function and line change constraints on electric bus scheduling. *Promet-Traffic&Transportation*, 33(4):527–538.

Zhou, B. and Peng, T. (2017). Scheduling the in-house logistics distribution for automotive assembly lines with just-in-time principles. *Assembly Automation*, 37(1):51–63.

Zhou, B.-h. and Tan, F. (2018). Electric vehicle handling routing and battery swap station location optimisation for automotive assembly lines. *International Journal of Computer Integrated Manufacturing*, 31(10):978–991.

Zhou, G.-J., Xie, D.-F., Zhao, X.-M., and Lu, C. (2020). Collaborative optimization of vehicle and charging scheduling for a bus fleet mixed with electric and traditional buses. *IEEE Access*, 8:8056–8072.