

University of Groningen

Workforce Scheduling with Order-Picking Assignments in Distribution Facilities

Rijal, Arpan; Bijvank, Marco; de Koster, René; Goel, Asvin

Published in:
Transportation Science

DOI:
[10.1287/trsc.2020.1029](https://doi.org/10.1287/trsc.2020.1029)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2021

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Rijal, A., Bijvank, M., de Koster, R., & Goel, A. (2021). Workforce Scheduling with Order-Picking Assignments in Distribution Facilities. *Transportation Science*, 55(3), 725-746.
<https://doi.org/10.1287/trsc.2020.1029>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.



Transportation Science

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

Workforce Scheduling with Order-Picking Assignments in Distribution Facilities

Arpan Rijal , Marco Bijvank , Asvin Goel , René de Koster

To cite this article:

Arpan Rijal , Marco Bijvank , Asvin Goel , René de Koster (2021) Workforce Scheduling with Order-Picking Assignments in Distribution Facilities. Transportation Science 55(3):725-746. <https://doi.org/10.1287/trsc.2020.1029>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2021, INFORMS

Please scroll down for article—it is on subsequent pages







With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

Workforce Scheduling with Order-Picking Assignments in Distribution Facilities

Arpan Rijal,^a Marco Bijvank,^b Asvin Goel,^c René de Koster^d

^a Faculty of Economics and Business, University of Groningen, 9747 AE Groningen, Netherlands; ^b Haskayne School of Business, University of Calgary, Calgary, Alberta T2N 1N4, Canada; ^c Kühne Logistics University, 20457 Hamburg, Germany; ^d Rotterdam School of Management, Erasmus University, 3062 PA Rotterdam, Netherlands

Contact: a.rijal@rug.nl,  <https://orcid.org/0000-0001-8701-2757> (AR); marco.bijvank@haskayne.ucalgary.ca,  <https://orcid.org/0000-0002-2652-7375> (MB); asvin.goel@the-klu.org,  <https://orcid.org/0000-0001-6821-6535> (AG); rkoster@rsm.nl,  <https://orcid.org/0000-0002-1740-7822> (RdeK)

Received: August 16, 2019

Revised: March 31, 2020; June 18, 2020

Accepted: September 14, 2020

Published Online in Articles in Advance:
March 8, 2021

<https://doi.org/10.1287/trsc.2020.1029>

Copyright: © 2021 INFORMS

Abstract. Scheduling the availability of order pickers is crucial for effective operations in a distribution facility with manual order pickers. When order-picking activities can only be performed in specific time windows, it is essential to jointly solve the order picker shift scheduling problem and the order picker planning problem of assigning and sequencing individual orders to order pickers. This requires decisions regarding the number of order pickers to schedule, shift start and end times, break times, as well as the assignment and timing of order-picking activities. We call this the *order picker scheduling problem* and present two formulations. A branch-and-price algorithm and a metaheuristic are developed to solve the problem. Numerical experiments illustrate that the metaheuristic finds near-optimal solutions at 80% shorter computation times. A case study at the largest supermarket chain in The Netherlands shows the applicability of the solution approach in a real-life business application. In particular, different shift structures are analyzed, and it is concluded that the retailer can increase the minimum compensated duration for employed workers from six hours to seven or eight hours while reducing the average labor cost with up to 5% savings when a 15-minute flexibility is implemented in the scheduling of break times.

Funding: This research was partially funded by NWO (Dutch Research Council) as part of the project “Sustainable Logistics in Fresh Food” [Grant 438-13-215].

Supplemental Material: The online appendices are available at <https://doi.org/10.1287/trsc.2020.1029>.

Keywords: facility logistics • personnel scheduling • order picker planning • branch-and-price • metaheuristic

1. Introduction

Scheduling order pickers is one of the fundamental decision problems in manual picker-to-part warehouses, where order pickers walk (or drive) to the storage locations of items to retrieve all the items specified in a picking list. The order-picking process is one of the most labor-, time- and capital-intensive activities in warehouses, responsible for more than 50% of the operating costs (Tompkins et al. 2010). Despite the rise of automated order picking, less than 3% of warehouses are fully automated and less than 10% of warehouses use automated parts-to-picker systems (Michel 2016). Specifically, Azadeh, De Koster, and Roy (2019) estimate that only 40 out of thousands of warehouses in Western Europe are fully automated. Consequently, manual order picking has been studied extensively in the literature, and most research focuses on the development of travel time or distance models for various storage assignment, picking routing, and order batching policies (Van Gils et al. 2018b). In contrast, the *order picker planning* problem, which

assigns and sequences orders to order pickers, has hardly been studied (Van Gils et al. 2018b). This is an important problem for warehouses where orders have temporal restrictions such as deadlines. The assignment and the sequence of execution of orders have a direct impact on the tardiness of orders and on the costs associated with the order-picking operations. Furthermore, as order pickers are humans, the order picker planning problem is further constrained by shift scheduling decisions, which include decisions regarding the start and end times of shifts and breaks, as well as the workforce level requirements for different shifts. The literature on order picker planning ignores these shift scheduling decisions and only considers a single shift horizon (i.e., shifts with one start and end time for all order pickers) without the need for breaks (Matusiak et al. 2014, Henn 2015, Matusiak, De Koster, and Saarinen 2017, Scholz, Schubert, and Wäscher 2017). Therefore, the available solution approaches in the literature can only be applied in a straightforward manner to manual order picker planning problems in warehouses where orders do not have temporal restrictions.

Many distribution centers in Western Europe face two main restrictions in the order-picking operations: due time windows of orders and flexible order pickers. On-time retrieval of customer orders has become more important nowadays, with companies offering deliveries to customers within a small time interval (e.g., one or two business days). To ensure that customer orders are delivered on time, trucks have departure deadlines from the warehouse. In retail logistics, these departure deadlines can also be imposed by strict city access time window regulations (Quak and De Koster 2007) and contractual agreements with retail stores (Bodnar, de Kostner, and Azadeh 2017). In addition to these temporal restrictions, there are also spatial restrictions due to limited capacity at the outbound staging areas of warehouses to consolidate orders that need to be delivered by the same truck. Consequently, every order has a due time window during which it needs to be picked and sent to the allocated staging lane. These due time windows present severe challenges to warehouse managers in maintaining the right order-picking workforce at the appropriate times. To cover demand during peak periods, a large number of order pickers is required. These order pickers can become superfluous when the volume of order-picking tasks decreases. To alleviate this problem, warehouses employ flexible order pickers who can be called upon to work on short notice. The shift start and end times vary for these employees, but they are guaranteed a minimum payment equal to the payment corresponding to the minimum compensated duration, which is defined as the duration of time an order picker is paid for even if the order picker is asked to work in a shift with a shorter shift length. Labor laws in many countries specify a minimum compensation duration. For instance, employees in the United States, Canada, and Australia must be paid for at least three hours each time they are required to report to work. Under these circumstances, the aim of the warehouse manager is to solve the order picker planning problem such that due time windows of orders are respected while minimizing the labor cost. This requires them to determine how many order pickers to schedule (including the start times, end times, and breaks for each order picker), assign the orders that need to be picked during each shift, as well as the sequence in which the orders are picked by the order pickers. We call this optimization problem the *order picker scheduling problem* (OPSP). Most warehouse managers rely on their experience and intuition to make these decisions. Even though our study is inspired by the largest grocery retail chain in The Netherlands, the use of flexible order pickers with minimum compensation and one or multiple break periods is common in many countries. Our definitions of flexible order pickers and break requirements are fully compliant with the current European Union (EU) Directive 91/533/EEC (European

Parliament, Council of the European Union 1991) as well as the new Directive (EU) 2019/1152 (European Parliament, Council of the European Union 2019) that will replace the current directive in 2022. An overview of other labor laws around the world is included in online Appendix D. Our study is generally applicable and relevant to many manual order-picking warehouses where orders have tight due times and the resources to prepare orders (such as the number of staging lanes) and order pickers are limited.

In this paper, we combine the order picker planning problem with the shift scheduling problem to jointly determine the scheduling of start, end, and break times for the shifts of flexible order pickers as well as the assignment and sequencing of orders with due time windows to these order pickers. The shift decisions have direct implications for the order-picking process that should not be ignored. In online Appendix A, an illustrative example is given that highlights the importance of explicitly constructing shifts that take breaks into account when orders have due time windows and order pickers are flexible. The contributions of our work are four fold. (i) We introduce the OPSP to the order-picking literature and formulate the OPSP as a mixed integer linear program (MILP). (ii) To solve the problem, we present an exact branch-and-price algorithm in combination with an efficient heuristic to generate tight upper bounds based on the savings algorithm. (iii) We propose a computationally efficient metaheuristic that is capable of producing near-optimal solutions for large instances. (iv) A case study is performed to investigate the practical impact of flexible shift structures and show the impact can be substantial.

The outline of this paper is as follows. Relevant literature is reviewed in Section 2. Section 3 presents the problem description and the model formulation of the problem. In Section 4, we present a branch-and-price algorithm to find optimal solutions for the problem. A metaheuristic to solve the problem is proposed in Section 5. Results from computational experiments and the case study follow in Section 6. Finally, Section 7 concludes the paper.

2. Literature Review

As identified in the previous section, the OPSP operates at the intersection of shift (or personnel) scheduling and order picker planning. More details on both research streams in the literature are provided in this section.

2.1. Order Picker Planning Problem

When orders have temporal restrictions (such as due time windows) or when they result in penalties when completed early or late, the assignment of orders to order pickers and the sequencing to execute these orders have a direct impact on the feasibility of workforce schedules and the associated costs.

Elsayed et al. (1993) and Elsayed and Lee (1996) are the first authors to study the joint order batching and sequencing problem for a single automated storage and retrieval system (AS/RS) where the objective is to minimize the earliness and tardiness of orders. The authors suggest simple heuristic methods to generate solutions for the problem. Henn and Schmid (2013) and Henn (2015) extend this work to multiple order pickers, which is considered the joint order batching, assignment, and sequencing problem (JOBASP). The authors suggest iterated local search and attribute-based hill climber, variable neighborhood search, and variable neighborhood depth algorithms to solve this problem. Tsai, Liou, and Huang (2008) introduce a joint order batching, assignment, sequencing, and routing problem (JOBASRP), which is an extension of the JOBASP with routing decisions for the order pickers within the warehouse. Chen et al. (2015) and Scholz, Schubert, and Wäscher (2017) propose heuristic solution approaches for this problem. Matusiak et al. (2014) investigate a variation of JOBASRP where the sequencing of batches is not relevant but the routing is part of the optimization problem, which aims at minimizing the overall travel distance. In most applications, the storage racks are stationary, however, Boysen, Briskorn, and Emde (2017) consider an interesting variation with mobile rack warehouses, where an entire storage aisle may need to be moved to access items in it. Here, the objective is to sequence orders to minimize the number of aisle relocations.

In a recent review on order-picking problems, Van Gils et al. (2018b) note that there is hardly any literature on the integration of the order assignment and sequencing decisions for order pickers (i.e., the order picker planning problem) while determining the order-picking workforce (i.e., the shift scheduling problem). All work in the literature on scheduling manual order pickers assumes a single shift start and end time without the need for a break, which can be traced back to Elsayed et al. (1993) and Elsayed and Lee (1996). This simplifying assumption is only valid for machine environments or for manual order-picking environments where a fixed number of order pickers can start and end their shift at only one given time, no breaks are scheduled, and orders do not have temporal restrictions (as discussed in Section 1). When order pickers have fixed employment contracts, the shift scheduling decisions are typically made at a tactical level or at least

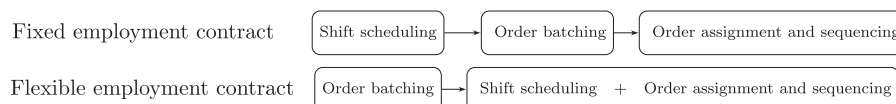
before any order assignment and sequencing decisions are made. However, when order pickers have flexible employment contracts, it is crucial to make shift scheduling decisions at the same time as the order assignment and sequencing decisions are made. Figure 1 illustrates the typical order in which decisions are made in the two types of employment contracts. These differences require us to review shift scheduling literature, which is done in the following. Note that batching is decoupled in both of the contracts because integrating optimal order batching with other decisions is computationally prohibitive in realistic settings. Furthermore, an appropriate batching policy alone can explain much of the variance in travel times of order pickers compared with related decisions (storage, zoning, and routing) (Van Gils et al. 2018a).

2.2. Shift (or Personnel) Scheduling Problem

In contrast to the literature on order-picking processes, the shift scheduling literature explicitly considers shift decisions as part of the planning problem. Shift scheduling is one of the oldest problems in the field of operations research. It dates back to Edie (1954) and Dantzig (1954), who scheduled toll booth operators, and it has received a lot of attention in the literature since then (Ernst et al. 2004a, b; Van den Bergh et al. 2013). Many of the mathematical formulations are based on a generalized set covering model where each possible shift (i.e., a combination of start time, end time, and break placement) is represented by a decision variable. The goal is to determine the optimal complement of shifts such that operational constraints are satisfied while optimizing some objective function. It has applications in many industries including airlines, public transportation, hospitality, military, healthcare, and call centers.

Shift scheduling problems can be divided into two broad categories based on the type of workload they consider: workload-coverage problems and task-coverage problems. The main distinction between these two categories relies on what is known prior to performing the personnel planning. In workload-coverage problems, the actual tasks that need to be executed during the planning horizon are not known when personnel are scheduled. Consequently, the demand for employees is forecasted based on expected workloads and workers are scheduled to cover these predicted personnel demands. Employees are

Figure 1. Sequence of Decision Problems with Fixed and Flexible Employment Contracts for Order Pickers



usually scheduled to perform one type of task that can be preempted between employees working in different shifts (e.g., manning a cash register in a shop).

In contrast to workload-coverage problems, the actual tasks that need to be executed are known in task-coverage problems. In addition to the creation of shifts, these problems include assignment decisions of tasks to individual employees or shifts such that all, or as many as possible, tasks are completed. Consequently, task-coverage problems are generally more complicated to solve than workload-coverage problems. Task-coverage problems can be further divided into two subcategories: fixed task timing problems and flexible task timing problems. In fixed task timing problems, when to execute each task is known a priori (therefore, sequencing decisions are not included). These problems aim to generate schedules that cover the fixed tasks with a minimum number of machines or shifts. Examples of these problems are fixed job scheduling problems (Fischetti, Martello, and Toth 1987, 1989), interval scheduling problems (Kroon, Salomon, and Van Wassenhove 1995; Kolen et al. 2007) and shift minimization personnel task scheduling problems (Krishnamoorthy, Ernst, and Baatar 2012). The navy personnel planning problem studied by Holder (2005) is closely related. Another example of the fixed task timing problem is the integrated task scheduling and personnel rostering problem, which generates the roster of employees while explicitly considering the coverage of tasks (Smet, Ernst, and Berghé 2016). Beliën and Demeulemeester (2008) present a branch-and-price algorithm for the integrated rostering problem of nurses while incorporating the scheduling of tasks that arise from surgery schedules. When the planning horizon of the fixed task timing problem is divided into periods and the duration to execute each task is equivalent to the length of a period, this is known in the literature as the multiactivity shift scheduling problem (Côté, Gendron, and Rousseau 2011; Elahipanah, Desaulniers, and Lacasse-Guay 2013; Dahmen, Rekik, and Soumis 2018).

In flexible task timing problems, when to execute tasks is a decision. Consequently, sequencing decisions have to be made in addition to shift scheduling and task assignment decisions. For instance, home care workers are assigned to locations where tasks (such as cooking, cleaning, and administering medicine) must be performed within specific time windows (Rasmussen et al. 2012). Closely related problems include the field workforce scheduling problem, where individual workers with appropriate skills are assigned to geographically distributed tasks (Alsheddy and Tsang 2011), and the technician task scheduling problem, where individuals with the correct skill mix are assigned to tasks of different priorities (Cordeau et al. 2010, Firat and Hurkens 2012).

2.3. Shift Scheduling Problem with Breaks

The inclusion of breaks in the personnel scheduling literature is mainly limited to workload-coverage problems. Thompson (1988) is one of the first authors to explicitly plan for breaks when shift schedules are generated. In its simplest form, the set covering formulation of Dantzig (1954) is extended with additional decision variables to represent breaks and reliefs. For problems involving a high degree of flexibility with respect to the timing of breaks, the number of enumerated shifts increases drastically and the resulting set covering problem can be very difficult to solve (if even possible). To overcome these challenges, Bechtold and Jacobs (1990) propose a compact formulation that implicitly considers breaks, but is tractable for realistic instances. This model is extended by Thompson (1995) to consider different types of breaks and even overtime. Aykin (1996) also presents a compact integer programming model that is capable of considering time windows for multiple breaks in one shift. Aykin (2000) shows that this model is computationally superior to the formulation in Bechtold and Jacobs (1990), who only consider one break in a shift. Sungur, Özgüven, and Kariper (2017) present a goal programming approach for the same problem studied by Aykin (1996).

The work of Bechtold and Jacobs (1990) is also extended by Brusco and Jacobs (2000) to introduce break and relief planning in tour scheduling problems. In this type of problem, the aim is to generate a schedule with multiple shifts for each employee as well as off days during which the employee is not working. Consequently, the planning horizon is longer than for shift scheduling problems. Bard, Morton, and Wang (2007) also model a tour scheduling problem with break and labor rules but in a stochastic environment of a parcel sorting center. Gérard, Clautiaux, and Sadykov (2016) present a heuristic that is based on column generation for a more extensive problem, which simultaneously considers off days, shift scheduling, shift assignments, and task assignments within shifts. A key difference of these problems compared with our OPSP is that the tasks have a fixed timing rather than a time window during which they need to be performed.

For flexible task timing problems, the scheduling of breaks is only included in the truck driver scheduling problem. In these problems, the sequence in which locations are visited by trucks has to be determined while satisfying appropriate time windows. The maximum amount of time a truck driver is allowed to be on the road is restricted such that breaks and rest periods have to be considered to satisfy the strict hours-of-service regulations (Goel 2010, Goel and Kok 2012). The truck driver scheduling problem is extended to vehicle routing decisions in Goel and Irnich (2017). In these studies, the objective of the problem is to minimize the

Table 1. Comparison of OPSP to the Shift Scheduling and Order Picker Planning Literature

Type of problem (representative paper)	Coverage		Task timing			Minimum compensated duration	
	Task	Workload	Fixed	Flexible	Window		Break timing
Parcel sorting center scheduling (Bard, Morton, and Wang 2007)		✓				✓	
Order assignment sequencing (Scholz, Schubert, and Wäscher 2017)	✓			✓	✓		
Fixed job scheduling (Fischetti, Martello, and Toth 1989)	✓		✓				
Interval scheduling (Kroon, Salomon, and Van Wassenhove 1995)	✓		✓				
Shift minimization (Krishnamoorthy et al. 2012)	✓						
Nurse rostering and task scheduling (Beliën and Demeulemeester 2008)	✓		✓				
Home care scheduling (Rasmussen et al. 2012)	✓			✓	✓		
Field workforce scheduling (Alsheddy and Tsang 2011)	✓			✓	✓		
Technician task scheduling (Cordeau et al. 2010)	✓			✓			
Call center scheduling (Bhandari, Scheller-Wolf, and Harchol-Balter 2008)		✓				✓	
Hotel staff scheduling (Thompson and Pullman 2007)		✓				✓	
Navy personnel planning (Holder 2005)	✓		✓			✓	
Tour scheduling (Brusco and Jacobs 2000)	✓		✓			✓	
Multiactivity shift scheduling (Dahmen, Reikik, and Soumis 2018)	✓	✓				✓	
Truck driver scheduling (Goel and Irnich 2017)	✓			✓	✓	✓	
Order picker scheduling problem	✓			✓	✓	✓	✓

travel distance. An alternative objective function for the problem is presented by Tilk and Goel (2020), where the problem aims to minimize the number of working days for a given route instead of the travel distance.

A comparison between OPSP and the available literature on shift scheduling problems can be found in Table 1. It becomes clear that the order picker planning problem does not consider shift scheduling decisions when orders have temporal restrictions. Furthermore, most of the flexible task timing problems in the shift scheduling literature do not consider the characteristics that are unique to warehouse environments (i.e., tasks with due time windows in combination with flexible workers who require breaks and a minimum payment).

In the shift scheduling literature with task assignments that have to be performed in a certain time window, only the truck driver scheduling problem considers breaks. It is therefore the most closely related to our problem formulation. The break requirements for truck drivers considered in Goel and Irnich (2017) are similar to the break requirements for order pickers considered in the OPSP. However, a major difference is that Goel and Irnich (2017) focus on minimizing travel distances, whereas schedule durations do not play a role. The objective in Tilk and Goel (2020) is to minimize the sum of labor costs and distance-related costs whereas labor costs are related to the number of working days required to complete the route. The number of hours worked within a working day does not play a role, and most schedules generated actually

include long waiting periods. The OPSP studied in our work combines elements of minimizing schedule duration with minimum compensated duration, which make the OPSP structurally different from the aforementioned problems and necessitates new solution approaches. Our work addresses this gap in the literature and combines the order picker planning literature and shift scheduling literature. In Section 4, we further explain the differences between our approach to solve the OPSP and other approaches in the literature.

3. Problem Description and Model Formulation

In this section, we explain the warehouse operations that define our OPSP and formulate the corresponding MILP model. Symmetry breaking constraints and additional constraints to tighten the model formulation are included in online Appendix B. An alternative formulation of the problem as a network flow problem is presented in online Appendix C. This formulation takes more computational effort to solve in our numerical experiments, so it is included for reference only. Extending the OPSP formulation with full-time order pickers and different types of break time constraints are discussed in online Appendix D.

Production facilities or retail stores place orders to receive items from a distribution warehouse based on their needs. An order is composed of multiple order lines, where each order line consists of a particular item and the corresponding requested quantity. The order lines that should be processed together create a

pick list. The list contains all items that need to be picked and it guides the order picker through the warehouse. Items that are collected are put in roll cages such that products in the same roll cage are sent to a single customer. However, a customer's order can result in multiple roll cages picked by one or more order pickers. An order picker's tour finishes when all roll cages from the pick list are delivered to the corresponding staging lanes at the outbound docks. The total number of order lines and roll cages can exceed hundreds, which prohibits a joint optimization of the personnel scheduling and order batching problems within a reasonable computational effort. Consequently, we assume that order batching (i.e., the construction of pick lists) is done a priori. In the remainder of the paper, we use the term "batch" to refer to a pick list that is to be completed by a single order picker in a single pick tour.

Let I be the set of batches that are generated a priori. The time required to pick batch $i \in I$ is denoted by t_i . It includes the time for an order picker to travel between product locations of items in the batch, search for the items, place them in roll cages, and transport the filled roll cages to the staging lanes. We assume that t_i is independent of the order picker and its value is deterministic since the picking route is determined by the storage locations of the items in the batch and the routing strategy of the warehouse. The company in our case study uses norm times that are set to pick a certain batch.

Each batch $i \in I$ has a corresponding delivery due time window $[r_i, d_i]$. All items in the batch have to be delivered to the designated staging area(s) within this time window. The values of r_i and d_i are determined based on the outbound truck departure schedule and the capacity of the staging lanes. The value of r_i usually corresponds to the departure time of the previous vehicle that departed from the same staging lane as where the vehicle for batch i is departing from, and d_i is the latest time batch i can be delivered at the staging lane for the vehicle to depart on time (i.e., without violating the delivery due time at the customer).

Let P represent the set of the flexible order pickers that can be employed by the warehouse, where $|P| = p_{\max}$. Flexible workers are scheduled to work when needed, and as such, they are assigned one of a variety of possible shift lengths with different start times on any day. They are only compensated for the amount of time they spend at the warehouse. Although there is often no restriction on the minimum shift length for a worker, warehouses favor providing a minimum compensation if an employee is scheduled to work. This improves the working relation between the flexible order pickers and the warehouse to increase employee retention. The time corresponding to the minimum compensation duration is denoted by T_{\min} .

The maximum amount of time an employee can work per day is restricted by law and gives an upper limit on the shift length, which we denote by T_{\max} .

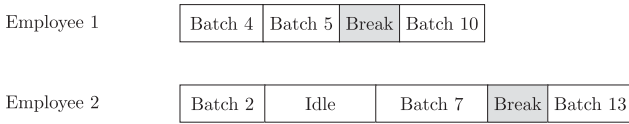
There are also labor rules and union agreements on breaks for human order pickers. The amount of time an employee can work without a break is denoted by T_{break} . An employee who works for a duration that exceeds T_{break} time units must be given an uninterrupted break of at least l_b time units. An employee can be entitled to more than one break in the same shift depending on the values of T_{break} and T_{\max} . The length of the planning horizon is T_{day} time units. Formulations for alternative types of breaks are presented in online Appendix D. We assume that order picking is scheduled nonpreemptively and breaks cannot interrupt this. Interrupting a pick tour and leaving picking equipment in the storage area creates congestion as well as safety and security hazards. Limited parking space for order-picking equipment in the break areas and issues of theft or responsibility of already picked items may also prevent preemptive batch scheduling. If breaks can preempt the order picking of a batch, we propose an updated solution framework and perform a numerical comparison in online Appendix K.

Even though flexible employees can potentially start and end their shifts at any time, many shift start and end times are an administrative and operational burden, and labor union agreements can also prohibit unrestricted shift start and end times (Brusco and Jacobs 1998). Furthermore, employees are paid in integral multiples of a certain duration (even if they completed the last task of their shift before the end of a certain time period). Therefore, we divide the planning horizon into W time periods of equal length, where each period consists of l time units. The set of admissible time periods to start or end a shift is denoted by S and E , respectively. Note that the discretization of the time horizon is only used for shift start and end times. The actual tasks that need to be executed can still start and end at any point in time during the shift (i.e., they do not have to coincide with the time periods) and the same holds for breaks.

We make the assumption that all order-picking operations associated with the batches in the planning horizon are performed within the same planning horizon, and we assume that all shifts of the order pickers start and end in the same planning horizon that they are scheduled for (i.e., there is no overlap between either order-picking tasks of a batch or shifts of order pickers in different planning horizons).

Furthermore, we define a task as an activity that needs to be scheduled—either picking orders of a batch or taking a break. Arranging tasks in a sequence creates a shift, and each task in the sequence has a position (first, second, and so on). This is illustrated

Figure 2. A Gantt Chart to Illustrate the Concepts of Tasks, Shifts, and Task Positions in a Sequence



with a Gantt chart in Figure 2. Employee 1 picks the items in batch 4 and 5 successively, then takes a break, and finally picks items in batch 10 before ending the shift. Note that the order picker completes four tasks but not necessarily consecutively (i.e., there can be an interruption or gap between two successive tasks), which is the case for employee 2. Each order picker can perform at most \bar{k} tasks in a shift. A summary of all parameters is provided in Table 2.

The following decision variables are used in our model formulation:

The variable x_{ikp} is 1 if batch $i \in I$ is scheduled to be picked at the k^{th} position in the shift for order picker $p \in P$, where $k \in K$, else 0.

The variable y_{kp} is 1 if a break is scheduled at the k^{th} position in the shift for order picker $p \in P$, where $k \in K$, else 0.

The variable s_{jp} is 1 if order picker $p \in P$ starts the shift at the beginning of period $j \in S$, else 0.

The variable e_{jp} is 1 if order picker $p \in P$ ends the shift at the end of period $j \in E$, else 0.

The variable c_{kp} is the completion time of the task scheduled at the k^{th} position in the shift for order picker $p \in P$, where $k \in K$.

Table 2. Overview of the Parameters for the Order Picker Scheduling Problem

Notation	Description
P	Set of order pickers that can be scheduled, $\{1, \dots, p_{\max}\}$
I	Set of batches that need to be picked
K	Set of positions in which an order picker can perform a task, $\{1, \dots, \bar{k}\}$
t_i	Duration to pick and deliver the items of batch $i \in I$
r_i	Earliest due time of batch $i \in I$
d_i	Latest due time of batch $i \in I$, where $d_i \geq \max\{r_i, t_i\}$
T_{\min}	Minimum time an order picker needs to be compensated if scheduled
T_{\max}	Maximum shift length
T_{break}	Maximum time duration an order picker can work consecutively without a break
T_{day}	Length of the planning horizon
l	Length of a time period
J	Set of time periods, $\{1, \dots, W\}$
S	Set of time periods where a shift can start at the beginning of that period, $S \subseteq J$
E	Set of time periods where a shift can end at the end of that period, $E \subseteq J$
l_b	Duration of a break
M	A very large number

The variable m_p is the amount of time for which order picker $p \in P$ is compensated.

The OPSP is formulated as a MILP model as follows:

$$\min \sum_{p \in P} m_p, \quad (1)$$

subject to

$$\sum_{i \in I} x_{ikp} + y_{kp} \leq 1 \quad \forall k \in K, p \in P, \quad (2)$$

$$\sum_{k \in K} \sum_{p \in P} x_{ikp} = 1 \quad \forall i \in I, \quad (3)$$

$$\sum_{i \in I} x_{i1p} + y_{1p} \leq \sum_{j \in S} s_{jp} \quad \forall p \in P, \quad (4)$$

$$\sum_{j \in S} s_{jp} = \sum_{j \in E} e_{jp} \quad \forall p \in P, \quad (5)$$

$$c_{1p} \geq \left(\sum_{j \in S} (j-1) s_{jp} \right) l + \sum_{i \in I} t_i x_{i1p} + l_b y_{1p} \quad \forall p \in P, \quad (6)$$

$$c_{kp} \geq c_{k-1,p} + \sum_{i \in I} t_i x_{ikp} + l_b y_{kp} \quad \forall k \in K \setminus \{1\}, p \in P, \quad (7)$$

$$\sum_{j \in E} (j e_{jp}) l \geq c_{\bar{k}p} \quad \forall p \in P, \quad (8)$$

$$c_{kp} + M(1 - x_{ikp}) \geq r_i \quad \forall i \in I, k \in K, p \in P, \quad (9)$$

$$c_{kp} - M(1 - x_{ikp}) \leq d_i \quad \forall i \in I, k \in K, p \in P, \quad (10)$$

$$c_{kp} - \left(c_{1p} - \sum_{i \in I} t_i x_{i1p} \right) \leq T_{\text{break}} + M \left(\sum_{k'=h+1}^k y_{k'p} \right) \quad (11)$$

$\forall h, k \in K, h < k, p \in P,$

$$\sum_{i \in I} x_{ik-1,p} + y_{k-1,p} \geq \sum_{i \in I} x_{ikp} + y_{kp} \quad \forall k \in K \setminus \{1\}, p \in P, \quad (12)$$

$$T_{\min} \sum_{j \in S} s_{jp} \leq m_p \quad \forall p \in P, \quad (13)$$

$$\left(\sum_{j \in E} j e_{jp} - \sum_{j \in S} (j-1) s_{jp} \right) l \leq m_p \quad \forall p \in P, \quad (14)$$

$$c_{kp} \geq 0 \quad \forall p \in P, k \in K, \quad (15)$$

$$x_{ikp} \in \{0, 1\} \quad \forall i \in I, k \in K, p \in P, \quad (16)$$

$$y_{kp} \in \{0, 1\} \quad \forall k \in K, p \in P, \quad (17)$$

$$s_{jp} \in \{0, 1\} \quad \forall j \in S, p \in P, \quad (18)$$

$$e_{jp} \in \{0, 1\} \quad \forall j \in E, p \in P, \quad (19)$$

$$0 \leq m_p \leq T_{\max} \quad \forall p \in P. \quad (20)$$

The objective function (1) expresses the minimization of the total labor cost over all order pickers who are scheduled to pick the items that need to be delivered during the planning horizon. Constraints (2) ensure that order pickers can perform at most one task in the

k^{th} position of their shift. Constraints (3) ensure that each batch is picked exactly once. Order pickers can only perform the first task in a shift if they are scheduled to start a shift according to constraints (4). Constraints (5) ensure that every order picker who starts a shift also ends a shift (and vice versa).

Constraints (6) and (7) determine that the task in the k^{th} position of the order picker's shift can only be labeled as completed after it is executed. Constraints (8) ensure that the order picker can only finish a shift after completing the last assigned task. Constraints (9) and (10) require that batches are completed within their due time windows. Note that an order picker can have fewer than \bar{k} tasks assigned to a shift. In that case, for all positions in a shift without an actual task assigned (i.e., for all k where $\sum_i x_{ikp} + y_{kp} = 0$), the completion times c_{kp} are set equal to the completion time of the last assigned task (i.e., $c_{kp} = c_{k-1,p}$).

Constraints (11) states that an order picker cannot work successively for a duration more than T_{break} time units without a break. The constraint specifies that the time between the start of the task at position h of the shift and the end of the task at position k , where $k > h$, has to be less than or equal to T_{break} if no break is scheduled between these two tasks. Constraints (12) specify that a task can only be assigned to a position if there is also a task assigned to the previous position.

Constraints (13) ensure that an order picker is compensated for at least T_{min} time units if the picker is scheduled to work. Constraints (14) ensure that an order picker is compensated for at least the amount of time the order picker is scheduled to work (i.e., from the start time of the shift to the end time of the shift). Constraints (15) to (20) define the domain and range of the decision variables.

Proposition 1. *Generating a feasible solution for the OPSP is NP-hard in the strong sense.*

Proof. The $P||C_{\text{MAX}}$ problem is a special case of the OPSP. \square

4. Branch-and-Price Algorithm for OPSP

This section outlines an exact procedure to solve the OPSP using a branch-and-price framework. In this solution approach, the linear relaxation in each node of a branch-and-bound tree is solved with column generation (Barnhart et al. 1998, Vanderbeck 2000). A branch-and-price solution approach remains a successful and popular solution strategy for generating optimal solutions for problems in a variety of fields, ranging from transport planning (Bertsimas et al. 2019), routing (Dellaert et al. 2018), to personnel scheduling (Van den Bergh et al. 2013). We also develop a branch-and-price algorithm for the OPSP. We first present the reduced master problem (RMP). The pricing problem

to verify the optimality of a linear programming (LP) solution is presented in Section 4.2. The branching that occurs when the LP solution does not satisfy the integrality conditions is discussed in Section 4.3.

The proposed framework for the branch-and-price algorithm has similarities to the one used by Goel and Irnich (2017). However, because we use the schedule duration in the objective function (which includes employee waiting times between the performance of two tasks) and include the minimum compensated duration as constraints, the details of the building blocks for the branch-and-price algorithm are different from the algorithm in Goel and Irnich (2017). Specifically, the augmented graph for the pricing problem requires information on shift starting and ending times. The definitions of resources and resource extension functions that are used to solve the pricing problem also differ and are more comparable to those used for the minimum tour duration problem (MTDP) (Tilk and Irnich 2017) than the truck driver scheduling problem. Furthermore, because of the constraints regarding the minimum compensated duration and flexible breaks, the problem suffers from significant issues of symmetry. Therefore, we develop a tailored acceleration strategy to address these issues (see the end of Section 4.2).

4.1. Reduced Master Problem

To present the RMP for the OPSP in a column-generation format, we first introduce the concept of a column as a feasible shift schedule that is specified by the start and end time as well as the assignment and sequence of tasks (both order picking and breaks) to be performed by a single order picker while respecting the due time windows of order-picking tasks, maximum shift length T_{max} , and maximum time between breaks T_{break} . Let Ω denote a set of all feasible schedules, where Ω' is a subset of Ω (i.e., $\Omega' \subseteq \Omega$). The cost for an individual schedule $q \in \Omega'$ is given by m_q . The parameter α_{iq} is set to one if batch i is processed (or picked) in schedule q , and zero otherwise. The decision variable θ_q represents the number of schedules of type q to be selected in the solution. The RMP can be formulated as a set covering problem:

$$\min \sum_{q \in \Omega'} m_q \theta_q \quad (21)$$

subject to

$$\sum_{q \in \Omega'} \alpha_{iq} \theta_q \geq 1 \quad \forall i \in I, \quad (22)$$

$$\sum_{q \in \Omega'} \theta_q \leq p_{\text{max}}, \quad (23)$$

$$\theta_q \geq 0 \quad \forall q \in \Omega'. \quad (24)$$

The objective in the RMP is the same as in the OPSP. Constraints (22) ensure that all batches are processed (or covered) with the selected schedules. Constraints (23) do not select more than p_{\max} schedules to be performed by order pickers. The constraints (2) and (4) to (20) of the OPSP are included in the pricing problem where columns are generated that result in feasible schedules.

4.2. Pricing Problem

The pricing problem for the OPSP can be formulated as an elementary shortest path problem with resource constraints (ESPPRC) (Feillet et al. 2004). This is a variation of the shortest path problem with resource constraints (SPPRC) where cycles are not allowed, that is, a node cannot be visited more than once. The SPPRC can be solved with pseudo-polynomial algorithms (Irnich and Desaulniers 2005), whereas the ESPPRC is NP-hard in the strong sense (Dror 1994). Nevertheless, ESPPRC is known to generate a superior lower bound compared with SPPRC when used as pricing problem (Contardo, Desaulniers, and Lessard 2015). A technique to solve the ESPPRC is a labeling algorithm based on dynamic programming (Feillet et al. 2004). This approach uses the concepts of resources in a graph and resource extension functions. A resource is an arbitrary one-dimensional piece of information that can be determined or measured at the vertices of a directed walk in a graph (e.g., cost, time, load). In this paper, time is the main resource. Labels are used to store the information on the resource values for partial paths. Labels reside at vertices and they are propagated via resource extension functions when they are extended along an arc. To keep the number of labels as small as possible, we define dominance rules to identify labels that need not be extended. We first introduce the graph structure, labels, resource extension functions, and dominance rules.

4.2.1. Graph Representation. Consider a subgraph $G = (V, A)$, where V is the set of vertices indicating the set of batches $i \in I$ that have to be picked and the arcs A indicate the subsequent sequence in which the batches are completed. The nodes in the sets S and E indicate the start and end times of a shift, respectively. Furthermore, dummy origin and destination nodes are indicated by o and d , respectively. The complete set of all vertices is $V' := \{o\} \cup S \cup V \cup E \cup \{d\}$.

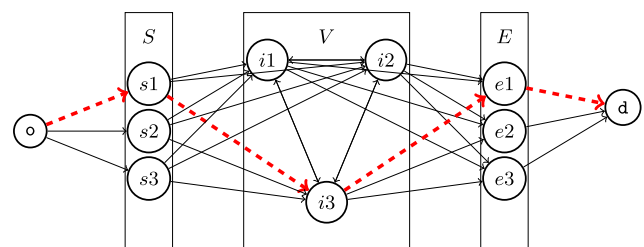
We introduce arcs between the dummy origin node o and the shift start time nodes in S , between each vertex in S and V , between each vertex in V and E , as well as between the shift end time nodes and the dummy destination node d . See Figure 3 for an example. The travel time for each arc is set to zero. The service time t_i at each node $i \in V$ equals the processing time of batch i , whereas the service time at the remaining vertices $V' \setminus V$ is zero.

The time windows for the origin and destination nodes are $[r_i; d_i] = [0; T_{\text{day}}]$ for $i \in \{o, d\}$ such that these nodes can be visited at any time during the time horizon. For the shift start time nodes $i \in S$, the value of $r_i = d_i$ equals the possible shift start times such that these nodes are visited at these specific times. Similarly, for the shift end time nodes $i \in E$, the value of $r_i = d_i$ equals the possible shift end times. A feasible schedule for an order picker comprises a tour from node o to node d respecting the due time windows $[r_i; d_i]$ for $i \in V'$, maximum shift length T_{\max} , and the time until breaks T_{break} . As an illustrative example, Figure 3 represents a graph where there are three possible shift start and end times. The dashed arrow indicates a feasible schedule that starts at $s1$, then executes batch $i3$ and ends at $e1$.

4.2.2. Labels. A partial schedule corresponds to a partial path in the graph G . A partial schedule h where vertex i is visited as last node is defined by label $L_h^i = (i, c_h^i, T_i, (V_h^1, \dots, V_h^{|V|}))$, where:

- i is the last vertex that has been visited in the partial schedule;
- c_h^i is the reduced cost of the partial schedule (i.e., the actual cost minus the dual values of the nodes visited, see Section 4.2.3 for more details);
- $T_i = (T_i^{\text{time}}, T_i^{\text{dur}}, T_i^{\text{start}}, T_i^{\text{work}}, T_i^{\text{brk}})$ indicates the resource vector, where the resource variables are:
 - T_i^{time} is the time when the batch at node i is completed,
 - T_i^{dur} is the minimum duration required to service all the nodes in the partial schedule including the waiting times if necessary to respect the due time windows,
 - T_i^{start} is the latest possible start time of the shift to feasibly visit all of the vertices in the partial schedule while respecting the due time windows,
 - T_i^{work} is the amount of time since the end of the last break,
 - T_i^{brk} is the latest time to start picking the first batch after the previous break to ensure feasibility of the schedule;
- V_h^v is one if node $v \in V$ is visited in the partial schedule or if it is infeasible to visit (due to the due time windows or maximum shift length), zero otherwise.

Figure 3. (Color online) Representation of a Graph Structure for the Pricing Problem of the OPSP with Three Shift Start and End Times and Three Batches



To guarantee elementarity of a (partial) path, it is sufficient to add the extra resources V_h^v for each node $v \in V$ indicating whether the node has been visited on the path. When this resource has the value one, it prohibits the path from re-entering previously visited nodes. Feillet et al. (2004) enhance this idea by observing that some nodes are not reachable due to the resource constraints, which they indicate by setting the resources V_h^v to one for these nodes without the path having to visit them. They use this to speed up the dominance check, which is explained later in this section.

The resource windows of resource vector T_j are given by $T_j^{time} \in [r_j; d_j]$, $T_j^{dur} \in [0; T_{\max}]$, $T_j^{start} \in (-\infty; T_{\text{day}}]$, $T_j^{work} \in [0; T_{\text{break}}]$, and $T_j^{brk} \in (-\infty; \infty)$. A path is called resource-feasible if there exist resource vectors for each node in the path that satisfy their resource windows. Therefore, a feasible schedule is a resource-feasible path that starts in o and ends in d . Furthermore, let L_i denote the set of all labels corresponding to partial schedules where node $i \in V'$ is the last visited node.

The initialization of a label is done at shift start nodes $i \in S$ as $L_h^i = (i, T_{\min} - \psi, T_i, (V_h^1, \dots, V_h^{|V|}))$, where ψ is the dual variable associated with constraint (23) of the RMP, $T_i^{time} = d_i$, $T_i^{dur} = 0$, $T_i^{start} = d_i$, $T_i^{work} = 0$, and $T_i^{brk} = \infty$, and $V_h^j = 0$ for all nodes $j \in V$.

4.2.3. Resource Extension Functions. A resource extension function (REF) is used to extend a label (or partial schedule) with an additional vertex such that all constraints related to the scheduling problem are still satisfied. There are two options to extend label L_h^i at vertex i to vertex j when $V_h^j = 0$. The first extension executes the batch in node j directly after finishing the batch in node i without a break. The second extension starts with a break before the execution of the batch in node j . Consequently, we consider the two REFs $f(\cdot)$ and $g(\cdot)$, respectively.

The REFs $f(T_i, j)$ for the extension of label L_h^i to node j without a break define the new resource variables of resource vector T_j as follows:

$$T_j^{time} = f^{time}(T_i, j) := \max\{T_i^{time} + t_j, r_j\}, \quad (25)$$

$$T_j^{dur} = f^{dur}(T_i, j) := \max\{T_i^{dur} + t_j, r_j - T_i^{start}\}, \quad (26)$$

$$T_j^{work} = f^{work}(T_i, j) := \max\{T_i^{work} + t_j, r_j - T_i^{brk}\}, \quad (27)$$

$$T_j^{brk} = f^{brk}(T_i, j) := \min\{d_j - (T_i^{work} + t_j), T_i^{brk}\}. \quad (28)$$

Similarly, when label L_h^i is extended with a break before the order-picking task is completed as indicated by node j , the REFs $g(T_i, j)$ define the resource vector T_j as follows:

$$T_j^{time} = g^{time}(T_i, j) := \max\{T_i^{time} + t_j + l_b, r_j\}, \quad (29)$$

$$T_j^{dur} = g^{dur}(T_i, j) := \max\{T_i^{dur} + t_j + l_b, r_j - T_i^{start}\}, \quad (30)$$

$$T_j^{work} = g^{work}(T_i, j) := t_j, \quad (31)$$

$$T_j^{brk} = g^{brk}(T_i, j) := \min\{d_j - t_j, \infty\}. \quad (32)$$

Note that the resource variable T_i^{start} is never updated after it is set in the shift start node. The REF for T_j^{time} is a classic REF from the routing literature (Irnich 2008). The REFs for T_j^{dur} and T_i^{start} bear resemblance to REFs from the MTDP (Tilk and Irnich 2017). The REFs for T_j^{work} and T_j^{brk} are new and specifically designed to determine the amount of time elapsed since the last break. Desaulniers and Villeneuve (2000) use similar extension functions to estimate the cost of waiting at nodes for the shortest path problem with time windows and linear waiting costs.

The reduced cost for the partial schedule when label L_h^i is extended to node j is given by $c_h^j := \max\{T_{\min}, T_j^{dur}\} - \pi_j - \psi - \sum_{j \in \hat{B}_h} \pi_j$, where π_j is the dual value of constraints (22) for vertex j , ψ is the dual value associated with constraint (23), and $\sum_{j \in \hat{B}_h} \pi_j$ indicates the accumulated dual values associated with constraints (22) for the set of batches previously added to the partial schedule represented by the set \hat{B}_h . Note that the payment to pickers for the entire period (even if they work only for a fraction of the period) is accounted for by the use of shift end nodes, which restrict the visit to the shift end nodes at the end of a period.

The resource V_h^j is set to one to prevent vertex j from being visited again. Furthermore, $V_h^{j'}$ is also set to one for any node $j' \in V'$ that cannot be visited anymore when node j is added to the partial path because of resource constraints. The new label is then given by $L_h^j := (j, c_h^j, T_j, V_h)$, which is only feasible if the resource variables of the resource vector T_j fall within the associated resource windows.

4.2.4. Dominance. A dominance principle can be used to accelerate the solution technique by eliminating unnecessary labels. To define dominance in our pricing problem, we note that the REFs are either nondecreasing or nonincreasing, such that an element-wise comparison can be made to determine dominance (Irnich and Desaulniers 2005). A label L_h^i dominates a label $L_{h'}^i$ if both labels reside at the same vertex $i \in V'$ and if, for each feasible extension of $L_{h'}^i$ to $L_{h'}^{j'}$, there exists a feasible extension of L_h^i to L_h^j where the value of each resource with a nondecreasing (or nonincreasing) REF is less than (or larger than) or equal to the value of the resource in the extension of $L_{h'}^i$, that is, $c_h^i \leq c_{h'}^i$, $T_{i,h}^{time} \leq T_{i,h'}^{time}$, $T_{i,h}^{dur} \leq T_{i,h'}^{dur}$, $T_{i,h}^{start} \geq T_{i,h'}^{start}$, $T_{i,h}^{work} \leq T_{i,h'}^{work}$, $T_{i,h}^{brk} \geq T_{i,h'}^{brk}$, $V_h^v \leq V_{h'}^v \forall v \in V$. Consequently, the partial schedule corresponding to label $L_{h'}^i$ cannot be part of the optimal solution. Note that the differentiation of time resources T_i for label h and h' is done for comparison required by dominance. For ease of

notation, we do not use the differentiation of time resources for specific labels in the remainder of the paper.

4.2.5. Labeling Algorithm. The pricing problem is solved by embedding the resource definitions, REFs, and dominance rules in the label correction algorithm by Feillet et al. (2004). The pseudocode for the labeling algorithm is presented in online Appendix E.

4.2.6. Acceleration Strategies. Acceleration strategies are commonly used to speed up branch-and-price algorithms and are key to successfully solving sizable problems (Kallehauge et al. 2005). We propose three acceleration strategies for the pricing problem.

4.2.6.1. Initial Columns. It is known that column generation with good initial upper bounds accelerates the convergence of the linear relaxation at the root node (Desaulniers, Desrosiers, and Solomon 2002). Therefore, we first generate initial primal solutions with the savings algorithm outlined in Section 5.1. This algorithm aims to rapidly find a feasible solution. If the savings algorithm is not able to generate a feasible solution with at most p_{\max} order pickers, the initial columns for column generation are initialized with an additional artificial column that covers all batches of the problem and has an arbitrarily high cost to ensure that this artificial column will not be part of the optimal solution.

4.2.6.2. Limited Extension. To exploit the time windows and processing time information between order-picking tasks to reduce the use of REFs, we first present the following proposition. The proof of this proposition is presented in online Appendix F.

Proposition 2. *If there is an optimal schedule with two batches i and j such that $r_i \geq r_j$, $d_i \geq d_j$, $t_i < t_j$, and i precedes j in the same order picker's schedule without a break between the execution of the two batches, the execution order can be reversed with the same objective function value.*

For any partial schedule h ending with node i (i.e., presented by label L_h^i), if there is a node j which $V_h^j = 0$ and the conditions in Proposition 2 satisfy, we only have to consider the extension with a break between the execution of the batches from node i and j . Consequently, we limit the extension of resources in the arc (i, j) with the REF $g(\cdot)$ only. This particular strategy allows us to have fewer extensions and maintain a smaller set of labels while solving the pricing problem.

4.2.6.3. Limited Discrepancy Search. Desaulniers, Lessard, and Hadjar (2008) and Spliet, Dabia, and Van Woensel (2018) show that the branch-and-price

algorithm can be solved more efficiently when the pricing problem is solved with heuristics until no negative reduced costs are found (such that no new columns are added to the RMP). Along the same principle, as proposed by Feillet, Gendreau, and Rousseau (2007) and Goel and Irnich (2017), we use limited discrepancy search (LDS) to heuristically accelerate the generation of columns with a negative reduced cost.

LDS speeds up the pricing problem by maintaining a limited set of labels and heuristically removing so-called unpromising labels from the problem. In our pricing problem, labels with batches that require large waiting times and numerous breaks are considered unpromising labels. The waiting time between two nodes i and j is measured as the time window distance $TW_{distance}(i, j) := \max\{0, r_j - d_i\}$. The outgoing arcs from each node i are partitioned into two sets—good arcs and bad arcs—based on $TW_{distance}$. An additional resource (denoted as l^{bad}) is included in the label that is increased by one if a label traverses through an arc from the set of bad arcs or if it is extended with a break. Only labels that have $l^{bad} \leq \Lambda$ are extended, where the threshold Λ is called the discrepancy limit. If the LDS is unable to find any columns with a negative reduced cost, the value of Λ increases by one and the LDS is repeated. When the discrepancy limit reaches an upper bound, the LDS terminates and the ESPPRC is solved with the labeling algorithm. Additionally, an iteration of LDS is terminated if 100 columns with negative reduced cost are generated. Note that the use of LDS does not impact the optimality of the branch-and-price technique since the last pricing problem at every node of the branch-and-bound tree is solved exactly with ESPPRC.

4.3. Branching

If the pricing problem cannot find columns with a negative reduced cost and the LP solution to the RMP is not integral, a node of the branch-and-bound tree is selected for branching. Branching is done on flow variables using the best-lower-bound-first strategy (Desaulniers 2010).

A good upper-bound solution improves the efficiency of the branch-and-price technique by reducing the number of branch nodes in the search tree (Danna and Le Pape 2005). In our solution procedure, before branching from the root node, we solve the MILP of the RMP where we only consider the columns that are generated at the root node. The solution to the MILP provides the upper bound before branching. If the root node is not solved within the time limit, the MILP of the RMP is solved with the available columns to derive the best known upper-bound solution for benchmarking purposes.

5. Metaheuristic for OPSP

Given the size of real-world instances of the OPSP and the computational complexity of the problem, even the branch-and-price technique developed in the previous section is not likely to be a viable solution approach in real-life applications. In this section, we present an efficient metaheuristic that adapts the classic savings principle by Clarke and Wright (1964) to generate an initial feasible solution, and that solution is improved by a large neighborhood search (LNS) algorithm with simulated annealing (Pisinger and Ropke 2010).

5.1. Savings Algorithm

The savings algorithm iteratively combines two schedules into one schedule based on the savings principle (Clarke and Wright 1964). The procedure begins by relaxing the maximum number of order pickers constraint and creating schedules that each consist of one batch to be picked. Then, it iteratively determines the saving in terms of the labor cost that is generated when two schedules are combined into one schedule (if possible). This saving is easy to calculate. Consider that schedule h' and h'' are combined in a feasible schedule h with the corresponding compensation $m_{h'}$, $m_{h''}$, and m_h , respectively, then the savings is $(m_{h'} + m_{h''}) - m_h$. Combining batches in two schedules into one schedule has the potential to overcome inefficiencies of individual schedules when these schedules have waiting times (or breaks) between tasks or the shift length is shorter than T_{\min} time units.

To verify whether two (randomly) selected schedules can be combined into one schedule, we try to solve a simplified (or reduced) version of our OPSP, which is formulated as a MILP model in online Appendix G. Since the reduced problem finds the optimal schedule for only one order picker (or one shift) with a small number of order-picking tasks, the MILP can be solved exactly in a reasonable amount of computation time. Even though the computation time of the MILP for the reduced problem is short, a set of infeasibility checks can be performed first as a preprocessing step to easily verify whether the order-picking tasks cannot be combined in a feasible schedule. See online Appendix G.1 for the infeasibility checks. If these checks do not rule out that a feasible schedule can be found, the reduced OPSP with one order picker is solved. If no feasible solution is found, it is concluded that the two schedules cannot be combined. Otherwise, the solution of the MILP model provides the combined schedule with the largest savings (i.e., it finds the optimal sequencing of the order-picking batches).

In the classical savings algorithm by Clarke and Wright (1964), the savings of combining any given two schedules are calculated first before combining solutions in a given iteration of the algorithm. However,

in this paper, if any two randomly selected schedules can be combined in a feasible schedule and result in a savings of at least T_{\min} time units, the combined schedule is accepted immediately and the two individual schedules will not be considered for other savings in the same iteration of the savings algorithm. If no two schedules exist that can be combined in a feasible schedule that also results in sufficient savings of at least T_{\min} , all possible combinations are first calculated and then the schedules are combined such that the maximum savings is achieved. The procedure continues until no savings can be realized while combining schedules. When no further savings can be realized and the number of schedules in the solution is less than p_{\max} , a feasible solution is found that satisfies all constraints of the OPSP, and the algorithm terminates. If no feasible solution is found, the savings algorithm enters the second phase, in which the batches of any pair of schedules are chosen to be combined in a new schedule that results in the largest savings (which can be the least negative savings or additional cost) until the number of schedules equals p_{\max} .

5.2. Large Neighborhood Search for Improved Solutions

After a feasible solution for the OPSP is generated by the savings algorithm, the solution is improved with an LNS procedure. Let us denote the feasible solution at the beginning of an iteration by π , where the corresponding cost (or objective function value) is $z(\pi) := \sum_{p \in P} m_p$. This solution is destroyed and then repaired in every iteration, which results in a new feasible solution π' with cost $z(\pi')$. Furthermore, let the best found solution so far be denoted by π^* . The decision whether π' becomes the starting solution in the next iteration is based on a simulated annealing principle: if $z(\pi') < z(\pi^*)$, then $\pi^* := \pi'$ and $\pi := \pi'$; otherwise π' is accepted as new solution π with probability $e^{-(z(\pi') - z(\pi^*)) / T}$, where T is the temperature that is initialized as $T := -w \cdot z(\pi^*) / \ln(0.5)$ (Ropke and Pisinger 2006). The value is updated at the end of every iteration: $T := \rho T$, where $0 < \rho < 1$ is the cooling parameter. Consequently, it becomes less likely for worse solutions to be accepted as the starting solution in the next iteration when the number of iterations increases. If the best solution is not improved in n_T iterations, the temperature is reset to the initial value $(-w \cdot z(\pi^*) / \ln(0.5))$, such that it is more likely to explore new areas in the feasible solution space.

5.2.1. Destruction and Repair. The LNS destroys and repairs the solution π in two stages. In the first stage, two order pickers are selected. The first order picker is selected probabilistically with a roulette wheel principle based on a wastage ratio. The wastage ratio of an order

picker is the fraction of the amount of unproductive duration spent by the order picker compared with the total unproductive hours spent by all of the order pickers in the solution. The wastage ratio for order picker $p \in P$, who is assigned to complete the batches B_p with the cost m_p in solution π , is given by

$$w_p := \frac{m_p - \sum_{i \in B_p} t_i}{\sum_{p' \in P} (m_{p'} - \sum_{i \in B_{p'}} t_i)} \quad \forall p \in P. \quad (33)$$

An order picker with a higher wastage ratio is likely to be chosen as the first picker. The second order picker is randomly selected among the remaining order pickers.

In the second stage, the batches previously assigned to the two selected order pickers are reassigned to generate a new (feasible) solution π' . For this purpose, we use one of two operators with equal probability. The swap operator exchanges a random subset of batches between the two order pickers. The insert operator randomly selects a subset of batches from the first order picker and assigns them to the second order picker. In the literature, swap and insert operators are typically designed to exchange or insert one job, task, or trip at a time. The swap and insert operator in this work swaps and inserts multiple batches at a time. This allows us to generate new solutions that would otherwise require multiple operations with the traditional operators. The number of batches to swap or insert from each order picker is uniformly sampled between one and σ (which is a user set parameter). If the best solution is not improved by n_σ iterations, the value of σ is reduced by one.

After the batches are reassigned to these two order pickers, the sequencing of the batches and scheduling of shifts for the order pickers is determined by solving the same MILP of the reduced problem as in the savings algorithm (see online Appendix G). Note that we also verify whether any of the infeasibility conditions is satisfied before solving the reduced problem. Rather than directly solving a MILP, other solution techniques can be proposed to solve the reduced problem. For instance, the pricing problem in Section 4.2 can be adapted to develop a dynamic programming (DP) algorithm by creating a new graph for each reduced problem, in which only the relevant batches assigned to a picker are included and a path starting at the dummy source (i.e., node o) has to visit all batch nodes in the graph before returning to the dummy sink (i.e., node d). At the dummy sink, the solution with the cheapest cost is selected and returned to the metaheuristic for evaluation. In limited numerical experiments, this DP approach to solve the reduced problem produced the same results as the MILP approach but with shorter computation times. However, the development of a DP algorithm requires labels,

REFs, dominance rules, and acceleration techniques that need to be tailored to solve a specific reduced problem. If there would be additional restrictions in the original problem formulation (such as an upper bound on the ratio between flexible to full-time order pickers, a maximum number of order pickers at any time or specific time windows for different types of breaks), these components of the DP algorithm need to be redefined. In contrast, the MILP approach is able to address different variations of the original problem without the need to change the code (see online Appendix D). To accommodate flexibility in our solution approach and easily adjust to different warehouse environments, we present the LNS that uses the MILP approach to solve the reduced problem.

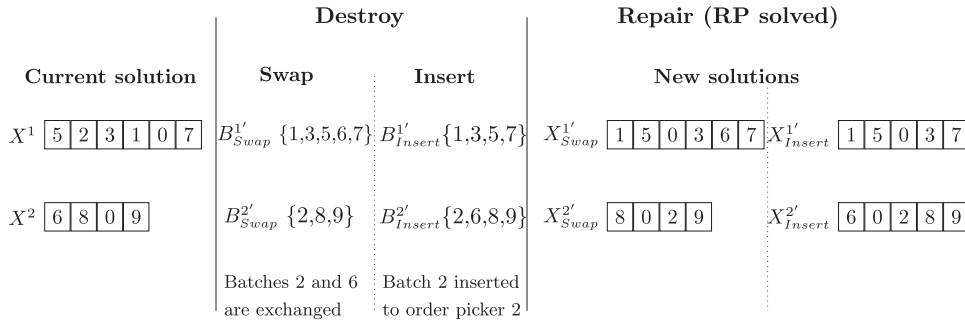
Figure 4 illustrates how the two destroy operators work based on a simple example. The initial schedules of the two selected order pickers are represented by X^1 and X^2 . With the swap operator, batch 2 and batch 6 are interchanged. The new batches assigned to the order pickers are indicated by $B_{Swap}^{1'}$ and $B_{Swap}^{2'}$, respectively. With the insert operator, batch 2 is unassigned from the first order picker and assigned to the second order picker. The new batches assigned to the order pickers are then indicated by $B_{Insert}^{1'}$ and $B_{Insert}^{2'}$, respectively. After solving the MILP as formulated in online Appendix G for each of the two order pickers individually, we obtain the new schedules $X^{1'}$ and $X^{2'}$, respectively.

The LNS terminates if $z(\pi^*)$ does not exceed the lower bound formulated in Equation (10) (see online Appendix B), if the number of iterations exceeds a maximum threshold or if the runtime exceeds a maximum threshold. Once the LNS terminates and time is available, we pass the LNS solution to the branch-and-price algorithm to improve the solution further by solving the pricing problem for one iteration without solving the ESPPRC exactly.

6. Results

This section presents a numerical comparison of the branch-and-price algorithm (Section 4), savings algorithm (Section 5.1), and metaheuristic (Section 5.2) to solve the OPSP. Since state-of-the-art commercial solvers such as Gurobi 9.0.1 (Gurobi Optimization 2020) cannot generate an optimal solution for even the smallest instances, and the branch-and-price algorithm outperforms Gurobi without exception, we do not report the performance of such commercial solvers here. See online Appendix H for a comparison of the performance of commercial solver Gurobi and the branch-and-price algorithm. Furthermore, no existing solution procedures from the literature are included as benchmark since the authors are not aware of

Figure 4. Illustration of the Destroy Operations in an Iteration of the LNS Algorithm, Where 0 in a Schedule Represents a Break



any other work that makes the same (or even similar) decisions and the objective function (see also Section 2).

All solution procedures are implemented in C++ and run on an i7 3.60 GHz machine with 16 gigabytes of RAM. For the parameters of the branch-and-price algorithm, the maximum number of good arcs from any node is set to two and the number of increments for the discrepancy limit in LDS (i.e., Λ) is set to 10. The parameter values for the metaheuristic are guided by the literature, where $\rho := 0.95$ and $n_T := 200$ (Stenger et al. 2013; Bodnar, de Kostner, and Azadeh 2017). The initial value of σ is set to 4 and n_σ to 1,250. Furthermore, $w := 0.1$ produced the best results in our numerical experiments, but we cannot guarantee optimality of this parameter value. The stopping criterion for the branch-and-price method is set to 1,800 seconds. For the metaheuristic, it is set to 360 seconds or 5,000 iterations (whichever comes first) to ensure that the method is suitable for practical applications.

6.1. Instances in Numerical Testbed

The instances are generated to mimic the operations of a retail grocery warehouse for which we had detailed data where the target departure times of the outbound trucks determine the staging lane operations as well as their earliest and latest due times. For all instances, we consider a 24-hour time horizon and we use minutes as our time unit. The warehouse operates 24 hours a day, seven days a week. However, all order-picking tasks and shifts of order pickers are disjoint between different planning horizons as all shifts in a day start and end between 11 p.m. and 11 p.m. the next day.

6.1.1. Due Time Windows. Two patterns of due time windows are considered in our instances: waved and waveless. In waved instances, trucks arrive at the staging lanes at the same time and depart from the staging lanes at the same time (i.e., batches to be picked in the same wave have the same due time windows). Alternatively, in the waveless operations, the arrival and departure times of trucks at different staging lanes are not related. The deadline for each truck departure

from a staging lane is taken from a uniform distribution in the range of 120–1,425 minutes.

To make sure that there is sufficient time for the staging and loading operations of a truck, we push back deadlines (if needed) to guarantee at least 30 minutes between two consecutive departure due times of batches destined for the same dock door (or staging lane). The earliest due time of a batch is set to the latest due time of the previous batches at the same staging lane plus 15 minutes to ensure that loads of different trucks are not mixed up, and previous trucks have finished loading. The earliest due time of the first batch that is due at a staging lane is set to zero. The number of staging lanes in the instances varies from one to eight (see Section 6.1.4).

6.1.2. Processing Time Distributions. The processing times of batches are taken either from one of the following uniform distributions— $U[30, 60]$, $U[60, 90]$ or $U[90, 120]$ —or from an exponential distribution with the same corresponding average (i.e., 45, 75, or 105 minutes, respectively). The maximum processing time of a batch is restricted to 330 minutes to ensure that employees do not violate the break constraint ($T_{break} = 330$ minutes (see Section 6.1.3)).

6.1.3. Shift Types. In accordance with Dutch and European working hours laws, the maximum shift length to employ an order picker (i.e., T_{max}) is 540 minutes (or nine hours), and the maximum time duration that an employee can work without a break (i.e., T_{break}) is 330 minutes (or 5.5 hours). The length of the break (i.e., l_b) has to be at least 45 consecutive minutes (European Parliament, Council of the European Union 2003).

We consider six shift structures. In shift structures SStr1, SStr2, and SStr3, shifts can start every eight hours and T_{min} equals eight, six, and four hours, respectively. In shift structures SStr4, SStr5, and SStr6, shifts can start every four hours and T_{min} equals eight, six, and four hours, respectively. In all shift structures, a shift can end at the end of any hour after T_{min} . Consequently, shift structure SStr1 is the most restrictive

and SStr6 is the most flexible. Table 3 summarizes the six shift structures we consider.

6.1.4. Number of Batches and Staging Lanes. Each outbound truck requires exactly four order batches to be picked and the number of trucks departing the warehouse in the planning horizon equals 10, 20, or 40 trucks. This results in instances with 40, 80, or 160 batches to be picked, respectively. The number of staging lanes in an instance is chosen such that the number of departures per staging lane is fixed at five, 10, or 20 trucks. As a result, the number of staging lanes ranges from one and eight lanes. Note that instances with 10 trucks can only have five or 10 departures per staging lane. Furthermore, for instances where the number of trucks equals the number of truck departures in a staging lane, there is only one staging lane (grouped under “waved” in Table 4). We assume that sufficient order pickers are available to schedule with $p_{\max} = 100$.

6.2. Algorithmic Performance

Table 4 summarizes the results over all 504 instances in the testbed, whereas the results for the individual instances are presented in online Appendix H. For the branch-and-price algorithm, *Root solved* indicates the number of instances for which the column generation was able to solve the linear relaxation within the runtime limit of 1,800 seconds. *Optimal solution* indicates the number of instances for which the optimal solution was found within this time limit. For those instances where the branch-and-price algorithm was not able to find the optimal solution, *Optimality gap %* presents the average relative percentage cost difference between the best lower bound found after branching and the best integer solution found after branching. The average time required to solve the root node and the overall branch-and-price algorithm is indicated by CPU^{LP} and CPU^{BP} , respectively. Note that CPU^{BP} also includes the time to generate an initial solution. The average relative performance gap between the solution generated by the savings algorithm and metaheuristic compared with the best branch-and-price solution is indicated by $\% \Delta^S$ and $\% \Delta^{MH}$, respectively,¹ where a positive number indicates that the branch-and-price

algorithm found a better solution. The average computation time of the savings algorithm and metaheuristic is indicated by CPU^S and CPU^{MH} , respectively.

Table 4 shows that the branch-and-price algorithm is capable of solving reasonable size instances. However, the size of the instances adversely affects the performance of the exact approach. For the instances with 40, 80, and 160 batches, the root node can be solved in 100%, 58.9%, and 26.3% of the instances, respectively, and the algorithm converges to an optimal solution within the runtime for 60.2%, 36.1%, and 19.4% of the instances, respectively. For the instances where the branch-and-price algorithm is not able to find an optimal solution, the average optimality gap is only 3.5%. Figure 5(a) shows the average optimality gap of the branch-and-price solutions for instances where we were able to solve the root node but the optimal solutions were not obtained.

When we compare the number of instances for which the root node (i.e., the linear relaxation) is solved and the number of instances for which an optimal solution is found within the runtime limit of 1,800 seconds, we make the following observations. First, waveless instances are more difficult to solve than waved instances. A reason why the branch-and-price algorithm can solve waved instances easier is because the limited extension property (see Proposition 2) exploits the fact that the batches have nonoverlapping due time windows when solving the pricing problem. As a result, the labeling algorithm does not have to explore as many extensions between nodes, and it is capable of solving the pricing problem more efficiently for waved instances. Second, instances with exponentially distributed processing times are more difficult to solve than instances with uniformly distributed processing times. Instances with exponentially distributed processing times have many batches with short processing times. On average, the number of tasks that can be assigned to an order picker is higher with the exponentially distributed processing times. As a result, the labeling algorithm has to consider more potential solutions and labels when solving the pricing problem. Third, instances with more truck departures per staging lane are easier to solve than instances with fewer truck departures. When there are more trucks departing from the same staging lane, the average length of the due time windows is smaller (see Figure 3 in online Appendix H). As a result, the pricing problem needs to consider fewer extensions from any node as many potential solutions are not feasible. See online Appendix H for a more detailed discussion of these observations.

The savings algorithm is able to quickly generate a feasible solution (on average within 4.3 seconds) for either the branch-and-price algorithm or the metaheuristic. However, the quality of these solutions

Table 3. Shift Structures Considered in Our Numerical Experiments

Shift structure	Starting hours (S)	T_{\min} (hours)
SStr 1	0, 8, 16	8
SStr 2	0, 8, 16	6
SStr 3	0, 8, 16	4
SStr 4	0, 4, 8, 12, 16, 20	8
SStr 5	0, 4, 8, 12, 16, 20	6
SStr 6	0, 4, 8, 12, 16, 20	4

Table 4. Summary of Results

Dep. per lane	Instance type	Batches	Branch-and-price algorithm			Savings algorithm		Metaheuristic				
			Number of instances		Optimality gap %	Average runtime		$\% \Delta^S$	$\% \Delta^{MH}$	CPU^{MH} (sec.)		
			Root solved	Optimal solution		CPU^{LP} (sec.)	CPU^{BP} (sec.)				CPU^S (sec.)	
5	Unif- Waved	40	18/18	11/18	4.00	10.5	809.7	11.6	2.5	0.2	48.2	
		80	13/18	9/18	1.7	671.7	1,016.1	12.5	8.5	0.0	234.3	
		160	3/18	1/18	0.6	1,601.5	1,758.4	12.1	33.8	0.7	361.4	
	Unif- Waveless	40	18/18	10/18	5.0	54.9	839.2	17.9	2.3	0.5	110.3	
		80	6/18	3/18	1.1	1,489.8	1,571.5	20.3	7.6	0.2	322.0	
		160	1/18	0/18	0.8	1,956.9	1,800.0	17.4	26.2	1.9	362.2	
	Exp- Waved	40	18/18	8/18	3.7	73.7	1,125.7	8.3	2.7	0.0	101.0	
		80	4/18	2/18	1.5	1,496.3	1,603.9	10.5	9.0	0.0	302.8	
		160	0/18	0/18	—	1,788.9	1,800.0	15.0	37.3	0.6	361.5	
	Exp- Waveless	40	18/18	13/18	3.3	76.4	843.1	14.0	2.2	0.7	147.1	
		80	4/18	0/18	1.4	1,637.2	1,800.0	14.7	7.9	-0.3	348.6	
		160	0/18	0/18	—	1,889.1	1,800.0	12.9	28.3	1.1	363.3	
10	Unif- Waved	40	18/18	16/18	2.1	1.7	204.5	12.2	1.8	0.6	30.7	
		80	18/18	14/18	1.9	142.2	492.3	14.2	6.7	0.1	103.8	
		160	14/18	11/18	0.4	507.2	785.1	11.3	26.1	0.2	210.0	
	Unif- Waveless	80	12/18	6/18	2.5	759.8	1,278.9	18.8	6.2	-0.1	239.4	
		160	6/18	5/18	0.3	1,402.5	1,447.3	17.8	21.9	0.3	336.3	
		Exp- Waved	40	18/18	7/18	5.2	22.3	1,102.7	11.5	1.8	0.1	64.9
	Exp- Waveless	80	12/18	8/18	3.2	758.2	1,030.2	15.0	6.5	0.4	205.0	
		160	2/18	1/18	0.7	1,654.7	1,702.8	16.4	25.9	0.8	333.4	
		80	8/18	4/18	1.3	1,134.5	1,497.7	17.4	6.0	0.3	272.0	
	20	Unif- Waved	80	17/18	14/18	1.2	108.1	410.0	13.0	4.9	0.0	66.4
			160	14/18	11/18	0.7	486.1	782.6	12.8	18.9	0.0	175.1
			Unif- Waveless	160	5/18	5/18	—	1,467.0	1,447.9	23.8	17.9	0.6
Exp- Waved		80	12/18	5/18	1.7	671.1	1,316.7	16.5	5.0	0.0	215.0	
		160	5/18	3/18	0.3	1,534.9	1,527.3	15.8	19.8	0.6	343.1	
		Exp- Waveless	160	6/18	5/18	2.8	1,476.7	1,417.4	16.4	18.0	0.4	335.8

Note: Optimality gap % with “—” indicates that lower bound is not available for instances with nonoptimal solutions for these set of instances.

is poor, with an average cost deviation of 15.0% compared with the best solutions found with the branch-and-price algorithm. In contrast, the solutions with the metaheuristic have an average performance gap of less than 0.4%, which is found within less than one-fifth of the computational time required for the branch-and-price algorithm. Figure 5, (b) and (c), present the performance gap of the heuristic procedures and the computational time for each of the three solution approaches, respectively.

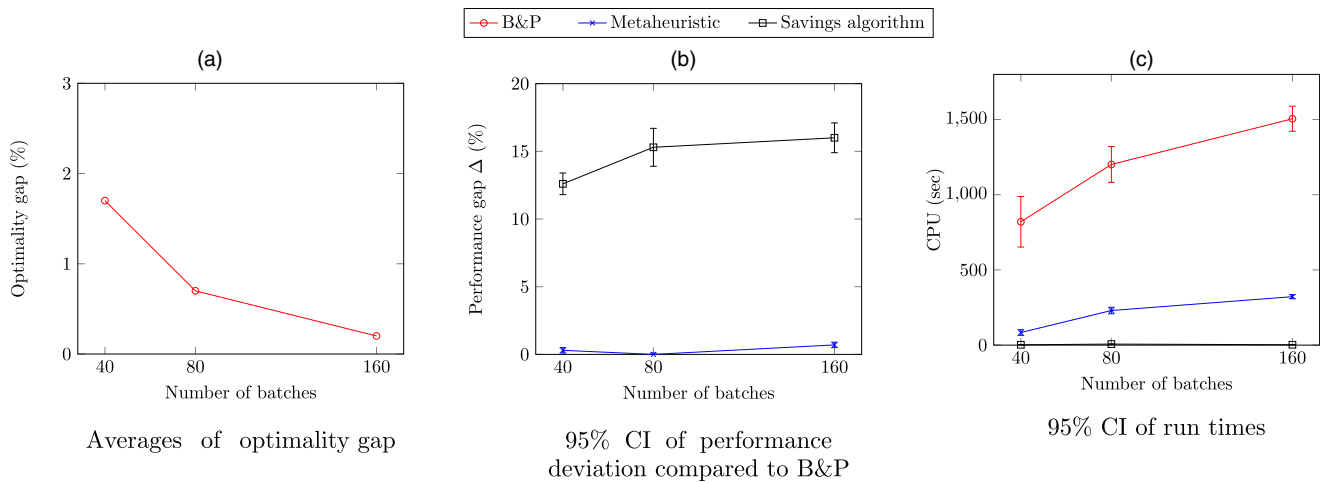
6.3. Flexible Shift Structures: A Case Study

In this subsection, we apply the metaheuristic to the OPSP at a warehouse with perishable products of a Dutch grocery retailer. The case study serves two purposes. First, it evaluates the usability of the metaheuristic that we propose to solve industrial instances. Second, the case illustrates some of the ways that the methodology in this paper can be used to evaluate warehouse operating policies of interest to managers. In particular, we study the impact of the shift structures

on the number of order pickers scheduled to perform the order-picking activities.

6.3.1. Description Instances. The retailer provided operational data regarding the processing times and due time windows of batches for two weeks of its operations. The first week represents a typical week in terms of the number of batches to be picked and shipped from the warehouse. The second week represents the busiest week of the year, which occurs during the Christmas season. There are 6.9% more batches to be picked in the busier week than in the typical week (see Figure 6(a), where day 1 is a Sunday). The warehouse has 53 staging lanes, and the number of trucks departing from the warehouse ranges from 128 to 227 trucks per day (see Figure 6(b)). When we consider the number of batches with a due deadline in a particular hour in Figure 6(c), we identify two peak periods of operations: between hour 5 and hour 7, and between hour 10 and hour 12. In this figure, hour 0 corresponds to 11:00 p.m. since

Figure 5. (Color online) Performance Comparison Between Solutions Found with the Branch-and-Price Algorithm, Savings Algorithm, and Metaheuristic



the warehouse starts its order-picking activities at that hour. The average processing time of a batch is around 41 minutes for both the busy and normal week, and the distribution of these processing times are similar in both weeks (see Figure 7(a)). The distribution of the duration of the due time windows is illustrated in Figure 7(b). The larger due time windows in the right tail in this figure occur on days with fewer trucks departing from the warehouse (i.e., on day 1).

6.3.2. Current Shift Structure. The employees are hired to work at the warehouse through third-party agencies. Their shifts can start at hour 0, 8, and 9 (i.e., at 11:00 p.m., 7:00 a.m., and 8:00 a.m.). The flexible workers are allowed to work for at most nine hours (i.e., $T_{\max} = 9$ hours) and are compensated for at least six hours

(i.e., $T_{\min} = 6$ hours). In contrast to our MILP formulation in Section 3, the order pickers receive three breaks at fixed times after they start their shift: a 15-minute break after two hours, a 30-minute break after 3.5 hours, and another 15-minute break after six hours. This shift structure is compliant with the EU and Dutch labor laws.

The warehouse manager has to determine the number of order pickers to schedule for each of the three shift start times, the shift duration of each order picker, as well as the batches to be picked by each order picker. Currently, these decisions are made based on experience and intuition of warehouse managers. Due to data privacy concerns, the retailer was not willing to share the actual order picker schedules.

There are four interesting research questions in our case study with the retailer: (i) Can the metaheuristic

Figure 6. (Color online) The Workload in the Case Study

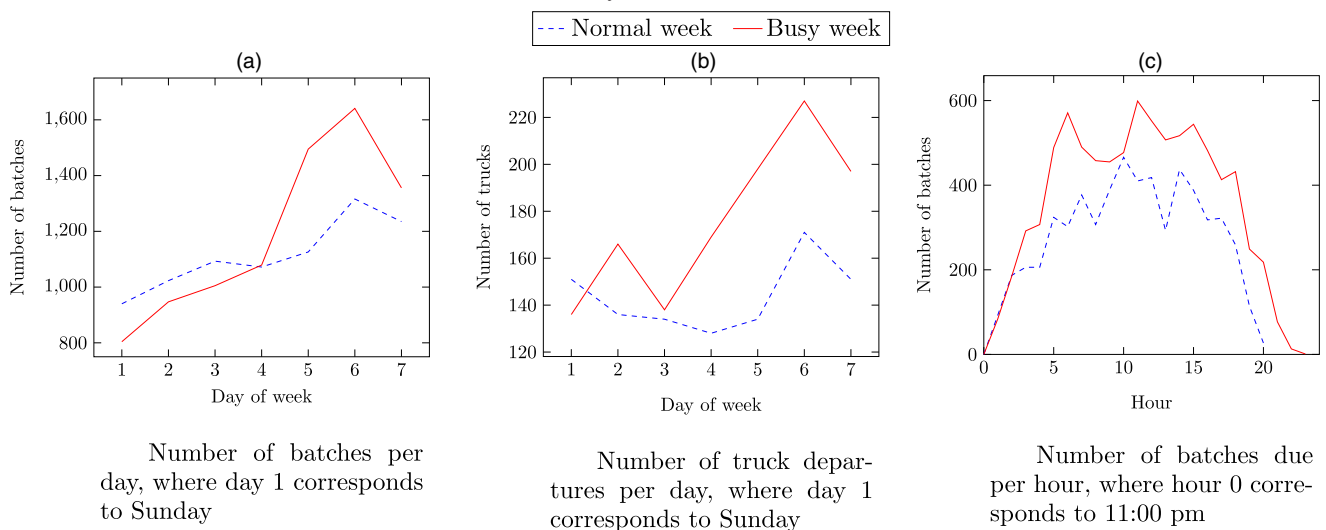
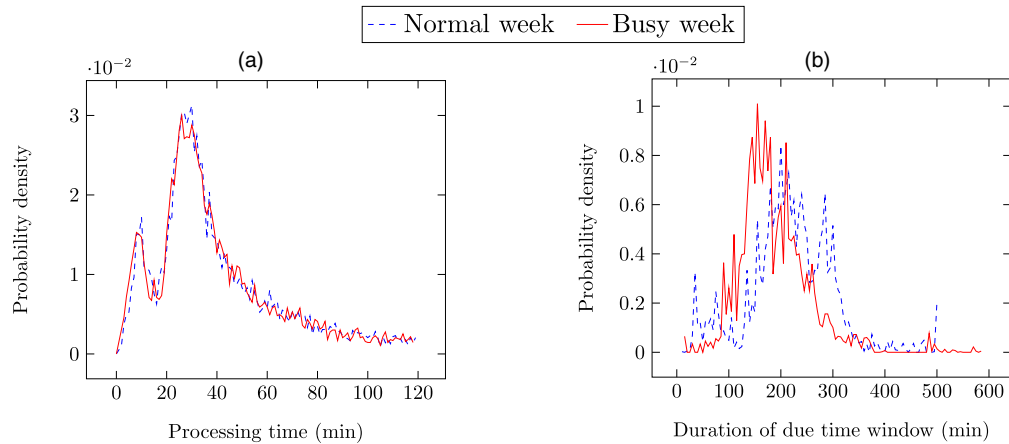


Figure 7. (Color online) Variability in the Processing Times and Durations of Due Time Windows for Picking Batches



Distribution of processing times for picking batches

Distribution of the due time window durations to pick and deliver batches

that is developed in Section 5 be used in practice as a decision support tool? (ii) What is the value of flexible break times rather than fixed break times (that are currently used by the retailer)? (iii) What is the value of an additional shift start time? (iv) Can the retailer leverage flexible break times and an additional shift start time to offer a larger minimum compensation T_{\min} without incurring higher labor costs? The last question especially is of interest to the retailer since the retailer believes that a larger minimum compensation helps to foster better working relationships with order pickers and improve the retention rates of employees.

To answer these questions, we first conducted multiple rounds of consultation with the planners and managers to develop plausible and actionable scenarios. The scenarios can be distinguished along three dimensions. First is flexible break times. This means that the breaks for order pickers are scheduled at the current break start times ± 15 minutes. Second, an

additional shift start time is introduced at hour 4 to account for the workload peak, as illustrated in Figure 6(c). We have also tried an additional start time at hour 3 and hour 5, but an additional shift start time at hour 4 resulted in the lowest objective function values. Third, the minimum compensation time can be increased to seven hours or even eight hours instead of six hours. Additionally, we introduce two shift structures that are in line with Section 3: a break of 20 minutes needs to be scheduled after at most two hours of work (i.e., $T_{\text{break}} = 2$ hours and $l_b = 20$ minutes). The minimum compensation time (i.e., T_{\min}) is still six hours. This shift structure is comparable to the current shift structure in the sense that an employee is compensated for either two or three breaks in any shift, and the values of T_{\min} and T_{\max} are the same. An overview of the 12 shift structure scenarios is provided in Table 5. Scenario 1 corresponds to the current shift structure, which serves as benchmark. Since the shift structure at the retailer is different than

Table 5. Shift Structure Scenarios to Analyze

Shift structure	Description	Flexible break times	Additional shift start	T_{\min} (hours)
Scenario 1	Current scenario (base case)			6
Scenario 2	Flexible breaks and $T_{\min} = 6$ hours	✓		6
Scenario 3	Flexible breaks and $T_{\min} = 7$ hours	✓		7
Scenario 4	Flexible breaks and $T_{\min} = 8$ hours	✓		8
Scenario 5	Extra shift and $T_{\min} = 6$ hours		✓	6
Scenario 6	Extra shift and $T_{\min} = 7$ hours		✓	7
Scenario 7	Extra shift and $T_{\min} = 8$ hours		✓	8
Scenario 8	Flexible breaks, extra shift, and $T_{\min} = 6$ hours	✓	✓	6
Scenario 9	Flexible breaks, extra shift, and $T_{\min} = 7$ hours	✓	✓	7
Scenario 10	Flexible breaks, extra shift, and $T_{\min} = 8$ hours	✓	✓	8
Scenario 11	Theoretical breaks and $T_{\min} = 6$ hours	$T_{\text{break}} = 2$ hours, $l_b = 20$ minutes		6
Scenario 12	Theoretical breaks, extra shift, and $T_{\min} = 6$ hours	$T_{\text{break}} = 2$ hours, $l_b = 20$ minutes	✓	6

discussed in Section 3, we adapted the reduced problem of the metaheuristic to consider flexible break times (see online Appendix I for details).

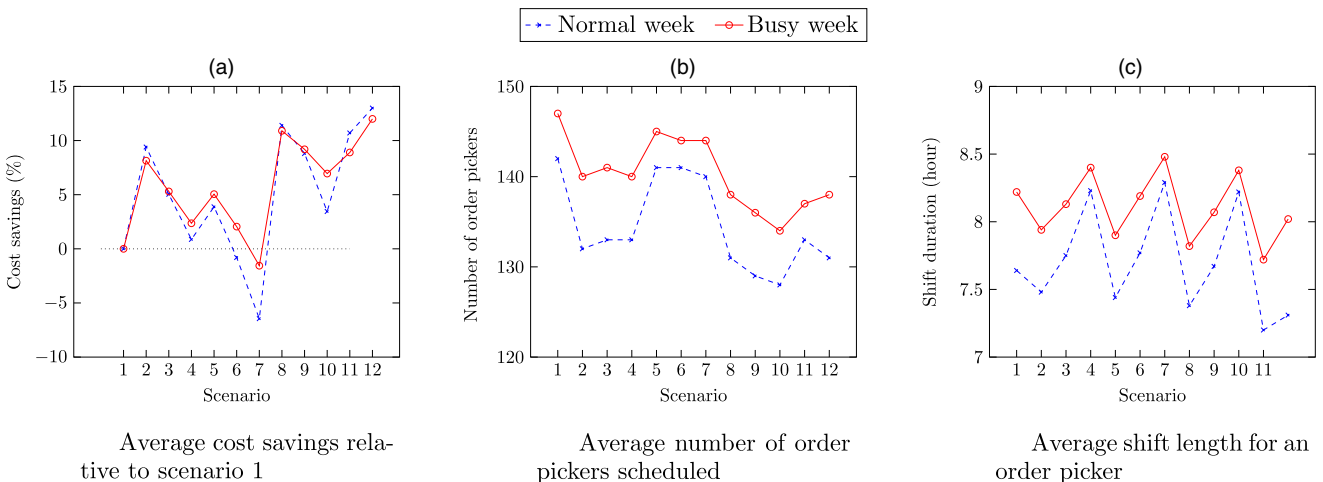
The overall cost savings as well as the impact on the average number of scheduled order pickers and the average shift length are presented in Figure 8, (a), (b), and (c), respectively; detailed results are presented in online Appendix J. By allowing 15 minutes of flexibility in the break times, the labor cost savings for the retailer are on average 8.8% (comparing scenario 2 to the base case of scenario 1). In particular, fewer employees have to be scheduled and the average shift length decreases as well. Example schedules under scenarios 1, 2, 5, and 11 are presented in online Appendix J. When the minimum compensation time is increased from six hours to seven or eight hours (i.e., scenario 3 and 4), the retailer can still expect to have an average cost saving of 5.2% and 0.7%, respectively, by adopting flexible break times. The number of employees to schedule remains similar in scenarios 2, 3, and 4, however, the average shift length increases. Interestingly, the average shift length in scenario 3 is comparable to scenario 1, that is, the cost savings of 5.2% in scenario 3 are mainly due to the scheduling of fewer order pickers. Increasing the minimum compensation time to eight hours (in scenario 4) still results in cost savings. This is good news for the retailer, since the additional cost of an increased value of T_{\min} is offset against 15 minutes of flexibility in the break start times. Allowing even more flexibility in scheduling breaks (in scenario 11), the average labor cost can decrease an additional 1% (the cost savings in scenario 11 is 9.8%, whereas it is 8.8% in scenario 2). However, this is considered not favorable by the retailer since there is less overlap between the breaks of employees in scenario 11

(see also online Appendix J), which is of social importance for the employees. Since the majority of the cost savings in scenario 11 are also captured by allowing 15 minutes of flexibility in the break times (as in scenario 2), these results provided sufficient motivation to initiate implementing this 15 minutes of flexibility.

The average cost savings of an additional shift start time at hour 4 is less substantial compared with flexible break times: 4.5% when T_{\min} equals six hours, only 0.6% when T_{\min} equals seven hours, and an average cost increase of 4% when T_{\min} equals eight hours (for scenarios 5, 6, and 7, respectively). This is mainly because the number of order pickers that need to be scheduled decreases significantly less compared with flexible break times, whereas the average shift lengths are comparable.

When combining flexible break times and adding a shift start time at hour 4, the average labor cost can (obviously) decrease even further. What is interesting to observe is that the number of order pickers that are scheduled is actually decreasing as the minimum compensation time T_{\min} increases from six to seven hours and from seven to eight hours (comparing scenario 8, 9, and 10). Since the increase in average shift length is similar to that in the previous case, the marginal decrease in average cost savings is less when T_{\min} increases. The corresponding average cost savings in these scenarios are 11.1%, 9.0%, and 5.2%, respectively. Finally, we observe that most of the cost savings of the flexible break times are captured by the 15-minute flexibility of the break times, since the cost savings in scenario 8 and 12 correspond to 11.1% and 12.5%, respectively (similar when comparing the cost savings between scenario 2 and 11). This reinforces our previous conclusion that it is sufficient to include only 15 minutes of flexibility when scheduling the break times.

Figure 8. (Color online) Performance of the Different Shift Structure Scenarios in the Case Study



7. Conclusion

In this paper, we study the OPSP where order-picking tasks can be done flexibly but are constrained with due time windows. The problem intersects with the personnel scheduling literature. However, unlike the available literature, our problem minimizes the labor cost while considering the minimum promised pay to order pickers, the shift start and end times of employees, and break times. Therefore, break times are explicitly included as scheduling variables as well as shift start and end times (with a minimum compensation time for each order picker). This is a common problem at warehouses with manual order pickers where batches of items need to be picked and delivered to outbound dock doors (or staging lanes) within the time windows that the trucks are scheduled to load the items. Since it combines the order picker planning problem and the shift scheduling problem, we call this the order picker scheduling problem. We present several formulations of the problem with a range of operational restrictions that are important to consider. Two methods are presented to solve the problem. First, an exact branch-and-price algorithm is developed. Since this algorithm can be prohibitive for practical applications, we also present an efficient metaheuristic that combines a savings algorithm and LNS. The results indicate that the heuristic has a stable performance and is capable of producing near-optimal solutions in a reasonable time for real-life instances.

In a case study, we show how the problem and solution approaches can be used to study different shift structures. In particular, the results show that the retailer can readily increase the minimum compensation duration for workers from six to seven or eight hours and still realize average labor cost savings of 5.2% or 0.7%, respectively, when a 15-minute flexibility in the scheduling of break times is implemented. By increasing the minimum compensation duration, order pickers might experience improved job satisfaction to promote job retention. More cost savings of around 4–4.5% can be achieved when an additional shift start time is introduced. Inspired by the result, the retailer under study has decided to implement additional shifts and flexible breaks. Moreover, the findings are applicable beyond this grocery retailer as most retailers in Western Europe operate their warehouses constrained by staging time windows with flexible order pickers in a similar manner.

For the sake of brevity, we only consider identical order pickers in our study. Since the evaluation of a schedule in both our solution approaches to the problem is on the individual employee, order picker specific characteristics such as age and seniority-based breaks as well as restricted and preferred shift starting times can be added to the pricing problem

for the branch-and-price algorithm and the reduced problem for the metaheuristic.

Furthermore, we assumed that shifts and order-picking tasks are nonoverlapping between different planning horizons (i.e., the shift start and end times of every shift are in the same planning horizon when batch orders can be picked). If this assumption were to be relaxed, we propose to use our solution methodology with a rolling horizon. In particular, we suggest extending the planning horizon with an additional time period during which no new shifts are allowed to start but employees who started their shift in the original planning horizon can finish their shift in the extended time period and perform order-picking tasks during that time period. Consequently, order pickers can be scheduled more efficiently at the end of the planning horizon and order-picking tasks in the next planning horizon can be performed already. Such a rolling horizon approach results in feasible solutions that may not be optimal as there may not be sufficient order-picking tasks available for the order pickers that start their shift in the next planning horizon. A shift scheduling problem that can dynamically include arrivals of new orders in an online environment could be of significant value for e-commerce companies.

Future research can take two additional trajectories within the offline retail environment. First, given the size of the instances in real-life business applications, order batching decisions are made a priori (similar to our approach). It can be worthwhile to jointly consider the order batching and shift scheduling problem. Second, we assume norm times to perform the order-picking activities in a deterministic manner. A compelling research direction would be to consider robust OPSPs with stochastic processing times of batches. These two research directions can be of significant value to both academia and practice.

Endnote

¹The performance metrics $\% \Delta^S = (z(S) - z(BP)) / z(BP) \times 100$ and $\% \Delta^{MH} = (z(MH) - z(BP)) / z(BP) \times 100$, where $z(BP)$, $z(S)$, and $z(MH)$ denote the objective function value of the best integer solution found by the branch-and-price algorithm, savings algorithm, and metaheuristic, respectively.

References

- Alsheddy A, Tsang EP (2011) Empowerment scheduling for a field workforce. *J. Scheduling* 14(6):639–654.
- Aykin T (1996) Optimal shift scheduling with multiple break windows. *Management Sci.* 42(4):591–602.
- Aykin T (2000) A comparative evaluation of modeling approaches to the labor shift scheduling problem. *Eur. J. Oper. Res.* 125(2): 381–397.
- Azadeh K, De Koster R, Roy D (2019) Robotized and automated warehouse systems: Review and recent developments. *Transportation Sci.* 53(4):917–945.

- Bard JF, Morton DP, Wang YM (2007) Workforce planning at USPS mail processing and distribution centers using stochastic optimization. *Ann. Oper. Res.* 155(1):51–78.
- Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 46(3):316–329.
- Bechtold SE, Jacobs LW (1990) Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Sci.* 36(11):1339–1351.
- Beliën J, Demeulemeester E (2008) A branch-and-price approach for integrating nurse and surgery scheduling. *Eur. J. Oper. Res.* 189(3):652–668.
- Bertsimas D, Chang A, Mišić VV, Mundru N (2019) The airlift planning problem. *Transportation Sci.* 53(3):773–795.
- Bhandari A, Scheller-Wolf A, Harchol-Balter M (2008) An exact and efficient algorithm for the constrained dynamic operator staffing problem for call centers. *Management Sci.* 54(2):339–353.
- Bodnar P, De Koster R, Azadeh K (2017) Scheduling trucks in a cross-dock with mixed service mode dock doors. *Transportation Sci.* 51(1):112–131.
- Boysen N, Briskorn D, Emde S (2017) Sequencing of picking orders in mobile rack warehouses. *Eur. J. Oper. Res.* 259(1):293–307.
- Brusco MJ, Jacobs LW (1998) Personnel tour scheduling when starting-time restrictions are present. *Management Sci.* 44(4):534–547.
- Brusco MJ, Jacobs LW (2000) Optimal models for meal-break and start-time flexibility in continuous tour scheduling. *Management Sci.* 46(12):1630–1641.
- Chen T-L, Cheng C-Y, Chen Y-Y, Chan L-K (2015) An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. *Internat. J. Production Econom.* 159:158–167.
- Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* 12(4):568–581.
- Contardo C, Desaulniers G, Lessard F (2015) Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks* 65(1):88–99.
- Cordeau J-F, Laporte G, Pasin F, Ropke S (2010) Scheduling technicians and tasks in a telecommunications company. *J. Scheduling* 13(4):393–409.
- Côté M-C, Gendron B, Rousseau L-M (2011) Grammar-based integer programming models for multiactivity shift scheduling. *Management Sci.* 57(1):151–163.
- Dahmen S, Reikik R, Soumis F (2018) An implicit model for multi-activity shift scheduling problems. *J. Scheduling* 21(3):285–304.
- Danna E, Le Pape C (2005) Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, New York), 99–129.
- Dantzig GB (1954) Letter to the editor—A comment on Edie’s “Traffic delays at toll booths”. *J. Oper. Res. Soc. Amer.* 2(3):339–341.
- Dellaert N, Dashty Saridarq F, Van Woensel T, Crainic TG (2018) Branch-and-price-based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Sci.* 53(2):463–479.
- Desaulniers G (2010) Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Oper. Res.* 58(1):179–192.
- Desaulniers G, Villeneuve D (2000) The shortest path problem with time windows and linear waiting costs. *Transportation Sci.* 34(3):312–319.
- Desaulniers G, Desrosiers J, Solomon MM (2002) Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. Ribeiro CC, Hansen P, eds. *Essays and Surveys in Metaheuristics* (Kluwer, Boston), 309–324.
- Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Sci.* 42(3):387–404.
- Dror M (1994) Note on the complexity of the shortest path models for column generation in VRPTW. *Oper. Res.* 42(5):977–978.
- Edie LC (1954) Traffic delays at toll booths. *J. Oper. Res. Soc. Amer.* 2(2):107–138.
- Elahipanah M, Desaulniers G, Lacasse-Guay E (2013) A two-phase mathematical-programming heuristic for flexible assignment of activities and tasks to work shifts. *J. Scheduling* 16(5):443–460.
- Elsayed E, Lee M-K (1996) Order processing in automated storage/retrieval systems with due dates. *III Trans.* 28(7):567–577.
- Elsayed E, Lee M-K, Kim S, Scherer E (1993) Sequencing and batching procedures for minimizing earliness and tardiness penalty of order retrievals. *Internat. J. Production Res.* 31(3):727–738.
- Ernst AT, Jiang H, Krishnamoorthy M, Owens B, Sier D (2004a) An annotated bibliography of personnel scheduling and rostering. *Ann. Oper. Res.* 127(1–4):21–144.
- Ernst AT, Jiang H, Krishnamoorthy M, Sier D (2004b) Staff scheduling and rostering: A review of applications, methods and models. *Eur. J. Oper. Res.* 153(1):3–27.
- European Parliament, Council of the European Union (1991) Council Directive 91/533/EEC of October 14, 1991 on an employer’s obligation to inform employees of the conditions applicable to the contract or employment relationship, <https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A31991L0533>.
- European Parliament, Council of the European Union (2003). Directive 2003/88/EC of the European Parliament and of the Council of November 4, 2003 concerning certain aspects of the organisation of working time, <http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32003L0088>.
- European Parliament, Council of the European Union (2019) Directive (EU) 2019/1152 of the European Parliament and of the Council of June 20, 2019 on transparent and predictable working conditions in the European Union, <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019L1152>.
- Feillet D, Gendreau M, Rousseau L-M (2007) New refinements for the solution of vehicle routing problems with branch and price. *INFOR: Inform. Systems Oper. Res.* 45(4):239–256.
- Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3):216–229.
- Firat M, Hurkens C (2012) An improved MIP-based approach for a multi-skill workforce scheduling problem. *J. Scheduling* 15(3):363–380.
- Fischetti M, Martello S, Toth P (1987) The fixed job schedule problem with spread-time constraints. *Oper. Res.* 35(6):849–858.
- Fischetti M, Martello S, Toth P (1989) The fixed job schedule problem with working-time constraints. *Oper. Res.* 37(3):395–403.
- Gérard M, Clautiaux F, Sadykov R (2016) Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce. *Eur. J. Oper. Res.* 252(3):1019–1030.
- Goel A (2010) Truck driver scheduling in the European Union. *Transportation Sci.* 44(4):429–441.
- Goel A, Irnich S (2017) An exact method for vehicle routing and truck driver scheduling problems. *Transportation Sci.* 51(2):737–754.
- Goel A, Kok L (2012) Truck driver scheduling in the United States. *Transportation Sci.* 46(3):317–326.
- Gurobi Optimization (2020) *Gurobi Optimizer Reference Manual*, version 9.0, https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf.
- Henn S (2015) Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. *Flexible Service Manufacturing J.* 27(1):86–114.
- Henn S, Schmid V (2013) Metaheuristics for order batching and sequencing in manual order picking systems. *Comput. Indust. Engrg.* 66(2):338–351.

- Holder A (2005) Navy personnel planning and the optimal partition. *Oper. Res.* 53(1):77–89.
- Irnich S (2008) Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum* 30(1):113–148.
- Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, New York), 33–65.
- Kallehauge B, Larsen J, Madsen OB, Solomon MM (2005) Vehicle routing problem with time windows. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, New York), 67–98.
- Kolen AW, Lenstra JK, Papadimitriou CH, Spieksma FC (2007) Interval scheduling: A survey. *Naval Res. Logist.* 54(5):530–543.
- Krishnamoorthy M, Ernst AT, Baatar D (2012) Algorithms for large scale shift minimisation personnel task scheduling problems. *Eur. J. Oper. Res.* 219(1):34–48.
- Kroon LG, Salomon M, Van Wassenhove LN (1995) Exact and approximation algorithms for the operational fixed interval scheduling problem. *Eur. J. Oper. Res.* 82(1):190–205.
- Matusiak M, De Koster R, Saarinen J (2017) Utilizing individual picker skills to improve order batching in a warehouse. *Eur. J. Oper. Res.* 263(3):888–899.
- Matusiak M, De Koster R, Kroon L, Saarinen J (2014) A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *Eur. J. Oper. Res.* 236(3):968–977.
- Michel R (2016) 2016 warehouse/DC operations survey: Ready to confront complexity. *Supply Chain Management Rev.*, (November 8), https://www.scmr.com/article/2016_warehouse_dc_operations_survey_ready_to_confront_complexity.
- Pisinger D, Ropke S (2010) Large neighborhood search. Gendreau M, Potvin JY, eds. *Handbook of Metaheuristics* (Springer, Boston), 399–419.
- Quak HJ, De Koster R (2007) Exploring retailers' sensitivity to local sustainability policies. *J. Oper. Management* 25(6):1103–1122.
- Rasmussen MS, Justesen T, Dohn A, Larsen J (2012) The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. *Eur. J. Oper. Res.* 219(3):598–610.
- Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Sci.* 40(4):455–472.
- Scholz A, Schubert D, Wäscher G (2017) Order picking with multiple pickers and due dates—Simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. *Eur. J. Oper. Res.* 263(2):461–478.
- Smet P, Ernst AT, Berghe GV (2016) Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. *Comput. Oper. Res.* 76:60–72.
- Spliet R, Dabia S, Van Woensel T (2018) The time window assignment vehicle routing problem with time-dependent travel times. *Transportation Sci.* 52(2):261–276.
- Stenger A, Vigo D, Enz S, Schwind M (2013) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Sci.* 47(1):64–80.
- Sungur B, Özgüven C, Kariper Y (2017) Shift scheduling with break windows, ideal break periods, and ideal waiting times. *Flexible Services Manufacturing J.* 29(2):203–222.
- Thompson GM (1988) A comparison of techniques for scheduling non-homogeneous employees in a service environment subject to non-cyclical demand. Unpublished PhD thesis, Florida State University.
- Thompson GM (1995) Improved implicit optimal modeling of the labor shift scheduling problem. *Management Sci.* 41(4):595–607.
- Thompson GM, Pullman ME (2007) Scheduling workforce relief breaks in advance vs. in real-time. *Eur. J. Oper. Res.* 181(1):139–155.
- Tilk C, Goel A (2020) Bidirectional labeling for solving vehicle routing and truck driver scheduling problems. *Eur. J. Oper. Res.* 283(1):108–124.
- Tilk C, Irnich S (2017) Dynamic programming for the minimum tour duration problem. *Transportation Sci.* 51(2):549–565.
- Tompkins JA, White JA, Bozer YA, Tanchoco J (2010) *Facilities Planning* (John Wiley & Sons, Hoboken, NJ).
- Tsai C-Y, Liou JJ, Huang T-M (2008) Using a multiple-GA method to solve the batch picking problem: Considering travel distance and order due time. *Internat. J. Production Res.* 46(22):6533–6555.
- Van den Bergh J, Beliën J, de Bruecker P, Demeulemeester E, De Boeck L (2013) Personnel scheduling: A literature review. *Eur. J. Oper. Res.* 226(3):367–385.
- Van Gils T, Ramaekers K, Braekers K, Depaire B, Caris A (2018a) Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. *Internat. J. Production Econom.* 197:243–261.
- Van Gils T, Ramaekers K, Caris A, De Koster RB (2018b) Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *Eur. J. Oper. Res.* 267(1):1–15.
- Vanderbeck F (2000) On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.* 48(1):111–128.