To appear in the *International Journal of Production Research*
Vol. 00, No. 00, 00 Month 20XX, 1–22

Order Picking Problems under Weight, Fragility, and Category Constraints

Thomas Chabot[a∗], Rahma Lahyani[b,c], Leandro C. Coelho[a,d], Jacques Renaud[a]

[a]*CIRRELT, Faculté des sciences de l'administration, Université Laval, Canada*
[b]*College of Business, Al Faisal University, Kingdom of Saudi Arabia*
[c]*LOGIQ Laboratory, Institut Supérieur de Gestion Industrielle, Sfax, Tunisia*
[d]*Canada Research Chair in Integrated Logistics, Université Laval, Canada*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Warehouse order picking activities are among the ones that impact the most the bottom lines of warehouses. They are known to often account for more than half of the total warehousing costs. New practices and innovations generate new challenges for managers and open new research avenues. Many practical constraints arising in real-life have often been neglected in the scientific literature. We introduce, model, and solve a rich order picking problem under weight, fragility, and category constraints, motivated by our observation of a real-life application arising in the grocery retail industry. This difficult warehousing problem combines complex picking and routing decisions under the objective of minimizing the distance traveled. We first provide a full description of the warehouse design which enables us to algebraically compute the distances between all pairs of products. We then propose two distinct mathematical models to formulate the problem. We develop five heuristic methods, including extensions of the classical largest gap, mid point, S-shape, and combined heuristics. The fifth one is an implementation of the powerful adaptive large neighborhood search algorithm specifically designed for the problem at hand. We then implement a branch-and-cut algorithm and cutting planes to solve the two formulations. The performance of the proposed solution methods is assessed on a newly generated and realistic test bed containing up to 100 pickups and seven aisles. We compare the bounds provided by the two formulations. Our in-depth analysis shows which formulation tends to perform better. Extensive computational experiments confirm the efficiency of the ALNS matheuristic and derive some important insights for managing order picking in this kind of warehouses.

Keywords: Order picking, warehousing, formulations, exact algorithms, heuristics.

1. Introduction

Order picking represents an important part of warehouse and inventory management activities (Gu et al. 2010). It consists of retrieving goods from specific storage locations to fulfill customers orders. It is considered as the most labor-intensive and costly warehousing activity (de Koster et al. 2007; Tompkins et al. 2010), and it may account for up to 55% of all warehouse operating expenses (Chiang et al. 2011). Therefore, warehouse productivity is highly affected by order picking activities (de Koster et al. 2007). Research in this area has rapidly expanded over the past decades. These studies can be categorized into books (Hompel and Schmidt 2006; Ackerman 2013; Manzini 2012), survey papers (Wäscher 2004; de Koster et al. 2007; Henn et al. 2012), and theoretical papers providing mathematical formulations and exact or approximate solution methods (Bortolini et al. 2015; Lu et al. 2016).

Order picking strategies and algorithms have often been studied for classical warehouses (Petersen 1997; Petersen and Aase 2004). The role of picker personality in predicting picking performance with

∗Corresponding author. Email: thomas.chabot@cirrelt.ca different picking technologies has been studied by de Vries et al. (2015). Recently, some attention has been devoted to researches oriented towards more realistic picking contexts (Chackelson et al. 2013; Matusiak et al. 2014; Chabot et al. 2015). These cases are either motivated by the complex characteristics of real-life warehousing activities, legal regulations, or the introduction of on-line shopping requiring faster and more customized services.

This paper is motivated by the situation prevailing in the grocery retail industry. In this industry, each company owns one or several distribution centers (DCs) to store its products. Almost all DCs are designed with reserve storage and fast pick areas. In the fast pick area, each stock keeping unit (SKU) is stored in a specific and dedicated location. However, in the reserve storage area, SKUs are placed at random available locations, but close to the picking and sorting area. Order pickers only have access to the pick area, which is replenished continually by employees from the reserve area. Each order is picked in a sequential way and transported to the sorting area by hand-lift trucks. Each picker deals with a list of products, i.e., a pick-list, to be collected from the storage aisles.

In general, when orders are small compared to the picking vehicle capacity, order batching occurs. Order batching consists of combining orders to reduce the distance traveled by the picker (de Koster et al. 2007; Petersen 1997; Petersen and Schmenner 1999). Batched orders must later be sorted. In our case, each individual order may consist of several hundreds of different SKUs, often resulting in thousands of units picked on many pallets.

The sequences followed by the pickers to retrieve all the SKUs of a given order are called routes. A fixed routing plan may perform well for some pick-lists, but poorly for others as decisions are made sequentially. Thus, the primary and most common objective in manual order-picking systems is to find the set of picking routes that minimizes the total traveled distance (de Koster and Van der Poort 1998; de Koster et al. 1998; Petersen et al. 2004).

In this paper we develop order picking routes in a warehouse under practical restrictions observed in the grocery retail industry. Each product characteristic considered adds an extra layer of difficulty to the problem. These product-specific properties are described next.

We introduce the order picking problem under weight, fragility, and category constraints (OPP-WFCC). Regarding the weight, as soon as the total weight of all the products transported on the hand-lift truck exceeds a threshold value, *heavy items* can no longer be collected, and the picker is only allowed to pick *light* SKUs. Thus, another tour is required to pick (some of) the remaining heavy items. This requirement, referred to as *weight* constraint, has two motivations. It helps avoid work accidents and back problems caused by lifting heavy charges to relatively high positions, and it ensures the vertical and horizontal stability of the pallet. Besides the weight, other constraints arise in practice regarding the fragility of the items. A product can support a certain weight without being crushed. Therefore, fragile products must not be placed underneath heavy products, i.e., heavy products must be on the bottom of the pallet and light products on the top. We refer to this restriction as *fragility* constraint and the maximum weight a product can support is referred to hereinafter as *self-capacity*. Finally, we consider two types of commodities: food and non-food products. Non-food products encompass household items. Food products should not be carried under non-food on the pallet in order to avoid contamination. Thus, one has to pick the non-food products separately or before any food products. This constraint is referred to as *category* constraint.

Order picking problems dealing with the physical properties of the products have not been widely studied in the literature, but similar constraints appear in other contexts. Matusiak et al. (2014) refer to these constraints as precedence constraints since they impose that some products must be picked before some others due to weight restrictions, fragility, shape, size, stackability, and preferred unloading sequence. They propose an heuristic method to solve the joint order batching and picker routing problem without any specific assumption regarding the layout and without any pre-determined sequencing constraints. Junqueira et al. (2012) introduce the problem of loading rectangular boxes into containers while considering cargo stability and load bearing constraints. Their mathematical model ensures the vertical and horizontal cargo stability and limits the maximum number of boxes that can be loaded above each other. Dekker et al. (2004) solved a real-world application arising in a warehouse storing tools and garden equipment. The authors considered different assumptions arising in this particular real-life application such as the design of the warehouse with non-coinciding start and end points, dead-end aisles, and two floors. For the sake of efficiency improvement, the authors examined the storage and the pickup policies of these products while ensuring that heavy products are picked first to prevent damaging fragile products. They proposed storage heuristics and routing heuristics to consider the application restrictions.

Routing problems with precedence constraints, in which one request must be served and/or loaded before another, have been widely studied in the VRP literature and may arise in several real-life contexts. Some practical applications include the dial-a-ride problem (Cordeau and Laporte 2007), airline scheduling (Medard and Sawhney 2007), and bus routing (Park and Kim 2010). If one introduces capacity constraints to the VRP, depending on the precedence constraints, the resulting problem is a pickup and delivery (Zachariadis et al. 2016). For more details on the VRP with precedence constraints, see Lahyani et al. (2014). The multi-constrained OPP studied in this paper is similar but quite different from the VRP with precedence constraints. Indeed, considering the self-capacity and the fragility constraints adds an extra layer of difficulty to the problem since it has a significant impact on the storage and routing strategies.

The contributions of this paper are threefold. First, we introduce a rich variation of the order picking problem under weight, fragility, and category constraints inspired from the grocery retail industry. Part of the complexity of the problem is due to the new practical constraints arising from the separation of the products, their fragility, the stability, and weight limit of the pallet. The second contribution is the development of two distinct formulations to model the problem, which include a precise computation of the distance matrix between all pairs of items within the warehouse and the development of original valid inequalities. Our third contribution is to develop heuristic algorithms and different exact methods to support warehouse picking operations in choosing the most suitable routing sequences to satisfy orders. Specifically, we propose branch-and-cut as well as an adaptive large neighborhood search (ALNS), and an extension of four classical order picking algorithms to solve the OPP-WFCC. We then solve large sets of instances reproducing realistic configurations using our algorithms, which aim at minimizing the total distance traveled for picking all items. We show that the classical order picking heuristics in the literature do not perform well for constrained order picking problems.

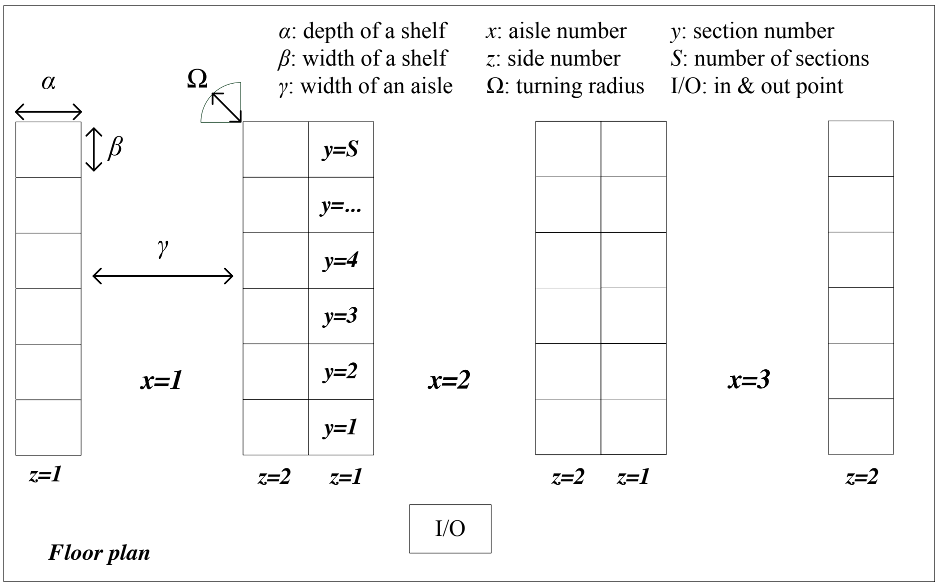
The remainder of the paper is organized as follows. In Section 2 we formally describe the problem and define its particularities. Section 3 presents two mathematical models along with a set of new valid inequalities. The details of the five heuristic algorithms and of the branch-and-cut procedures are provided in Section 4. The results of extensive computational experiments are presented in Section 5, and our conclusions follow in Section 6.

2. Problem description and distance modeling

In order to properly model the problem, which aims to minimize the total distance traveled by the pickers, one needs to know precisely the distances between all pairs of positions within the warehouse under study. We have modeled a warehouse constituted of several parallel aisles on a plane. There are complete racks in the middle of the warehouse and two half-racks on either side. These aisles are perpendicular to two corridors, one in the front and one in the rear. This configuration is commonly used and is generic enough for the purpose of this paper. Besides, the location of a product is fixed and each product takes just one location. In other words, each product exists in only one location within the warehouse. Finally, we suppose that the distance between the two sides of an aisle is large, such that the picker cannot pick items from both sides at the same time. We also suppose the one-dimensional stacking. The warehouse has one input/output (I/O) point which is considered as the departure and the arrival point for all the pickers. In addition, the warehouse is symmetric with respect to the I/O, with the same number of aisles on either side.

The distance between different locations is computed by solving a shortest path problem. Observe that in order to change aisles, the picker can move through the rear or the front corridor, and these two paths usually yield different distances. Let x_i represent the aisle number of SKU i, and y_i its section number, where $y_i \in \{1, \ldots, S\}$. The total number of sections S represents the number of in-depth locations of the aisle. A warehouse with one aisle and 20 sections per aisle contains 40 locations, by considering both sides. Note that aisles with vertical picking are modeled as well, but it is not a common layout in the grocery industry. The higher rack levels contain bulk stock, which is used to replenish the ground level, which contains the pick stock of the items. The side of product i within an aisle is z_i, with $z_i \in \{1, 2\}$. Using this notation, one can fully represent the location of product i as its coordinates (x_i, y_i, z_i) (Chabot et al. 2015). The width of an aisle is γ units of distance, the depth of a location is α units of distance, and its width is β. Since there are S sections in an aisle, the total length of the aisle is βS. The minimal distance between two aisles is given by 2α. In order to consider the complex characteristics of the real-world application under study, we consider a turning radius, denoted Ω. These features can be visualized in Figure 1.

Figure 1. Overview of the warehouse layout in the grocery retail industry



We define the distance matrix $\mathcal{D} = d_{ij}$ where $i < j$ and $i \in \{0, \ldots, m-1\}$, $j \in \{1, \ldots, m\}$, over the I/O point and all m product locations. Several scenarios must be considered when defining $\mathcal{D}$ as follows.

(i) Assume that i and j refer to two distinct products, then two cases may arise. The first one appears when both products are in the same aisle. The distance d_{ij} between i and j is then:

$$\underset{i<j}{d_{ij}} = |y_i - y_j|\,\beta + |z_i - z_j|\,\gamma \quad \text{if } x_i = x_j. \tag{2.1}$$

The first term computes the distance between the two sections, and the second term accounts for the aisle width γ if the products are on different sides of the aisle.

The second case arises when the two products are in different aisles. With the aim of easing the notation, we separate the equations into two segments: the length-wise distance, denoted υ, and the width-wise described next. The length proportion is expressed by $\upsilon = \min(\beta(2S - y_i - y_j), \beta(y_i + y_j)) + 2\Omega$. The total distance can then be computed as:

$$d_{\substack{ij\\i<j}} = \begin{cases} |x_i - x_j|\,(2\alpha + \gamma) + \upsilon & \text{if } z_i = z_j & (2.2a) \\ (x_j - x_i)(2\alpha + \gamma) + \gamma + \upsilon & \text{if } z_i = 1, z_j = 2 & (2.2b) \\ (x_j - x_i)(2\alpha) + (x_j - (x_i + 1)) + \upsilon & \text{if } z_i = 2, z_j = 1. & (2.2c) \end{cases}$$

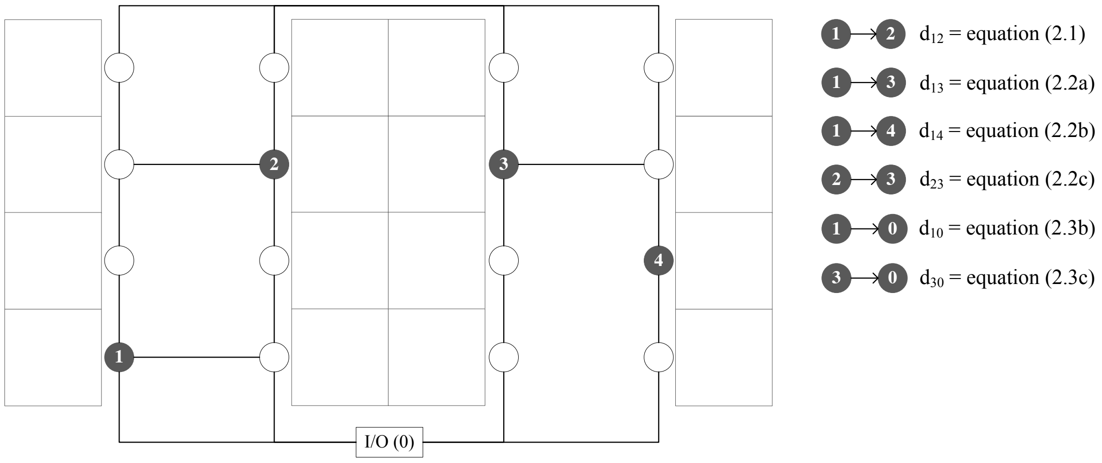
The first case appears when products are on same side, but in different aisles. In the second case, the items are in two different sides, such that the picker has to cross the aisle width at the departure and at the arrival aisles. In the third case, products i and j are placed such that the start and arrival aisles are not crossed.

(ii) Now, assume that i refers to a product and j refers to the I/O point. The distances between a product and the I/O point are symmetric and must be computed by taking into account three cases. The first and easiest one is when section x_i of item i coincides with section x_0 of the I/O point. The second appears if $x_i < x_0$, and the third if $x_i > x_0$. These distances are then computed as:

$$d_{i0} = \begin{cases} y_i\beta + \frac{1}{2}\gamma & \text{if } x_i = x_0 & (2.3a) \\ (x_0 - x_i)2\alpha + (x_0 - x_i - z_i + 1)\gamma + \frac{1}{2}\gamma + \Omega + y_i\beta & \text{if } x_i < x_0 & (2.3b) \\ (x_i - x_0)2\alpha + (x_i - x_0 + z_i - 2)\gamma + \frac{1}{2}\gamma + \Omega + y_i\beta & \text{if } x_i > x_0. & (2.3c) \end{cases}$$

In the first case, in which $x_i = x_0$, the distance is the total of the number of sections plus half of the width of an aisle. The second case appears when the product is located on the left of the I/O point, and the third case if the product is on the right. These two cases are symmetrical and are composed of the width of the cells to be traversed, the width of the aisles to be crossed, the turning radius distance, and the length of the sections needed to traverse to reach the product. The different scenarios defining the distance matrix $\mathcal{D}$ are illustrated in Figure 2.

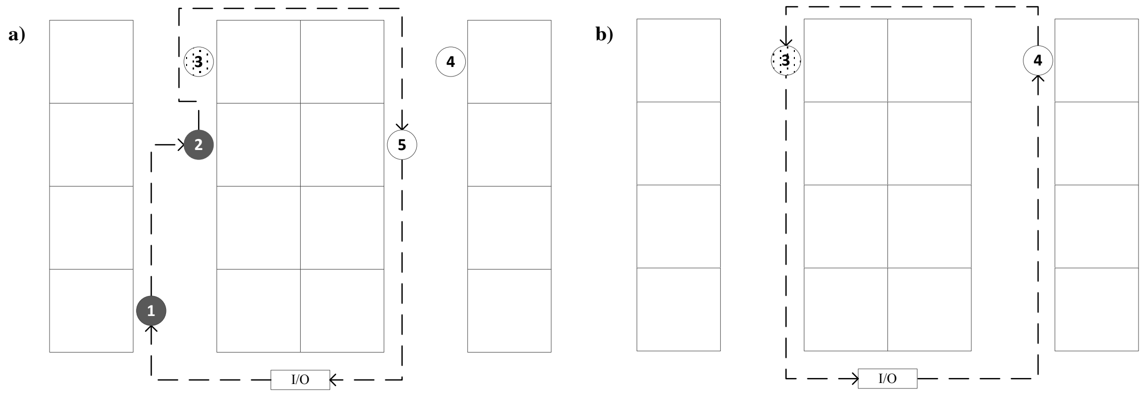
Figure 2. Illustration of the distance matrix and the corresponding equations


1
2
3
4
I/O (0)
1 → 2 d12 = equation (2.1)
1 → 3 d13 = equation (2.2a)
1 → 4 d14 = equation (2.2b)
2 → 3 d23 = equation (2.2c)
1 → 0 d10 = equation (2.3b)
3 → 0 d30 = equation (2.3c)

The OPP-WFCC is formally defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the vertex set and $\mathcal{A}$ is the arc set. The set $\mathcal{V} = \{0, \ldots, m\}$ contains the location of the I/O point and the m products locations constituting the set $\mathcal{V}' = \{1, \ldots, m\}$. The set $\mathcal{A} = \{(i, j) : i \in \mathcal{V}, j \in \mathcal{V}', i \neq j\}$ is the arc set. Each product $i \in \mathcal{V}'$ has a weight q_i. In the OPP-WFCC products have three important characteristics. First, a product is said to be light if its weight is under B units, otherwise it is considered as a heavy item. Second, a product can also be fragile or non-fragile. A weight limit w_i, i.e., a self-capacity, is associated with each fragile item i. If a given item i is considered as non-fragile, then w_i is assumed to be Q which corresponds to the total weight that can be loaded on a pallet. Finally, products are also categorized as food or non-food items. Non-food items cannot be loaded on top of food items. With this requirement, we observe that arcs (i, j) with i being a food item and j being a non-food item can be removed from $\mathcal{A}$. Finally, when the total weight of all picked items on the pallet reaches a limit L, no more heavy products can be picked in the same tour. The objective of OPP-WFCC is to find the set of tours minimizing the total distance while respecting the weight, fragility, and category constraints.

To better describe the particularities of this problem, in Figure 3 we sketch a solution for the OPP-WFCC in which precedence constraints may interfere. The output of the example solution includes two routes where each route begins and ends at the I/O node. The picking points are visited in a specific order respecting the weight, fragility, capacity, and category constraints. Items 1 and 2 are non-food products while items 3, 4, and 5 are food products. Besides, products 2 and 3 are considered fragile. The first route, presented in Figure 3a, starts by picking product 1 then product 2. Since these are non-food products, the picker can load food-products on the top of the stack while respecting the category constraints. However, the total weight of the pallet reaches the threshold $L = 100$ so the picker can no longer load any heavy products. Since the self-capacity of product 2 equals to 20, product 3 ($q_3 = 50$) or product 4 ($q_4 = 45$) cannot be loaded on top of it. Finally route 1 picks product 5 before going back to the I/O node. The products left in the warehouse are 3 and 4. Since product 4 is heavier than the self-capacity of product 3, route 2 first picks product 4, then product 3, and then goes back to the I/O (Figure 3b). The set of constraints requires the picker to do two tours, both going around the warehouse and the two aisles. Without constraints, the picker could only make one tour to pick everything. In such a case, the distance to pick all products is almost the double.

Figure 3. Example of a solution for the OPP-WFCC



$Q = 150$
$L = 100$
$B = 40$

● Non-food
○ Food
Fragile

Weight	Self-cap.	Food	Non-food	Fragile
$q_1 = 100$	$w_1 = 150$		✓	
$q_2 = 30$	$w_2 = 20$		✓	✓
$q_3 = 50$	$w_3 = 40$	✓		✓
$q_4 = 45$	$w_4 = 150$	✓		
$q_5 = 20$	$w_5 = 150$	✓		

3. Mathematical formulations

We now provide two different formulations for the OPP-WFCC. In Section 3.1 we present a capacity-indexed formulation which makes explicit the remaining capacity of the pallet traversing each arc.

In Section 3.2 we present a two-index vehicle flow formulation.

Recall that d_{ij} denotes the distance between two nodes i and j defining arc (i,j) computed as described in Section 2. Let $\mathcal{V}_h^{'}$ be the set of nodes associated with heavy products.

3.1 *Capacity indexed formulation*

In this section, we propose a new formulation to model the OPP-WFCC, referred to as capacity-indexed formulation. This type of formulation has only appeared a few times for basic and rich variants of VRPs (Picard and Queyranne 1978; Pessoa et al. 2009; Poggi de Aragão and Uchoa 2014; Lahyani et al. 2015). This formulation is compact enough to enumerate all variables and constraints for small and medium instances of the problem. We later incorporate new procedures to further reduce the number of variables. Let binary variables x_{ij}^q indicate that arc (i,j) is used with a remaining capacity of q units. Let $\mathcal{C} \subset Q$ be the subset of possible values of q from 1 to Q. The formulation is defined by:

$$\text{(F1) minimize} \sum_{(i,j)\in\mathcal{A}} d_{ij} \sum_{q\in\mathcal{C}} x_{ij}^q \tag{3.1}$$

subject to

$$\sum_{j\in\mathcal{V}} \sum_{q\in\mathcal{C}} x_{ij}^q = 1 \quad i \in \mathcal{V} \tag{3.2}$$

$$\sum_{i\in\mathcal{V}} \sum_{q\in\mathcal{C}} x_{ij}^q = 1 \quad j \in \mathcal{V} \tag{3.3}$$

$$\sum_{i\in\mathcal{V}} x_{ij}^q - \sum_{k\in\mathcal{V}} x_{jk}^{q-q_j} = 0 \quad j \in \mathcal{V}^{'}, q > q_i \in \mathcal{C} \tag{3.4}$$

$$\sum_{i\in\mathcal{V}'} x_{i0}^q = 0 \quad q > 1 \in \mathcal{C} \tag{3.5}$$

$$\sum_{j\in\mathcal{V}'} x_{0j}^0 = 0 \tag{3.6}$$

$$x_{ij}^q \in \{0,1\} \quad i,j \in \mathcal{V}, q \in \mathcal{C}. \tag{3.7}$$

The traveled distance is minimized in (3.1). Equations (3.2) and (3.3) are the in and out degree constraints. They ensure that each node is visited exactly once. Equations (3.4) guarantee that if the picker visits a node j with a remaining capacity q, then he must leave this node after picking load q_j with a remaining capacity of $q - q_j$. These constraints ensure the connectivity of the solution and the capacity requirements. Infeasible and unnecessary variables are removed with equalities (3.5) and (3.6). Equations (3.5) forbid arcs to return to the I/O point with a remaining capacity different from zero. Similarly, equations (3.6) state that all arcs visiting a node i must carry a load with available space for item i. Constraints 3.7 define the domain of the variables.

The main advantage of formulation F1 is that it enables to preprocess the model to decrease its size and to remove infeasible variables by adding appropriate constraints in the preprocessing of the variables. Indeed, because of the physical characteristics of the products, precedence constraints should be considered when defining picking route to avoid putting non-food products above food products on the pallet. One can identify arcs going from non-food to food products beforehand. Moreover, one can ensure that the safety requirements for fragile products are always respected by removing infeasible variables. Because the index q carries the information about the remaining weight capacity, all arcs with q greater than the self-capacity w_j and the weight q_j of the destination product j, i.e., $q > w_j + q_j$, cannot exist. Thus, all the arcs generated respect the weights that fragile products can support. We can also eliminate variables x_{ij}^q traversing arcs (i, j) with infeasible remaining capacity, i.e., with a capacity $q < Q - q_i$. In what concerns the *weight* constraint, the picker cannot load any heavy product if the total weight of the pallet reaches the threshold L. We model this restriction with inequalities (3.8):

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L, l \in \mathcal{S}, h \in \mathcal{V}_h^{'} \backslash \mathcal{S}. \tag{3.8}$$

Recall that a heavy product is a product whose weight exceeds the limit B. Consider a subset of nodes $\mathcal{S}$ associated with products whose total weight is greater than or equal to L. We check if the potential item to be picked in the same route is a heavy product and we impose that the picker can only pick light items. The arc (l, h) links the last node l of the subset $\mathcal{S}$ and the node corresponding to the heavy product h. Constraints (3.8) can be improved by considering all the potential heavy products that cannot be picked on the same route. We lift the second term of the left-hand side over the nodes related to heavy products, i.e., the nodes in the subset $\mathcal{V}_h^{'}$. Constraints (3.8) can thus be replaced by:

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{h \in \mathcal{V}_h^{'} \backslash \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L, l \in \mathcal{S}. \tag{3.9}$$

Constraints (3.9) can be further generalized by removing the restriction to leave the subset $\mathcal{S}$ from the last node visited in this subset. We then replace constraints (3.9) by:

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{l \in \mathcal{S}} \sum_{h \in \mathcal{V}_h^{'} \backslash \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L. \tag{3.10}$$

Since the number of constraints (3.10) is exponential (in the order of $O(2^m)$), they cannot be generated a priori. We propose a branch-and-cut algorithm to add them dynamically. Details about this algorithm are provided in Section 4.2.

It is possible to strengthen the formulation by adding some valid inequalities. Taking into account the physical characteristics of the products and their structural capacity, we can identify general situations in which a solution is not feasible. We already use several of these characteristics in the variables creation, and we could derive the following new cuts. In constraints (3.11), we consider an arc between product i and j, and its self-capacity w_i. If a route follows the path from i to j, and from j to k, then one can impose the following constraints:

$$x_{ij}^{q} + \sum_{\substack{k \in \mathcal{V}' \\ w_i < q_j + q_k}} x_{jk}^{q-q_j} \leq 1 \quad i, j \in \mathcal{V}', q \geq q_j \in \mathcal{C}. \tag{3.11}$$

We can also use the self-capacity to derive a new family of valid inequalities. Let $\mathcal{S}_1$ represent a set of fragile products and its complement denoted $\mathcal{S}_2$, and all arcs going from $i \in \mathcal{S}_1$ to any product j. We know that all arcs going out of j to a product k with the sum of weights $q_k + q_j$ is larger than the maximum self-capacity of the products in $\mathcal{S}_1$ will not be permitted. So all other self-capacity of any products in $\mathcal{S}_1$ are also violated. Thus, valid inequalities (3.12) forbid this situation:

$$\sum_{i \in \mathcal{S}_1} x_{ij}^{q} + \sum_{\substack{k \in \mathcal{S}_2 \\ \max_{i \in \mathcal{S}_1}\{w_i\} < q_k + q_j}} x_{jk}^{q-q_j} \leq 1 \quad j \in \mathcal{V}', \mathcal{S}_1 \subseteq \mathcal{V}' : w_i < Q, \mathcal{S}_2 \subseteq \mathcal{V}', q \geq q_j \in \mathcal{C}. \tag{3.12}$$

3.2 *Two-index flow formulation*

Formulation F1 has the drawback that the number of variables is dependent on the number of q values that can be obtained by different picking routes. We now provide a two-index flow formulation to determine the best picker routes, based on the model described in (Laporte 1986; Toth and Vigo 2014) for the asymmetric VRP. We define new binary variables x_{ij} equal to one if arc (i, j) is used, and an integer variable K indicating the number of picking tours required to satisfy all the orders. This model can be stated as follows:

$$\text{(F2) minimize} \sum_{(i.j) \in \mathcal{A}} d_{ij} x_{ij} \tag{3.13}$$

subject to

$$\sum_{j \in \mathcal{V}'} x_{0j} = K \tag{3.14}$$

$$\sum_{i \in \mathcal{V}'} x_{i0} = K \tag{3.15}$$

$$\sum_{i \in \mathcal{V}} x_{ij} = 1 \quad j \in \mathcal{V}' \tag{3.16}$$

$$\sum_{j \in \mathcal{V}} x_{ij} = 1 \quad i \in \mathcal{V}' \tag{3.17}$$

$$\sum_{i,j \in \mathcal{S}} x_{ij} \leq |\mathcal{S}| - r(s) \quad \mathcal{S} \subseteq \mathcal{V}', |\mathcal{S}| \geq 2 \tag{3.18}$$

$$K \in \mathbb{N}_+ \tag{3.19}$$

$$x_{ij} \in \{0,1\} \quad i,j \in \mathcal{V} \tag{3.20}$$

The objective function (3.13) minimizes the total traveled distance. Constraints (3.14) and (3.15) impose the degree requirements for the I/O point. Equations (3.16) and (3.17) are in-degree and out-degree constraints. They state that each product i must be picked exactly once. Constraints (3.18) correspond to generalized subtour elimination constraints. They simultaneously forbid subtours and ensure the capacity requirements with $r(s) = \left\lceil \frac{\sum_{j \in \mathcal{S}} q_j}{Q} \right\rceil$. Constraints (3.19) and (3.20) impose integrality and binary conditions on the variables.

Similarly to the capacity-indexed formulation, one can add beforehand the *category* constraints when creating the variables. We also remove irrelevant arcs going from non-food to food products. However, unlike formulation F1, formulation F2 cannot handle the *fragility* constraints in a pre-processing phase by eliminating infeasible variables. Thus, we propose a branch-and-cut routine that dynamically adds the fragility constraints defined by (3.21). For a subset of nodes $\mathcal{S}$, we check whether the products related to the potential nodes that may follow node i respect the maximum weight supported by i, w_i. If this condition is not met, we then impose the fragility constraints:

$$\sum_{j \in \mathcal{S}} x_{ij} + \sum_{j,k \in \mathcal{S}} x_{jk} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s > w_i, i \in \mathcal{V}' \backslash \mathcal{S}. \tag{3.21}$$

The *weight* constraints (3.10) proposed for formulation F1 can be adapted for F2 by removing the capacity index, which results in inequalities (3.22):

$$\sum_{i,j \in \mathcal{S}} x_{ij} + \sum_{l \in \mathcal{S}} \sum_{h \in \mathcal{V}'_h \backslash \mathcal{S}} x_{lh} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L. \tag{3.22}$$

We have also adapted the two sets of valid inequalities (3.11) and (3.12) from model F1 to model F2. The new valid inequalities (3.23) and (3.24) may be written as follows:

$$x_{ij} + \sum_{\substack{k \in \mathcal{V}' \\ w_i < q_j + q_k}} x_{jk} \leq 1 \quad i,j \in \mathcal{V}' \tag{3.23}$$

$$\sum_{i \in \mathcal{S}_1} x_{ij} + \sum_{\substack{k \in \mathcal{S}_2 \\ \max_{i \in \mathcal{S}_1}\{w_i\} < q_k + q_j}} x_{jk} \leq 1 \quad j \in \mathcal{V}' \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{V}'. \tag{3.24}$$

4. Solution algorithms

We now describe the proposed heuristics and exact algorithms to solve the OPP-WFCC introduced in this paper. We describe four classical heuristic algorithms and the ALNS metaheuristic in Section 4.1. Then, we present the branch-and-cut algorithm and the cutting planes in Section 4.2.

4.1 *Heuristic algorithms*

In this section we describe five heuristic algorithms to solve the OPP-WFCC. The first four are extensions of classical and well-known heuristics that we have adapted to solve the OPP-WFCC, and the last one is a metaheuristic designed specifically for our problem. In Section 4.1.1 we describe how we have adapted the S-shape, mid point, and largest gap heuristics of Hall (1993) along with the combined heuristic of Roodbergen and de Koster (2001) for our problem. These solution methods are modified to handle the precedence constraints considered in the OPP-WFCC. Section 4.1.2 presents the main procedures of the ALNS metaheuristic.

4.1.1 Classical heuristics

Under the S-shape heuristic, the picker will completely traverse any aisle containing at least one item to be picked. Thus, the warehouse is completely traversed, by leaving an aisle and entering the adjacent one, picking products as the picker advances.

In the mid point heuristic we divide the warehouse into two halves. There are picks on the front side, reached by the front aisle, and picks on the back side reached from the back side. The picker only crosses through the first and last aisles of the picking tour. Intermediate aisles are never crossed completely. According to Hall (1993), this method performs better than the S-shape when the number of picks per aisle is small.

In the largest gap heuristic, the picker follows the same idea of the S-shape by visiting all adjacent aisles, but instead of traversing the aisle completely, he enters the aisle up to the item that will leave the largest gap for the next item in that aisle. This heuristic tries to maximize the parts of the aisles that are *not* traversed. When the largest gap is identified, the picker returns and leaves the aisle from the same side (back or front) used to enter it. We proceed by making the first pick as in the original heuristic and by traversing the warehouse following the original rules.

In the combined heuristic, picking routes visit every aisle that contains at least one item. We have adapted the dynamic programming algorithm of Roodbergen and de Koster (2001) to determine whether it is better to traverse the current aisle in full or to go back, then entering the next aisle from the front or from the rear side. In general, this method is best suited for pickings that follow a TSP-like tour.

The following modifications are required to these four heuristics in order to handle weight, fragility, and category constraints. We start by picking the first available product in the left most aisle as in the original version of the heuristics. Any subsequent product is picked if it respects all the constraints of the problem, and skipped otherwise. We continue following the heuristic rules traversing the warehousing, and skipping any infeasible pick until the pallet is full or the last item is reached. At this point, the picker returns to the I/O point and starts a new route going back to the first skipped item, which will be picked and the same procedure is repeated as long as there are products to be picked.

4.1.2 Adaptive large neighborhood search heuristic

We present an implementation of an Adaptive large neighborhood search heuristic (ALNS) metaheuristic, widely based on Ropke and Pisinger (2006). The ALNS is composed of a set of simple destruction and reconstruction heuristics in order to find better solutions at each iteration using an adaptive layer to keep track of the performance of the invoked heuristics. An initial solution can be considered to speed up the search and the convergence of the algorithm. We have implemented a fast sequential insertion heuristic, which performs a greedy search for the best insertion for one product at a time.

The ALNS selects one of many destroy and repair operators at each iteration. We have implemented three destroy and two repair operators. Destroy operators include the Shaw removal (Shaw 1997), the worst removal, and a random removal. Repair operators include a greedy parallel insertion and a k-regret heuristic (Potvin and Rousseau 1993). Each operator is selected with a probability that depends on its past performance, and a simulated annealing acceptance criterion is used. Each procedure of the ALNS is adapted to deal with the weight, fragility, and category constraints.

The ALNS is run for 50,000 iterations of destroy-repair operations. After every 100 iterations the weight of each operator is updated according to its past performance. Initially, all the operators have the same weight. A sketch of our ALNS implementation is provided in Algorithm 1 (for further details see Ropke and Pisinger (2006)).

Algorithm 1 Adaptive large neighborhood search metaheuristic

```
1: Create an initial solution $s$ using sequential insertion operators : $s' = s$
2: $D$ : set of removal operators, $I$ : set of repair operators
3: Initiate probability $\rho^d$ for each destroy operator $d \in D$ and probability $\rho^i$ for each repair operator $i \in I$
4: while stopping criterion is not met do
5:    Select remove $(d \in D)$ and insert $(i \in I)$ operators using $\rho^d$ and $\rho^i$
6:    Apply operators and create $s^t$
7:    if $s^t$, is accepted by the simulated annealing criterion then
8:       $s = s^t$
9:    end if
10:   if $cost(s) < cost(s')$ then
11:      $s' = s$
12:   end if
13:   Update $\rho^d$ and $\rho^i$
14: end while
15: return $s'$
```

4.2 *Exact algorithms*

In this section we present the various algorithms used to solve exactly the mathematical models from Section 3. We present in Section 4.2.1 a subset sum algorithm that enables us to drastically reduce the number of required variables for model F1. This is followed in Section 4.2.2 by an overview of the branch-and-cut algorithm we have used as well as the detailed description of the procedures used to dynamically identify and generate violated valid inequalities.

4.2.1 Subset sum algorithm to reduce the number of variables of model F1

The variables of formulation F1 presented in Section 3.1 can only be fully enumerated for small and medium size instances. However, it is easy to observe that some variables are never used in the model, e.g., the ones for which some values of q cannot be obtained by any combination of the weights of the products. These variables can be generated and fed into the solver, which will set them to zero in any feasible solution. If one can identify these variables beforehand, it is possible to set them to zero and remove them from the model at a preprocessing phase. Thus, one can (substantially) decrease the size of the model and the memory footprint by preprocessing the model and the instance a priori, identifying the subset of variables that should not be generated.

We use a subset sum algorithm to identify all possible values of q from 1 to Q that can be achieved by any combination of weights q_i. The ones that are found not to be feasible are not generated.

From a theoretical point of view, its performance is directly related to the distribution of the weights of the items in the instance. For example, an instance for which all products have a weight of five units will have five time less variables than the model with all the values of q.

4.2.2 Branch-and-cut algorithms

Models F1 and F2 can be fed straightforwardly into a general purpose solver and solutions are obtained by branch-and-bound if the number of constraints (3.10), (3.18), (3.21) and (3.22) is not excessive. However, for instances of realistic size, the number of these constraints is too large to allow a full enumeration and they must be dynamically generated throughout the search process. Indeed, polynomial constraints may be added a priori to the model while other constraints cannot be generated a priori since their number is exponential.

The exact algorithm we present is a branch-and-cut scheme in which inequalities (3.10), (3.18), (3.21), and (3.22) are generated and added into the program whenever they are found to be violated. It works as follows. At a generic node of the search tree, a linear program containing the model with a subset of the subtour elimination constraints and relaxed integrality constraints is solved, a search for violated inequalities is performed, and some of these are added to the current program which is then reoptimized. This process is reiterated until a feasible or dominated solution is reached, or until there are no more cuts to be added. At this point, branching on a fractional variable occurs. We provide a sketch of the branch-and-cut scheme in Algorithm 2. Note that for formulation F1, we apply Algorithm 2 but we skip the process to identify violated subtour elimination constraints (lines $8-12$) as connectivity requirements are ensured by constraints (3.4).

Algorithm 2 Branch-and-cut algorithm

```
1: Subproblem solution: Solve the LP relaxation of the current node
2: Termination check:
3: if there are no more nodes to evaluate then
4:     Stop
5: else
6:     Select one node from the branch-and-cut tree
7: end if
8: while the solution of the current LP relaxation contains subtours do
9:     Identify connected components as in Padberg and Rinaldi (1991)
10:    Add violated subtour elimination constraints
11:    Subproblem solution. Solve the LP relaxation of the current node
12: end while
13: if the solution contains no disconnected components then
14:    Apply Algorithms 3 and 4, and add violated cuts
15: end if
16: if the solution of the current LP relaxation is integer then
17:    Go to the termination check
18: else
19:    Branching: branch on one of the fractional variables
20:    Go to the termination check
21: end if
```

In this branch-and-cut algorithm, weight, and fragility inequalities are used as cutting planes to strengthen the linear programming relaxation at each node of the branch-and-bound tree. Constraints (3.10), (3.21), and (3.22) cannot be generated a priori since their number is exponential. These are initially relaxed and dynamically generated as cuts as they are found to be violated.

When model F1 without *weight* constraints (3.10) (similarly with (3.22) for F2) is solved, two situations can occur. The first one consists of finding an integer solution. Then one can easily verify if the picking tour exceeds the weight limit L or the self-capacity w_i by calculating the cumulative weight of a set $\mathcal{S}$. In the second case, when the solution is fractional, one can identify connected components by means of the maximum flow algorithm as in Padberg and Rinaldi (1991). This procedure, sketched in Algorithm 3, consists of constructing an auxiliary graph as follows. First, a node i is selected. Then the node j associated with the maximum flow value leaving i is identified and added to the set $\mathcal{S}$. We check whether the sum of weights of the products included in $\mathcal{S}$, referred to by $q_{\mathcal{S}}$, respects the threshold L. If it exceeds this limit, we add cuts to forbid such solution. This is achieved by adding the appropriate constraints (3.10) for the capacity indexed formulation, and (3.22) for the two-index flow formulation associated with the nodes in $\mathcal{S}$.

Similarly, in Algorithm 4, we describe the procedure used to dynamically generate constraints (3.21). We identify a subset $\mathcal{S}$ such that it respects the *fragility* constraints. Otherwise, we add the violated constraints associated with the nodes in $\mathcal{S}$.

Algorithm 3 *Weight* constraints algorithm

```
1: for $i = 1$ to $m$ do
2:   $\mathcal{S} = \{i\}$
3:   $j^* = argmax_{k \in \mathcal{V}' \setminus \mathcal{S}} \{x_{ik}\}$
4:   $\mathcal{S} = \mathcal{S} \cup \{j^*\}$
5:   if $q_{\mathcal{S}} > L$ and $q_{\mathcal{S}} \leq Q$ then
6:     for $l \in \mathcal{S}$ do
7:       if the solution violates (3.10) (or (3.22)) then
8:         Add weight constraints (3.10) (or (3.22))
9:       end if
10:    end for
11:  end if
12:  if $q_{\mathcal{S}} > Q$ then
13:    Continue to next $i$ from step 1
14:  else
15:    Go to Step 3
16:  end if
17: end for
```

5. Computational experiments

In this section, we provide details on the implementation, benchmark instances, and describe the results of extensive computational experiments. Implementation and hardware information is provided in Section 5.1. The description of the benchmark instances is presented in Section 5.2, followed by the results of our computational experiments. Heuristic results are presented in Section 5.3 and the results of experiments carried out in order to assess the performance of the proposed exact methods are presented in Section 5.4.

5.1 *Implementation details*

All the formulations described in Section 3 and the algorithms described in Section 4 were implemented in C++. The branch-and-cut algorithm uses the CVRPSEP library (Lysgaard 2003) for the sub-tours elimination constraints, the newly proposed cutting methods, and IBM CPLEX Concert Technology 12.6 as the branch-and-bound solver. All computations were executed on machines equipped with two Intel Westmere EP X5650 six-core processors running at 2.667 GHz, and with up to 8 GB of RAM installed per node running the Scientific Linux 6.3. All algorithms were provided a time limit of 7200 seconds.

Algorithm 4 *Fragility* constraints algorithm

```
1: for $i = 1$ to $m$ do
2:   if $i$ is a fragile product then
3:     $\mathcal{S} = \{i\}$, best $= i$
4:     while stop criterion is not met do
5:       $j^* = argmax_{k \in \mathcal{V}' \setminus \mathcal{S}} \{x_{best,k})\}$
6:       if $x_{best,k} < 0.5$ then
7:         Stop
8:       end if
9:       $\mathcal{S} = \mathcal{S} \cup \{j^*\}$, Best$=j$
10:      if $q_{\mathcal{S}} \leq Q$ and $q_{\mathcal{S}} > w_i$ then
11:        if the solution violates (3.21) then
12:          Add fragility constraints (3.21)
13:        end if
14:      end if
15:      if $q_{\mathcal{S}} > Q$ then
16:        Stop
17:      end if
18:    end while
19:  end if
20: end for
```

5.2 *Data sets generation*

Since no data sets are available for the OPP-WFCC, we have created three groups of instances to represent different configurations and combinations of the new features introduced in this paper. To reflect common practice, non-food items are grouped in the same aisle(s). In the first group of instances, called G1, only one side of the first aisle is dedicated to non-food items, whereas all fragile and non-fragile products are randomly placed throughout the warehouse. In the second group of instances (G2), we split the picking area in two symmetric zones. This allows non-food items to be placed in the lateral extremities of the warehouse. All remaining items are placed randomly elsewhere. In the third group (G3), non-food items are placed in the extremities like in G2, but solid products (SP), i.e., non-fragile, with large self-capacity are grouped together in the sections close to the I/O point. These three types of layouts are illustrated in Figure 4.

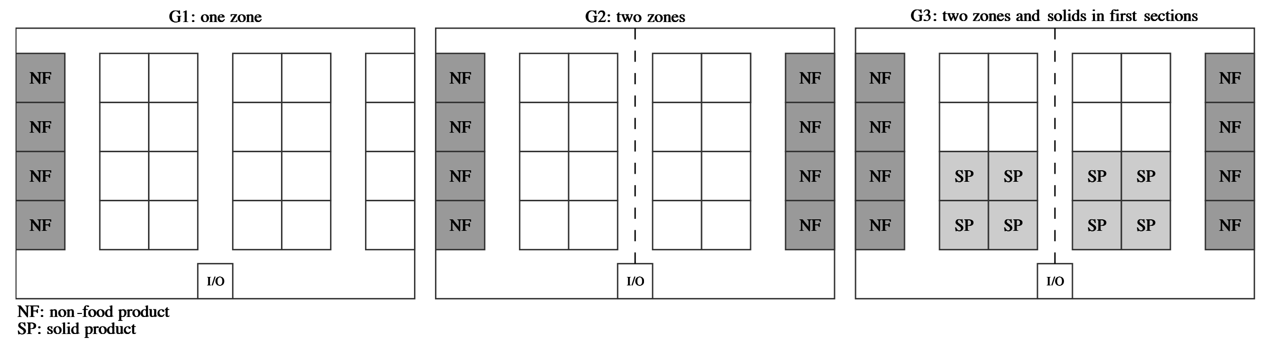


Figure 4. Schematic representation of the three groups of instances

In each group of instances, we have kept the same proportion of food and non-food products, and ranges of weight and self-capacity. The generator has a 50% probability to create a fragile product. Each fragile product has a 2/3 probability to be a food product and 1/3 to be non-food. A non-fragile product, food or otherwise, has a weight between five and 50 kg, and infinite self-capacity. A fragile product has a weight between one and 10 kg, and a self-capacity between five and 50 kg.

The general characteristics of this large test bed are summarized in Table 1. An instance is characterized by the number of picks, the number of aisles, and the capacity of the picking vehicles. The number of SKU to be picked is between 20 and 100, by steps of 10. The number of aisles is equal to 3, 4, 5, 6 or 7, with the I/O point located in the middle of the warehouse. The capacity of the pallet may take the value 150 or 250. In total, there are 270 different parameter combinations represented in Table 1. For each combination, we have generated randomly three instances, for test bed of 810 distinct instances. Recall that each aisle side contains 20 storage locations and the number of SKU per warehouse depends on the number of aisles. Regarding the parameters values, we have used the following physical distance parameters: $\alpha = 10$, $\beta = 5$, $\gamma = 15$, and $\Omega = 5$.

Table 1. General characteristics of the generated test bed

Parameter	Variations	Values
group	3	G1, G2, G3
# of picks	9	20, 30, 40, 50, 60, 70, 80, 90, 100
# of aisles	5	3, 4, 5, 6, 7
capacity of the pallet	2	150, 250

5.3 *Heuristic results*

This section presents the results of extensive computational experiments of the five heuristic algorithms presented in Section 4.1 for the test bed containing 810 instances. Table 2 presents the solutions of the heuristic algorithms for all the three groups of instances. Specifically, for each heuristic, group, and pallet capacity, and for each number of pickups we provide the total distance which is the average over 15 instances, i.e., three instances for each scenario. We also provide the average distance over each group.

The best average results of classical heuristics are underlined for each capacity and each group. In terms of computational times required by the different heuristic algorithms, the CPU time needed is less than one second for instances with 100 pickups and seven aisles. The classical heuristics results presented in Table 2 highlight that their performance slightly differ over the three instances groups. The mid-point heuristic is most likely less effective than the other heuristics. For the first group, the combined heuristic outperforms the other three. For G2, the S-shape heuristic is the best one. Finally, for G3, the largest gap heuristic yields the best results. These results hold for both capacity scenarios. More generally, the four classical heuristics yielded very similar solutions, whereas these may be significantly enhanced by the ALNS metaheuristic.

A richer and more intricate metaheuristic, such as the ALNS, significantly improves the solution, as shown in Table 2. The average solution is reduced by almost 50% for all groups. This may be explained by the fact that the ALNS metaheuristic provides more compact solutions with fewer picking tours and much shorter distances, thanks to its intensification and diversification procedures. Regarding the running time, the ALNS improves the solution quality with no significant time increase. The average running time is around 60 seconds for the largest instances with 100 pickups and 7 aisles. As expected, increasing the capacity of the pallet (as done in the second scenario) reduces the distance traversed, and this for all heuristics. Moreover, being the most advanced one, the ALNS better exploits this extra capacity and reduces distances significantly.

Table 2. Average heuristic results on instances with $Q = 150$ and $Q = 250$

	#			$Q = 150$					$Q = 250$		
	pickups	S-shape	Mid point	L-Gap	Combined	ALNS	S-shape	Mid point	L-Gap	Combined	ALNS
G1	20	2169.00	2224.00	2205.67	2016.67	1234.33	1917.00	2036.00	1826.33	1852.00	1082.33
	30	2949.00	3014.33	2902.33	2850.00	1555.33	2638.67	2891.67	2715.67	2443.67	1357.67
	40	3665.33	3812.33	3558.33	3611.67	1904.67	3075.67	3281.67	3145.33	2967.00	1552.67
	50	4075.67	4223.33	4126.00	3972.33	2135.33	3754.00	3917.67	3892.00	3562.33	1831.33
	60	5181.33	5114.33	4912.67	5023.33	2540.33	4357.67	4311.00	4240.67	4157.00	2063.33
	70	5694.67	5680.00	5409.00	5600.67	2739.33	4816.33	4935.33	4788.33	4615.00	2259.33
	80	6159.33	6135.00	6012.33	5750.33	3046.33	5329.00	5226.67	5215.67	5029.33	2382.33
	90	7011.67	6878.00	6886.67	6877.00	3408.33	5651.67	5919.67	5579.00	5331.00	2580.00
	100	7663.67	7550.00	7452.67	7147.00	3716.67	5981.00	6084.33	6045.67	5897.33	2738.00
	Average	4952.19	4959.04	4829.52	<u>4761.00</u>	2475.63	4169.00	4289.33	4160.96	<u>3983.85</u>	1983.00
G2	20	2231.00	2441.00	2288.00	2213.00	1290.67	1959.33	2171.33	2077.00	1961.33	1171.00
	30	3093.33	3274.67	3154.67	2996.00	1641.33	2651.33	2950.00	2835.00	2747.00	1450.00
	40	3695.33	3757.67	3765.33	3728.33	1967.00	3398.33	3514.33	3527.00	3263.67	1693.67
	50	4150.00	4303.00	4295.33	4244.33	2219.67	3878.00	4322.33	3961.33	3814.67	1883.67
	60	5034.33	5403.00	5044.00	5249.33	2568.00	4445.67	4469.33	4487.00	4473.33	2135.33
	70	5608.67	5874.67	5714.67	5648.00	2797.00	4667.33	5089.33	4858.67	4993.33	2351.00
	80	6048.00	6376.33	6132.67	6142.67	3133.33	5201.00	5223.00	5250.67	5218.33	2446.67
	90	6921.67	7087.67	6973.67	6719.00	3466.33	5448.67	5808.00	5469.00	5519.00	2665.33
	100	7453.67	7680.67	7338.00	7625.67	3773.67	6180.67	6239.00	6042.67	6074.00	2764.44
	Average	<u>4915.11</u>	5133.19	4967.37	4951.81	2539.67	<u>4203.37</u>	4420.74	4278.70	4229.41	2062.35
G3	20	2311.00	2438.33	2205.67	2199.67	1279.67	2047.67	2147.33	2003.33	2025.67	1195.00
	30	2977.00	3183.00	3071.33	2982.67	1620.67	2686.00	2990.00	2695.00	2742.00	1439.00
	40	3590.00	3714.00	3455.67	3732.00	1919.00	3370.33	3484.67	3356.00	3230.67	1676.67
	50	4173.67	4400.00	4110.67	4248.67	2158.33	3736.33	4113.33	3758.33	3908.33	1862.33
	60	4911.00	5084.33	4980.33	5099.67	2487.33	4387.33	4501.33	4325.67	4455.00	2102.67
	70	5415.33	5651.00	5428.00	5779.00	2707.33	4660.00	5265.00	4734.00	4980.33	2328.67
	80	5949.00	6030.67	5904.67	6111.67	3011.67	5410.33	5243.00	5113.00	5224.33	2377.00
	90	6892.33	6828.33	6857.67	6738.00	3327.00	5371.67	5764.33	5353.00	5568.00	2606.33
	100	7262.67	7380.33	7132.67	7786.67	3586.67	5959.33	6125.33	5826.33	6023.33	2764.67
	Average	4831.33	4967.78	<u>4794.07</u>	4964.22	2455.30	4181.00	4403.81	<u>4129.41</u>	4239.74	2039.15
Overall average		4899.54	5020.00	4863.65	4892.35	2490.20	4184.46	4371.30	4189.69	4151.00	2028.16

5.4 *Exact algorithms results*

Tables 3 and 4 present the results of the best heuristic proposed (ALNS) compared to the results obtained by solving the two mathematical models, F1 and F2, when applying the branch-and-cut algorithms with the proposed cutting planes.

For each instance group and for each number of pickups, we provide the average results over all the instances and over all the number of aisles, with $Q = 150$ in Table 3 and $Q = 250$ in Table 4. We report the upper bound (UB) and the lower bound (LB) for models F1 and F2. We provide the average percentage gaps, given by the ratio $(\frac{UB-LB}{LB} \cdot 100)$. We also provide the average running time in seconds.

From Tables 3 and 4, we observe that the results provided by the ALNS metaheuristic are close to the best UBs yielded by both models. For instance, when $Q = 150$, ALNS gives an overall average distance of 2496.83 compared to 2482.68 and 2479.07 for F1 and F2. This remark also holds when $Q = 250$. Moreover, the ALNS algorithm provides near-optimal solutions within very short computing times (few seconds) compared to the time spent by both models to prove optimality or to provide better UBs. However, deriving better results at the expense of longer run times was one of our goals to provide the best possible results for the newly proposed testbed and serve the warehousing research community.

A deeper analysis of the formulations shows that the solutions are tight, and the optimality gap is in average 9.35% when $Q = 150$ for F1, and 6.91% for F2. Model F2 proves optimality over all instances with 20 pickups and almost all instances with 30 pickups within very short computing times. Both models had problems closing the gap and proving optimality for instances with more than 30 pickups.

Note that the time limit of 7200 seconds is often exceeded by F1. This is due to the fact that the time spent to instantiate model F1 is also considered in the total time, and the size of model F1 may become an issue when the instance size increases. For example, for instances including 100 pickups, almost two hours are needed only to create the model.

Finally, it is important to notice that the subset sum algorithm presented in Section 4.2.1 was able to reduce an average of 12.55% variables compared to a complete enumeration of all possible variables for model F1. In our test bed, we have observed a reduction of up to 23.56% in the number of variables, and a minimum reduction of 5.87%.

Regarding the three groups of instances, we observe that the gap between the upper and lower bounds is slightly lower for group G1 for both models. There are no major differences among groups in terms of the traveled distance except for very small variations. Indeed, it seems more difficult to solve instances where the products are placed in groups, in such a way as to facilitate the picking work.

Table 3. Average results of the exact algorithms per group of instances and number of pickups on instances with $Q = 150$

Group	Pickups	ALNS	Time (s)	Capacity indexed formulation (F1)				Two-index formulation (F2)			
				UB	LB	%Gap	Time (s)	UB	LB	%Gap	Time (s)
G1	20	1251.00	1.53	1232.00	1232.00	0.00	118.27	1232.00	1232.00	0.00	8.00
	30	1568.00	3.33	1542.67	1521.86	1.37	1577.60	1542.67	1540.30	0.15	541.60
	40	1928.00	5.53	1892.67	1798.71	5.22	7278.67	1888.67	1873.72	0.80	1656.40
	50	2127.33	9.60	2120.67	1937.19	9.47	7604.07	2104.33	2031.58	3.58	5321.40
	60	2530.33	16.40	2529.33	2319.77	9.03	8678.60	2525.67	2380.87	6.08	7200.67
	70	2734.00	24.47	2733.33	2491.91	9.69	9527.27	2732.33	2595.60	5.27	7201.00
	80	3040.67	31.20	3038.33	2790.07	8.90	10031.27	3035.00	2854.34	6.33	7201.20
	90	3403.33	50.33	3402.67	3083.69	10.34	16523.53	3403.33	3153.14	7.93	7202.67
	100	3722.67	65.60	3722.67	3373.26	10.36	19710.80	3722.67	3426.66	8.64	7202.93
	Average	2478.37	23.11	2468.26	2283.16	8.11	9005.56	2465.19	2343.13	5.21	4837.32
G2	20	1300.33	1.07	1287.67	1252.50	2.81	188.53	1287.67	1287.67	0.00	99.73
	30	1661.67	3.20	1627.67	1601.29	1.65	3185.60	1627.00	1626.52	0.03	1066.07
	40	2012.33	6.13	1962.67	1820.88	7.79	7140.73	1955.67	1890.01	3.47	4403.53
	50	2230.00	9.40	2221.33	1993.66	11.42	7946.80	2193.00	2079.29	5.47	6028.47
	60	2565.33	17.13	2557.00	2301.68	11.09	8804.27	2546.00	2371.23	7.37	6923.13
	70	2803.33	24.80	2797.33	2519.51	11.03	9967.53	2798.00	2574.98	8.66	7201.13
	80	3129.67	33.73	3129.33	2805.88	11.53	11105.93	3113.00	2848.36	9.29	7201.60
	90	3453.67	51.00	3453.67	3116.54	10.82	14786.20	3453.67	3165.65	9.10	7202.33
	100	3783.00	67.53	3781.00	3392.42	11.45	17634.47	3783.00	3420.28	10.60	7202.93
	Average	2548.81	23.78	2535.30	2311.60	9.68	8973.34	2528.56	2362.66	7.02	5258.77
G3	20	1275.33	1.67	1253.00	1140.00	9.91	211.47	1253.00	1253.00	0.00	7.67
	30	1614.67	3.13	1592.33	1592.33	0.00	1559.87	1592.33	1586.33	0.38	850.07
	40	1958.33	5.60	1871.67	1781.19	5.08	6868.73	1879.33	1816.53	3.46	3475.00
	50	2139.67	8.87	2119.00	1939.38	9.26	7849.73	2109.33	1995.65	5.70	6013.20
	60	2490.67	16.27	2474.00	2206.25	12.14	8744.40	2473.00	2254.63	9.69	6729.33
	70	2706.67	20.80	2706.67	2431.64	11.31	9096.13	2702.67	2448.13	10.40	7201.00
	80	3042.33	31.00	3041.67	2678.45	13.56	11614.67	3041.67	2679.62	13.51	7201.80
	90	3360.67	43.13	3360.67	2991.94	12.32	12734.93	3358.67	3029.84	10.85	7201.80
	100	3581.33	66.87	3581.33	3188.99	12.30	16577.47	3581.33	3195.08	12.09	7202.13
	Average	2463.30	21.93	2444.48	2216.69	10.28	8361.93	2443.48	2250.98	8.55	5098.00
Overall average		2496.83	22.94	2482.68	2270.48	9.35	8780.28	2479.07	2318.93	6.91	5064.70

Table 5 presents the number of optimal solutions obtained by both models over the 810 instances. These results are separated in groups of instances, number of aisles, and by the value of the capacity Q. This helps highlight the effect of each of these characteristics. In the first and second lines we present the capacity indexed formulation F1 and the two-index formulation F2 with the two capacities, $Q = 150$ and $Q = 250$. We then present the results for each group of instances and each number of aisles. The number of optimal solutions is slightly higher with F2. This corroborates the results already observed from the gaps and running time of these models. A transversal analysis of Tables 3, 4, and 5 points out again the difficulty of the problem. We observe that the best exact algorithm is able to prove optimality for only 27% of the instances.

Table 6 presents a deeper statistical comparison between the proposed formulations. For each model, we provide averages for the percentage optimality gap, the computation time in seconds, the number of variables, and constraints generated along with the number of cuts added to fractional and integer solutions, and finally the number of nodes explored in the branch-and-bound tree. The last column of the table shows the relative difference of F1 with respect to F2.

The first outstanding result from Table 6 is related to the huge number of variables and constraints generated in F1. These two figures are more than 100 times higher than the number of variables and constraints generated in F2. Although the subset sum algorithm helps decreasing the size of F1 by removing irrelevant variables, the size of formulation F1 remains huge compared to F2 and requires much more time to load the instance and create the model.

Table 4. Average results of the exact algorithms per group of instances and number of pickups on instances with $Q = 250$

Group	Pickups	ALNS	Time (s)	Capacity indexed formulation (F1) UB	LB	%Gap	Time (s)	Two-index formulation (F2) UB	LB	%Gap	Time (s)
G1	20	1093.67	1.73	1064.33	1048.65	1.50	1644.87	1063.33	1063.33	0.00	1.07
	30	1388.00	4.40	1360.33	1314.20	3.51	4645.33	1355.67	1355.67	0.00	45.80
	40	1568.67	6.87	1559.33	1422.70	9.60	7523.00	1530.33	1517.27	0.86	2241.00
	50	1859.33	12.07	1858.67	1584.14	17.33	8551.40	1836.00	1701.57	7.90	7081.73
	60	2057.67	20.33	2057.00	1746.16	17.80	12133.93	2016.33	1847.55	9.14	7065.67
	70	2236.67	31.80	2236.67	1911.78	16.99	14281.20	2228.00	1987.49	12.10	7200.93
	80	2387.00	46.07	2387.00	1999.96	19.35	17920.47	2377.33	2043.08	16.36	7201.13
	90	2568.00	73.67	2568.00	2157.94	19.00	35708.87	2567.33	2214.82	15.92	7202.07
	100	2716.67	86.27	2716.67	2283.53	18.97	41148.93	2716.00	2323.99	16.87	7201.73
	Average	1986.19	31.47	1978.67	1718.78	15.12	15950.89	1965.59	1783.86	10.19	5026.79
G2	20	1209.33	1.67	1164.67	1149.44	1.32	2309.47	1164.67	1164.67	0.00	0.33
	30	1499.67	3.87	1470.33	1357.62	8.30	5233.27	1448.00	1444.69	0.23	529.07
	40	1728.67	7.40	1714.67	1526.73	12.31	8008.13	1699.00	1641.20	3.52	4543.80
	50	1935.67	12.07	1935.67	1613.44	19.97	9346.67	1891.33	1673.86	12.99	6787.13
	60	2179.67	22.80	2179.67	1810.35	20.40	12049.20	2141.67	1876.63	14.12	6893.53
	70	2379.67	31.20	2377.67	1973.70	20.47	15551.13	2354.00	2005.31	17.39	7201.27
	80	2473.00	46.27	2473.00	2037.14	21.40	21498.93	2473.00	2072.88	19.30	7201.53
	90	2694.67	65.47	2694.67	2230.08	20.83	33595.00	2694.00	2266.36	18.87	7202.27
	100	2825.33	80.07	2825.33	2351.37	20.16	40648.80	2823.33	2356.54	19.81	7202.47
	Average	2102.85	30.09	2092.85	1783.32	17.36	16471.18	2076.56	1833.57	13.25	5284.60
G3	20	1235.33	1.53	1193.33	1177.85	1.31	2571.73	1193.00	1193.00	0.00	0.60
	30	1481.00	4.00	1452.00	1357.24	6.98	4982.13	1441.00	1441.00	0.00	178.53
	40	1730.00	6.87	1720.00	1508.52	14.02	7646.07	1676.00	1633.55	2.60	3854.47
	50	1876.00	13.27	1876.00	1602.06	17.10	8903.73	1856.67	1680.63	10.47	6233.33
	60	2149.00	20.27	2149.00	1795.72	19.67	11513.80	2106.33	1817.12	15.92	7200.73
	70	2321.67	28.13	2321.67	1938.48	19.77	14725.33	2304.33	1962.90	17.39	7201.07
	80	2384.67	44.53	2384.67	2008.36	18.74	22755.27	2383.33	2007.33	18.73	7201.67
	90	2641.00	55.67	2641.00	2179.95	21.15	28728.27	2640.33	2202.66	19.87	7201.73
	100	2780.67	80.00	2780.00	2311.15	20.29	35836.53	2778.67	2291.00	21.29	7201.60
	Average	2066.59	28.25	2057.52	1764.37	16.61	15295.87	2042.19	1803.24	13.25	5141.53
Overall average		2051.88	29.94	2043.01	1755.49	16.38	15905.98	2028.11	1806.89	12.24	5150.97

Table 5. Number of optimal solutions per group of instances and capacity of the truck

		G1						G2						G3						
	Capacity	3	4	5	6	7	Total	3	4	5	6	7	Total	3	4	5	6	7	Total	Total
F1	Q=150	6	7	5	6	6	30	4	5	4	6	6	25	8	7	6	7	5	33	88
	Q=250	8	9	10	11	8	46	8	7	9	8	9	41	8	8	9	9	7	41	128
F2	Q=150	5	4	4	4	6	23	4	4	4	3	5	20	4	4	3	4	5	20	63
	Q=250	9	9	8	9	9	44	6	7	6	8	10	37	7	8	7	9	10	41	122

However, F1 uses almost no cuts during the search process to limit the size of the solution while F2 invokes a huge number of cuts for both fractional and integer solutions. Indeed, over all instances, F2 uses on average 316.2 cuts on fractional solutions and 9703.5 on integer ones. This is due to the fact that F1 does not need to add sub-tours elimination constraints or self-capacity constraints during the search process. Moreover, for both models and on instances with more than 40 pickups, the number of visited nodes gradually decreases while the instances size increases. This is due to the increasing number of cuts generated while the instances size increases and then a smaller tree size is explored. This shows that the average number of visited nodes in F1 is less than the number of visited nodes in F2 by 33%. These results clearly show that formulation F1 is again outperformed by formulation F2.

Table 6. Statistical comparison of models F1 and F2 over all instances per number of pickups

	Pickups	20	30	40	50	60	70	80	90	100	Average	vs. F2
F1	% Gap	3.2	4.1	9.2	14.5	15.5	15.0	15.7	15.8	15.6	12.0	42%
	Time (s)	1174.1	3530.6	7410.9	8367.1	10320.7	12191.4	15821.1	23679.5	28592.8	12343.1	142%
	# Variables	37479.6	72740.4	130246.9	203011.6	280638.7	384998.3	497471.6	665514.9	793555.8	340628.6	10528%
	# Constraintes	12962.8	23821.5	37205.6	57058.8	75440.1	98905.7	130409.9	159564.4	194357.4	87747.3	11308%
	Fractional cuts	0.6	1.1	1.4	0.2	0.1	0.2	0.1	0.0	0.0	0.4	-100%
	Integer cuts	12.6	17.2	23.2	10.9	7.3	5.5	6.7	2.3	3.3	9.9	-100%
	# Nodes	12998.2	32487.2	31799.1	13686.6	4279.6	2436.6	1504.5	373.6	320.4	11098.4	-33%
F2	% Gap	0.0	0.2	2.7	8.0	10.9	12.0	14.1	13.8	14.9	8.5	
	Time (s)	19.6	535.2	3362.4	6244.2	7002.2	7201.1	7201.5	7202.1	7202.3	5107.8	
	# Variables	330.9	703.7	1218.8	1895.9	2698.1	3691.6	4748.4	6121.6	7435.2	3204.9	
	# Constraintes	101.5	198.8	312.9	486.3	671.4	880.7	1163.6	1378.6	1728.5	769.2	
	Fractional cuts	96.7	277.4	455.4	581.3	476.9	283.7	273.4	206.9	193.9	316.2	
	Integer cuts	128.5	1144.2	5485.5	10890.1	14157.9	14544.8	14606.4	13246.6	13127.8	9703.5	
	# Nodes	1838.3	19307.3	35759.4	34536.0	19465.7	14024.2	10649.1	6882.4	5773.8	16470,7	

6. Conclusions

We have tackled a real-world-based and rich order picking problem arising in the grocery retail industry which, to our knowledge, was studied here for the first time. This practical problem extended classical warehousing problems by incorporating more challenges regarding the physical characteristics of the products to be picked. Specifically, products can support a maximum weight when being transported on a lift-truck used for warehouse picking, some products are more fragile than others, and products belonging to food and non-food categories must be picked in a given order such as to avoid contamination. We have adapted four classical order picking heuristics from the warehousing literature to handle these new features. Notably, we have shown how the S-shape, the largest gap, the mid point, and the combined heuristics can yield feasible solutions within very short computation times. Moreover, we have also proposed a more powerful metaheuristic, ALNS, which outperforms the best results obtained by the four heuristics. We have presented two mathematical models including known and new valid inequalities for this challenging picking problem. The first one is the capacity-indexed formulation, which is a based compact single commodity flow using binary variables to indicate the flow on each arc. The second formulation is a two-index flow formulation, in which individual hand-lift trucks are not explicitly identified. We have proposed exact algorithms for their resolution.

We have tested the proposed heuristics, metaheuristic, and the two formulations on large sets of newly generated and realistic instances. The results show the effectiveness of the proposed heuristics in finding high-quality solutions within a negligible computation time. The solutions provided by the ALNS metaheuristic dominated those from the other heuristics, as it traversed smaller distances due to fewer tours. Moreover, we have been able to prove optimality for several instances and to obtain better solutions and tight gaps with two mathematical models that were solved with classical and ad-hoc valid inequalities and cutting planes. Extensive tests have shown that the proposed exact algorithms outperformed the solutions of the heuristic algorithms. Moreover, we have shown that the two-index formulation outperforms the capacity-indexed formulation for this problem. The most remarkable conclusion for this work is that the running times of the heuristics are very low, and those of the exact algorithms are still acceptable, allowing for their usage in practical applications.

Our future works aim to show how the proposed solution methods can be adapted to cover a variety of other warehousing applications, in other industries and under different assumptions. In addition, there might be a strong correlation between the instances characteristics, e.g., warehouse design, number of aisles, capacity of the truck, and the results obtained. Consequently, more research is needed to demonstrate such interactions through design experiments, as they are supposed to significantly help find the best warehouse design with respect to the routing policy under different conditions. Similar work has been already performed in Chackelson et al. (2013). Future research that supports this study may include more computational experiments to assess the impact of each precedence constraint on the performance of the heuristic and exact algorithms. Previous studies evaluating the impact of different operating conditions on the route distance exist, e.g., Petersen (1997), Petersen and Schmenner (1999).

Acknowledgments

This research was partly supported by grants 2014-05764 and 0172633 from the Canadian Natural Sciences and Engineering Research Council. This support is gratefully acknowledged. We also thank Calcul Québec for providing computing facilities. We also thank Calcul Québec for providing computing facilities. We thank an associate editor and two anonymous referees for their comments on an earlier version of this paper.

References

K. B. Ackerman. *Practical Handbook of Warehousing.* Springer Science & Business Media, New York, 2013.

M. Bortolini, M. Faccio, M. Gamberi, and R. Manzini. Diagonal cross-aisles in unit load warehouses to increase handling performance. *International Journal of Production Economics*, 170:838–849, 2015.

T. Chabot, L. C. Coelho, J. Renaud, and J.-F. Côté. Mathematical models, heuristic and exact method for order picking in 3d-narrow aisles. Technical Report CIRRELT-2015-18, Québec, Canada, 2015.

C. Chackelson, A. Errasti, D. Ciprés, and F. Lahoz. Evaluating order picking performance trade-offs by configuring main operating strategies in a retail distributor: a design of experiments approach. *International Journal of Production Research*, 51(20):6097–6109, 2013.

D. M.-H. Chiang, C.-P. Lin, and M.-C. Chen. The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres. *Enterprise Information Systems*, 5(2):219–234, 2011.

J-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

R. de Koster and E. Van der Poort. Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. *IIE Transactions*, 30(5):469–480, 1998.

R. de Koster, E. S. Van der Poort, and K. J. Roodbergen. When to apply optimal or heuristic routing of orderpickers. In B. Fleischmann, J. A. E. E. Van Nunen, M.G. Speranza, and P. Stähly, editors, *Advances in Distribution Logistics*, pages 375–401. Springer, Berlin, 1998.

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

J. de Vries, R. de Koster, and D. Stam. Exploring the role of picker personality in predicting picking performance with pick by voice, pick to light and RF-terminal picking. *International Journal of Production Research*, pages 1–15, 2015.

R. Dekker, M. B. M. De Koster, K. J. Roodbergen, and H. Van Kalleveen. Improving order-picking response time at ankor's warehouse. *Interfaces*, 34(4):303–313, 2004.

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

R. W. Hall. Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, 25(4): 76–87, 1993.

S. Henn, S. Koch, and G. Wäscher. Order batching in order picking warehouses: A survey of solution approaches. In R. Manzini, editor, *Warehousing in the Global Supply Chain*, pages 105–137. Springer, London, 2012.

M. Hompel and T. Schmidt. *Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems.* Springer, Berlin, 2006.

L. Junqueira, R. Morabito, and D. S. Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85, 2012.

R. Lahyani, F. Semet, and B. Trouillet. Vehicle routing problems with scheduling constraints. In *Metaheuristics for Production Scheduling*, pages 433–463. Wiley Online Library, United Kingdom, 2014.

R. Lahyani, L. C. Coelho, and J. Renaud. Alternative formulations and improved bounds for the multi-depot fleet size and mix vehicle routing. Technical Report CIRRELT-2015-36, Québec, Canada, 2015.

G. Laporte. Generalized subtour elimination constraints and connectivity constraints. *Journal of the Operational Research Society*, 37(5):509–514, 1986.

W. Lu, D. McFarlane, V. Giannikas, and Q. Zhang. An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, 248(1):107–122, 2016.

J. Lysgaard. *CVRPSEP: A package of separation routines for the capacitated vehicle routing problem.* Department of Management Science and Logistics, Aarhus School of Business, Denmark, 2003.

R. Manzini. *Warehousing in the Global Supply Chain.* Springer, London, 2012.

M. Matusiak, R. de Koster, L. Kroon, and J. Saarinen. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *European Journal of Operational Research*, 236 (3):968–977, 2014.

C. P. Medard and N. Sawhney. Airline crew scheduling from planning to operations. *European Journal of Operational Research*, 183(3):1013–1027, 2007.

M. W. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

J. Park and B.-I. Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010.

A. Pessoa, E. Uchoa, and M. V. S. Poggi de Aragão. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, 54(4):167–177, 2009.

C. G. Petersen. An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, 17(11):1098–1111, 1997.

C. G. Petersen and G. R. Aase. A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92(1):11–19, 2004.

C. G. Petersen and R. W. Schmenner. An evaluation of routing and volume-based storage policies in an order picking operation. *Decision Sciences*, 30(2):481–501, 1999.

C. G. Petersen, G. R. Aase, and D. R. Heiser. Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, 34 (7):534–544, 2004.

J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, 1978.

M. V. S. Poggi de Aragão and E. Uchoa. New exact algorithms for the capacitated cehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 59–86. MOS-SIAM Series on Optimization, Philadelphia, 2014.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

K. J. Roodbergen and R. de Koster. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43, 2001.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK, 1997.

J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. *Facilities Planning*. John Wiley & Sons, New York, 2010.

P. Toth and D. Vigo. The family of vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 1–23. MOS-SIAM Series on Optimization, Philadelphia, 2014.

G. Wäscher. Order picking: a survey of planning problems and methods. In H. Dyckhoff, R. Lackes, and J. Reese, editors, *Supply Chain Management and Reverse Logistics*, pages 323–347. Springer, Berlin, 2004.

E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. The vehicle routing problem with simultaneous pick-ups and deliveries and two-dimensional loading constraints. *European Journal of Operational Research*, 251(2):369–386, 2016.