Computers & Industrial Engineering 160 (2021) 107517

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A heuristic approach for the online order batching problem with multiple pickers

Sergio Gil-Borrás [a], Eduardo G. Pardo [b,*], Antonio Alonso-Ayuso [b], Abraham Duarte [b]

[a] Dept. Sistemas Informáticos, Universidad Politécnica de Madrid, Spain
[b] Dept. Computer Science, Universidad Rey Juan Carlos, Spain

ARTICLE INFO

Keywords:
Online Order Batching Problem
Multiple pickers
Multistart search
Variable Neighborhood Descent

ABSTRACT

The Online Order Batching Problem with Multiple Pickers (OOBPMP) consists of optimizing the operations related to the picking process of orders in a warehouse, when the picking policy follows an order batching strategy. In this case, this variant of the well-known Order Batching Problem considers the existence of multiple workers in the warehouse and an online arrival of the orders. We study three different objective functions for the problem: minimizing the completion time, minimizing the picking time, and minimizing the differences in the workload among the pickers. We have identified and classified all previous works in the literature for the OOBPMP. Finally, we propose a multistart procedure hybridized with a Variable Neighborhood Descent metaheuristic to handle the problem. We test our proposal over well-known instances previously reported in the literature by empirically comparing the performance of our proposal with previous methods in the state of the art. The statistical tests corroborated the significance of the results obtained.

1. Introduction

The Online Order Batching Problem (OOBP) is an optimization problem which occurs in a warehouse when the picking policy follows an order batching strategy, i.e., orders are grouped into batches prior to be picked, and all orders in the same batch are collected together. The OOBP is considered a dynamic optimization problem since orders are received online in the system, which means that the arrival of orders occurs continuously (24 h a day/7 days a week) while the optimization algorithms are running. Therefore, an order can arrive to the system while the picking process of other orders is in progress. The main task within the OOBP consists of determining the best assignation of the orders into batches of a maximum predefined capacity (the maximum load that a picker can carry at the same time), with the aim of performing an efficient picking operation. However, this assignation can only be considered as one of the subproblems that need to be handled within the OOBP. Other necessary operations include: establishing a sequence of the constructed batches, assigning each batch to a picker, determining the moment in the time for starting the picking (time window), or designing the route to follow by the picker. Despite the fact that the previous ones are also optimization problems that could be considered in isolation, in the context of the OOBP, they are only parts that need to be taken into account to calculate the main objective function.

Additionally, the picking operation in this context is also influenced by different static and dynamic parameters (Petersen, 1997). Among the static ones, we can find parameters related to the warehouse design such as: the number of blocks, the number of aisles, the width of each aisle, the number of depots, the position of the depots, etc. Also, the distribution of the products in the warehouse can sometimes be considered as a static parameter (other times the same product is stored in a different position through the time). Furthermore, products can be placed at random in the warehouse, or following an ABC distribution (i.e., the most demanded products are placed closer to the depot). Additionally, it is possible to consider one or more locations for each kind of products. On the other hand, the dynamic parameters include aspects such as: the variable number of pickers, the number and size of the orders arrived to the warehouse, the existence of due dates in some orders, or the priorities among the items in the same order. Also, if the storage location of the product varies through the time, it can be considered a dynamic parameter too.

Previous works in the literature of the OOBP with multiple pickers (also known in the literature as OOBPMP) have reported different objective functions within this context. Also, they have studied several important parameters for the problem. However, in general, they do not provide a wide comparison framework. Our main hypothesis is the existence of relationships among the optimization of some of the previously studied objective functions. Moreover, there might exist

* Corresponding author.
E-mail addresses: sergio.gil@upm.es (S. Gil-Borrás), eduardo.pardo@urjc.es (E.G. Pardo), antonio.alonso@urjc.es (A. Alonso-Ayuso), abraham.duarte@urjc.es (A. Duarte).

https://doi.org/10.1016/j.cie.2021.107517
Received 26 December 2020; Received in revised form 15 May 2021; Accepted 22 June 2021
Available online 26 June 2021

important parameters, such as the number of pickers, the time horizon considered, or the congestion in the arrival of orders that could have a deep impact on the performance of the algorithms, depending on the considered objective function. Therefore, the main objective of this research is to study the different objective functions tackled in the literature in the context of the OOBPMP. Particularly, we compare the performance of the most relevant previous methods in combination with parameters such as the time horizon or the number of pickers, which influence the congestion rate in the system. Furthermore, we explore the existence of algorithms with good performance in a wide range of scenarios.

In this paper, we focus our attention on the OOBP with multiple pickers for rectangular-shaped single-block warehouses by studying three objective functions: the minimization of the total picking time, the minimization of the maximum completion time, and the minimization of the differences in the workload of the pickers. These objective functions are explained in detail Section 3. Our main contributions are: (i) a comprehensive classification of the studied problem within the OBP literature and the study and analysis of the previous methods in the state of the art for the OOBPMP; (ii) a new algorithmic proposal based on the combination of a multistart procedure (MS) with the Variable Neighborhood Descent (VND) metaheuristic, named MS-VND; (iii) the use and study of several objective functions to tackle the OOBPMP; (iv) an empirical study of the influence of two important parameters (the congestion in the arrival of orders, and the number of pickers) in the performance of the algorithms; and (v) the improvement of the results of the state of art methods, previously proposed for the same problem, supported by statistical tests.

The rest of this paper is structured as follows: in Section 2, we provide a comprehensive classification of the studied problem in the literature and we review the main approaches related with the online order batching with multiple pickers. In Section 3, we describe in detail the problem tackled in this work. Then, in Section 4, we present the algorithms proposed in this paper to tackle the OOBPMP. Particularly, we use a multistart procedure combined with a Variable Neighborhood Descent as a local search procedure. Then, a wide number of numerical experiments are compiled in Section 5. Finally, conclusions and open research lines are given in Section 6.

2. State of the art

This paper is focused on a variant of the online order batching problem, which considers multiple pickers in a warehouse. As part of the picking process of orders in a warehouse, the order batching strategy has guided practitioners in the field to a wide range of related optimization problems. Within this family, problems can be classified as single/multi-picker and offline/online.

The majority of the previous studies in the literature handle the variant with the restriction of considering a single picker (Henn, Koch, & Wäscher, 2012), while a more general and realistic scenario, the existence of multiple pickers in the warehouse, has been less studied. In fact, the multiple-pickers version is a generalization of the case of a single picker. Similarly, offline versions of the OBP (i.e., all orders are available at the beginning of the process) have received more attention in the literature than the more realistic online versions (i.e., orders arrive to the system while the picking process is in progress). Notice that the online version of the problem is a generalization of the offline one.

The most common objective functions studied in the literature of order batching problems are the minimization of the picking time (Chen, Wei, & Wang, 2018; Rubrico, Higashi, Tamura, & Ota, 2011), the minimization of the traveled distance (Öncan, 2015; Pérez-Rodríguez & Hernández-Aguirre, 2015), or the minimization of the tardiness (Menéndez, Bustillo, Pardo, & Duarte, 2017; Zhao, Jiang, Bao, Wang, & Jia, 2019). However, the appearance of multiple workers uncovers additional objective functions such as: finding a balance in the workload of the pickers (Zhang, Wang, Chan, & Ruan, 2017), or minimizing the average waiting time of the pickers, due to blocking situations among them (Chen, Wang, Qi, & Xie, 2013; Chen, Wang, Xie, & Qi, 2016; Hahn et al., 2017). Similarly, some objective functions only make sense when considering the online version of the problem such as: minimizing the turnover time (Gil-Borrás, Pardo, Alonso-Ayuso, & Duarte, 2020b; Tang & Chew, 1997) or minimizing the completion time (Gil-Borrás, Pardo, Alonso-Ayuso, & Duarte, 2020a; Henn, 2012).

In this paper, we focus our attention on the Online Order Batching Problem with multiple pickers. For this variant of the OBP, we have found eight different previous proposals in the literature. However, taking a closer look to each of them, we can find differences among the constraints considered.

As far as we know, the first approach for the OOBPMP was proposed by Yu and De Koster (2009). In that paper, the authors dealt with a version of the OOBPMP which considers different picking zones within the warehouse for different pickers. Additionally, the warehouse instances used present random storage policy, and the objective function was the minimization of the average throughput time. Yu and De Koster (2009) proposed an approximation model based on the queuing network theory, to tackle the batching problem. They also considered a S-shape routing strategy and a Poisson distribution for the order arrivals.

Rubrico et al. (2011) tackled the Online Rescheduling Problem with multiple pickers. In this variant, they considered the existence of static and dynamic arrival of orders, with the restriction that the newly arrived orders were composed of only one type of product. The studied objective function in this case was the minimization of the makespan. Rubrico et al. (2011) proposed a Steepest Descent Insertion method with a Multistage Rescheduling strategy to perform the batching task. They also considered the S-shape routing algorithm.

Zhang et al. (2017) tackled the OOBPMP with the aim of minimizing the turnover time, which is also known as the maximum completion time of all batches (i.e., the time needed to collect all orders including waiting, routing, batching and service time). Further than the pursued objective function, they also reported the obtained average workload and average idle time per picker. Zhang et al. (2017) proposed a Hybrid Rule-Based Algorithm (which includes a strategy based on seed methods for batching), and they used the S-shape routing strategy.

In 2018, first, Chen et al. (2018), studied the OOBPMP for a multiple-block warehouse with narrow aisles (two pickers cannot cross their routes). In this case, the authors considered that the batches can be altered once the picker has already begun picking, and that an order can be split into more than one batch. In this sense, not only the order arrival is dynamic but also the batch composition might change at the picking time. The studied objective function consists of minimizing the service time of a single order. Chen et al. (2018) proposed a heuristic batching strategy named Green Area, and they compared their proposal with several time-window-based strategies, consisting of batching together the orders arrived in a particular chunk of time. They studied either the fixed time window and variable time window strategies. For the routing task, they considered both: the S-Shape and the Largest Gap routing algorithms.

Also in 2018, Van Der Gaast, Jargalsaikhan, and Roodbergen (2018) considered the OOBPMP with the possibility of modifying a batch which is currently being picked, by adding new orders arrived to the warehouse. In this case, the objective function studied was the minimization of the order throughput time (the time that an order remains in the system, which is also known in the literature as the order turnover time). However, despite of minimizing that objective function, they reported additional metrics (orders in backlog, tour duration, replanned tours, picker walking distance, etc.) which were used in other papers as objective functions (Henn, Koch, Doerner, Strauss, & Wäscher, 2010; Menéndez, Pardo, Alonso-Ayuso, Molina, & Duarte, 2017; Öncan, 2015; Scholz, Schubert, & Wäscher, 2017; Zhang et al., 2017). In this case, the layout of the studied warehouse includes

multiple blocks. Van Der Gaast et al. (2018) proposed the combination of a method based on linear programming with column generation, together with Tabu Search and Branch-and-bound pricing, to tackle the problem. This time, the routing strategy was based on three different proposals: the Nearest Neighbor, the S-shape, and the Largest Gap.

A more recent approach in the literature within this context is found in Hojaghania, Nematian, Shojaiea, and Javadi (2019), where the authors studied the maximum turnover time of an online multipicker variant, which considers different zones within the warehouse, each of them operated by a picker. In this paper, the objective function, further than the turnover time, includes the idle time of the pickers. They used the proposal in Zhang et al. (2017) as a baseline to compare their approach. The algorithmic proposal included an Ant Colony Algorithm and an Artificial Bee Colony for the batching task, and the S-Shape strategy for the routing task.

Finally, Alipour, Mehrjedrdi, and Mostafaeipour (2020) made an extension of a previous algorithm proposed by Henn (2012). The original work was designed for an online context with just a single picker with the aim of minimizing the completion time. This time, the authors studied the maximum completion time in a multipicker scenario by using the Iterated Local Search algorithm proposed by Henn as the batching method, and the S-shape and the Largest-gap as the routing methods. The proposal was compared against Clarke & Wright II (C&W II) and First Come First Served (FCFS) methods as baseline.

As it is possible to observe, despite the fact that all previous variants handle the Online Order Batching Problem with multiple pickers, the additional constraints or objective functions studied are not the same, which sometimes make difficult the comparison among the methods.

In this paper, we handle the Online Order Batching Problem with Multiple Pickers by studying the minimization of the maximum completion time. However, we also evaluate the workload balance of the pickers and the picking time of our solutions, to provide a wide comparison framework for other approaches. In this matter, we consider the following characteristics/restrictions for the problem: the warehouse has a single block (instead of the multiple blocks considered in Chen et al. (2018) or Van Der Gaast et al. (2018)); there are no narrow-aisles restrictions (as the ones introduced in Chen et al. (2018)); once a picking route has started, the batch and the route cannot be modified (as it happens in Chen et al. (2018) or Van Der Gaast et al. (2018), where newly arrived orders can be added to batches being collected at that moment, and the associated routes adapted); orders cannot be split in multiple batches (as it is the case in Chen et al. (2018)); the whole warehouse is handled as a single zone in terms of picking (instead of considering multiple zones as in Yu and De Koster (2009)); orders can be composed of different kind of products (unlike in the dynamic arrival of orders used in Rubrico et al. (2011)).

With the previous assumptions at hand and to end this section, we analyze in depth the two most similar previous works to our proposal, which are used in the experimental section as a comparison framework for our algorithms: Zhang et al. (2017) and Alipour et al. (2020).

Particularly, Zhang et al. (2017) proposed a batching method based on the "seed" strategy. This strategy had been previously introduced in the context of clustering problems (Ho & Tseng, 2006). The adaptation from a clustering problem to the batching problem is trivial, since it consists of selecting an order (that will represent the "seed") as a centroid of an empty batch (which in this case represents the cluster). Then, other available orders might be added to the same batch, depending on the similarity with respect to the selected seed order. Notice that the addition of orders to that batch is bounded by the maximum capacity of the batch. The general strategy based on "seed methods" consists of deciding the criterion to select the seed order and determining the similarity function between orders. Zhang et al. (2017) used the Smallest Arrival Time rule to select the "seed" order, which consists of selecting the earliest available order arrived to the system. Also, they used the Aisle-Time-Based strategy to determine the similarity, which takes into consideration two variables: the percentage of common products between the compared orders; and the proximity of the arrival time between the orders compared. In both cases, the selection is made on a greedy basis. Once no other orders can be added to the batch (i.e., the batch is full), the algorithm chooses a new seed order and so on. Further details of the proposed method can be found in Zhang et al. (2017). This method was tested over the data sets provided by Henn (2012) and the arrival of orders is scheduled on a time horizon of 4 h.

The proposal made by Alipour et al. (2020) is based on the well-known Iterated Local Search (ILS) previously proposed in Henn (2012), but adapted to the multipicker scenario. The initial solution was constructed based on the First Come First Serve strategy. That solution was then improved by the ILS method, which is formed by two different phases: perturbation and improvement. The perturbation selects two batches at random and exchanges n orders also selected at random. The improvement phase follows a first improvement strategy and it is based on two different neighborhoods: swap and shift moves. The swap move selects two orders in different batches and exchanges their assignation. The shift move inserts one order in another batch. Once all batches have been conformed, the authors studied different strategies for selecting the next batch to be collected: FIRST, SHORT, LONG, and SAV (see Alipour et al. (2020), Henn (2012) for a detailed description of each of them) being FIRST the most suitable approach. The order is then assigned to the first picker who becomes available. Finally, the S-shape and the Largest-gap were tested and compared as routing methods. The experiments were conducted only with two pickers over the instances reported in Henn (2012) and the arrival of orders was scheduled on a time horizon of 8 h.

3. Problem description

The OOBPMP tackled in this paper consists of performing an efficient picking operation of all products within the orders arrived online to a warehouse, by following an order picking strategy based on batches, when multiple pickers are available. The OOBPMP is a dynamic optimization problem which studies the efficient picking of orders which arrive online (24/7) to a warehouse. Since the described scenario is a non-stopping context, to study the problem and the associated proposals, it is necessary to observe the behavior of the algorithms in a particular chunk of time (denoted as time horizon).

An order is a list of products demanded by the same customer at the same time. The products in the same order must be collected together (i.e., orders cannot be splitted into more than one batch). A batch is a group of orders with a predefined maximum capacity. It is assumed that no order is larger than the maximum capacity of a batch. Each batch is assigned to one picker and all orders in the same batch are collected together in a single route through the warehouse. An important issue of the OOBPMP is that the orders are not fully available at the beginning of the process, but they arrive at the warehouse while the picking operation has already begun. This is why the problem is considered online. To handle online problems, it is necessary to define a time horizon and to observe the behavior of the proposals in that chunk of time. An additional characteristic of the OOBPMP is that there are multiple pickers in the warehouse available to perform the picking operation. Notice that we will not consider interblocking situations (aisles in the warehouse are wide enough for allowing several pickers crossing their routes).

When solving the OOBPMP, it is necessary to handle several subproblems: to decide when to consider the new orders arrived to the system (adding); to determine how the orders are grouped into batches (batching); to choose which batch is going to be collected next (selecting/sequencing); to determine which picker is going to retrieve that batch (assigning); to set the moment in the time when the picker starts its route (waiting); and to design the route that the picker will follow (routing).

To better understand the processes involved in the OOBPMP and the focus of this research, we introduce the activity diagram depicted

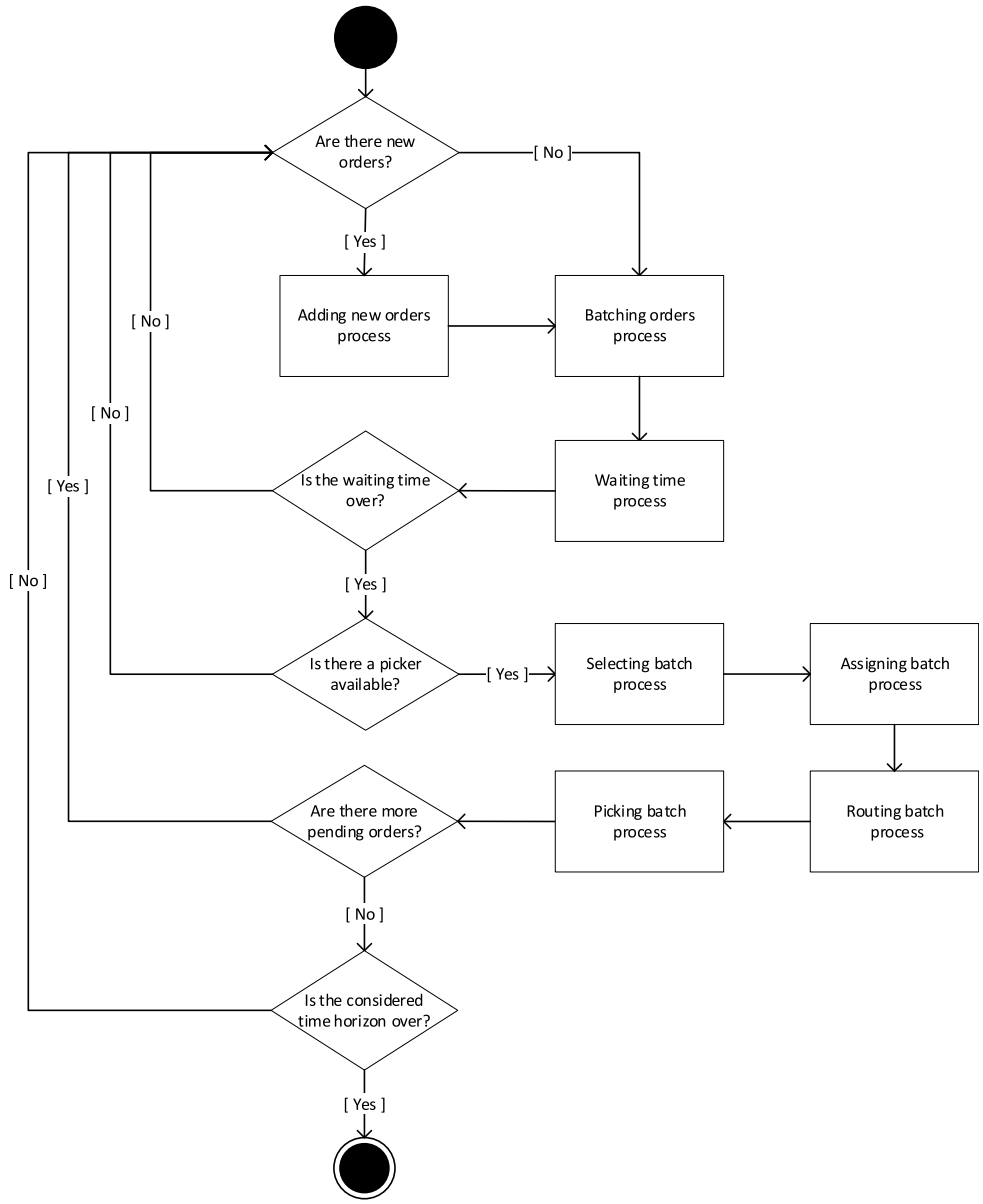


Fig. 1. Activity diagram of the processes involved in the OOBPMP.

in Fig. 1. In this figure we can observe the main processes involved in the OOBPMP in the time horizon considered: Adding, Batching, Waiting, Selecting, Assigning, Routing, and Picking (depicted as activities/rectangles in the diagram). Also, the diagram represents the decisions that need to be taken (depicted as rhombuses) and the possible answers considered (Yes/No). The process flow starts by checking if there are new orders arrived at the system waiting to be collected. Then, it tries to group the available orders in efficient batches, depending on the objective pursued. Once a solution has been conformed, if there is still available time for batching (usually denoted as "time window"), the process checks again if new orders have arrived at the system. If so, it incorporates them to the current solution. Otherwise, the best solution until that moment is moved ahead in the process. At this point, if there is at least one available picker, a batch is selected and assigned to the picker. Finally, a picking route is calculated, and the picking task starts. The process continues repeatedly until all arrived orders in the time horizon considered have been processed. In this research, we are interested in the general behavior of the algorithms for the OOBPMP. Particularly, we focus on the batching task by proposing, in Section 4.2, a new batching algorithm for the problem. For the rest of the activities, we describe the strategy followed.

3.1. Objective functions

Among the different objective functions previously introduced in the literature, we focus our attention on three of them:

- **Minimization of the picking time**: the objective is to minimize the sum of the picking time spent by each picker in the warehouse in the duty of collecting a batch. Each tour of a picker in the warehouse has a picking time associated, which is calculated as the sum of: the setup time (time needed by the picker to get ready for starting a new route); the routing time (time needed by the picker to traverse the warehouse to reach each picking position); and the extraction time (time needed by the picker to decelerate/accelerate the picking cart and to extract the items from their picking locations). Among others, this objective function has been considered in Chen et al. (2018), Rubrico et al. (2011) and Gil-Borrás et al. (2020b).
- **Minimization of the completion time**: the objective is to minimize the total time needed to collect all items from a set of orders received in a warehouse. This time can also be described as the

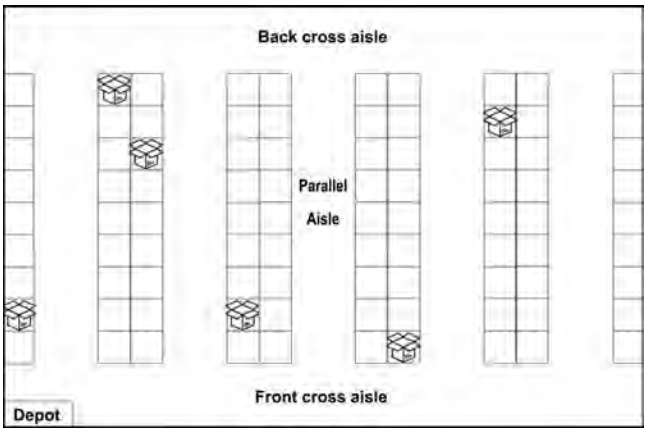


Fig. 2. Layout of the warehouse studied in this paper, with two cross aisles and a variable number of parallel aisles.

time elapsed between the moment when the picking starts and the moment when the last order is handed. Among others, this objective function has been considered in Henn (2012) and Zhang et al. (2017).

- **Minimization of the differences in the workload**: in the case of multiple pickers, a common objective function is also to minimize the differences among the workload of the pickers in any sense: number of orders processed, distance traversed, number of items retrieved, total time retrieving items, etc. In this case, we look for the minimization of the maximum difference between the picking time spent by any picker and the average picking time of all pickers. Among others, this objective function has been considered in Menéndez, Pardo, Sánchez-Oro, and Duarte (2017) and Zhang et al. (2017).

Finally, it is important to notice that a solution for the OOBPMP consists of the assignation of the orders received in the warehouse to batches, the assignation of each batch to a picker and the determination of the moment of time when each picker should start a new route.

3.2. Type of instances/warehouse description

The warehouse structure studied in this paper is the one depicted in Fig. 2. This structure is the most classical one used in the context of order batching problems. It presents a rectangular shape with only one block of parallel aisles and two crossing aisles (one at the back and one at front of the warehouse). Each parallel aisle contains several picking positions at each side of the aisle. In the example in Fig. 2, the warehouse has five parallel aisles formed by shelves with 18 picking positions considering the shelves at both sides of the aisle. The depot is the place where pickers start and finish their routes, and it is placed at the front cross aisle (either at the leftmost corner or at the center of the aisle).

We consider that only one product is stored per picking position. Also, a product can only be stored in one picking position along the whole warehouse and this position does not change through the time. Products can be stored at random or following the well-known ABC distribution (i.e., the most demanded products are placed closer to the depot). No stock limitations are considered for the products.

3.3. Mathematical formulation

In this section, we present a Mixed Integer Non-Linear Mathematical Model for the OOBPMP. It is important to notice that it is not possible to mathematically formulate and solve a problem if all input information is not known beforehand, as it is the case of online problems. However, in this case, the online version of the problem is equal to the offline version if we observe a particular instant in time, which considers only the orders which are currently in the system. Previous formulations for offline variants of the problem are available in the literature. Particularly, Gil-Borrás et al. (2020b), Henn (2012) studied the offline model of the OBP with a single picker, while Zhang et al. (2017) studied the offline model of the OBP with multiple pickers. The ideas presented here are based on those formulations.

A solution in the context of the OOBPMP consists of the assignation of the available orders to batches, and the assignation of each batch to a picker. Then, the evaluation of a solution requires the use of a routing algorithm, which determines the route necessary to pick all items in a batch. Therefore, we first present the formalization of the routing algorithm used in this paper (S-Shape routing algorithm, Hall (1993)). Particularly, in the context of this problem, we are only interested in the time that the picker needs to collect the items in the batch.

The S-Shape routing algorithm was previously formalized in Öncan (2015) and Zhang et al. (2017). First, in Table 1 we compile the parameters and variables needed to define the algorithm, while in Table 2, we compile the rest of the parameters and variables needed to define the OOBPMP.

For the sake of simplicity, the variable dis_j contains the traveled distance to collect all orders in batch j. Notice that the traveled distance is the sum of the distance traveled through the cross aisles (D_j^H) and the distance traveled through the parallel aisles (D_j^V):

$$dis_j = D_j^H + D_j^V \tag{1}$$

with D_j^H and D_j^V calculated as follows:

$$D_j^H = |A_j^L - A^D| \cdot C + |A_j^L - A_j^R| \cdot C + |A_j^R - A^D| \cdot C. \tag{2}$$

$$D_j^V = \begin{cases} A_j L, & odd_j = 0 \\ (A_j - 1)L + 2D_j^F, & odd_j = 1. \end{cases} \tag{3}$$

and the variables needed to compute these distances are calculated as follows:

$$z_{jq} = p_{iq} \cdot x_{ji}, \quad \forall\, j \in \{1, \dots, m\}, \forall\, i \in \{1, \dots, n\}, \forall\, q \in \{1, \dots, Q\}. \tag{4}$$

$$A_j = \sum_{q=1}^{Q} z_{jq}, \quad \forall\, j \in \{1, \dots, m\}. \tag{5}$$

$$odd_j = A_j \mod 2, \quad \forall\, j \in \{1, \dots, m\}. \tag{6}$$

$$A_j^L = \min_{q \in \{1,\dots,Q\}} q \cdot z_{jq} : (q \cdot z_{jq} > 0), \quad \forall\, j \in \{1, \dots, m\}. \tag{7}$$

$$A_j^R = \max_{q \in \{1,\dots,Q\}} q \cdot z_{jq}, \quad \forall\, j \in \{1, \dots, m\}. \tag{8}$$

$$D_j^F = \max_{i \in \{1,\dots,n\}} last_{i,A_j^R} \cdot x_{ji}, \quad \forall\, j \in \{1, \dots, m\}. \tag{9}$$

Once we have defined the distance needed to collect the batch j, we can define the function which calculates the routing time as follows:

$$T_j^{routing} = \frac{dis_j}{v_{routing}}, \quad \forall\, j \in \{1, \dots, m\}. \tag{10}$$

where $v_{routing}$ is a parameter which represents the velocity of the picker. This parameter is compiled in Table 2 together with the rest of the parameters needed to evaluate a solution.

As it was previously introduced in Henn (2012) and Gil-Borrás et al. (2020b), the time needed to collect the batch not only depends on the routing time, but also on the setup time and the extraction time. Let us denote as $T_j^{picking}$ as the time needed to collect the items in the batch j. More formally,

$$T_j^{picking} = T_j^{routing} + T_j^{extraction} + t_{setup}, \quad \forall\, j \in \{1, \dots, m\}. \tag{11}$$

Considering that w_i is the number of items in the order i and $v_{extraction}$ is the number of items that the picker is able to search and

Table 1
Parameters and variables needed for the definition of the S-Shape routing algorithm.

Parameters		
C	→	Center-to-center distance between two parallel aisles.
L	→	Length of a parallel aisle.
A^D	→	Aisle placed in front of the depot.
Q	→	Number of parallel aisles in the warehouse.
$last_{iq}$	→	Distance from the front cross aisle to the furthest article placed at aisle q, demanded in the order i.
p_{iq}	→	$\begin{cases} 1, & \text{if order } i \text{ has an item in the aisle } q, \\ 0, & \text{otherwise.} \end{cases}$
Variables		
A_j	→	Number of aisles that contain at least one pick location in batch j.
A_j^L	→	Leftmost aisle that contains at least one pick location in batch j.
A_j^R	→	Rightmost aisle that contains at least one pick location in batch j.
D_j^F	→	Given the rightmost aisle with an item in batch j, it measures the distance from the farthest item to collect to the front aisle.
D_j^H	→	Distance traveled through the cross aisles to collect batch j.
D_j^V	→	Distance traveled through the parallel aisles to collect batch j.
odd_j	→	$\begin{cases} 1, & \text{if } A_j \text{ is odd,} \\ 0, & \text{otherwise.} \end{cases}$
z_{jq}	→	$\begin{cases} 1, & \text{if batch } j \text{ has an item in aisle } q, \\ 0, & \text{otherwise.} \end{cases}$

Table 2
Parameters and variables for the OOBPMP.

Parameters		
n	→	Number of orders at the system in a particular time instant.
m	→	Upper bound of the number of batches (a straightforward value is $m = n$).
l	→	Number of pickers.
$v_{routing}$	→	Number of units of length that the picker can traverse in the warehouse per unit of time (velocity).
$v_{extraction}$	→	Number of items that a picker can search and pick per unit of time.
t_{setup}	→	Constant time needed by the picker to process each batch.
w_i	→	Number of items in order o_i $(1 \le i \le n)$.
W	→	Maximum capacity of a batch.
ar_i	→	Arrival time of order i.
Variables		
st_j	→	Start time of batch j.
x_{ji}	→	$\begin{cases} 1, & \text{if order } i \text{ is assigned to batch } j, \\ 0, & \text{otherwise.} \end{cases}$
y_{jk}	→	$\begin{cases} 1, & \text{if picker } k \text{ is assigned to batch } j, \\ 0, & \text{otherwise.} \end{cases}$

pick per unit of time, the extraction time for a batch j can by defined as follows:

$$T_j^{extraction} = \sum_{i=1}^{n} \frac{w_i \cdot x_{ji}}{v_{extraction}}, \quad \forall j \in \{1, \dots, m\}. \tag{12}$$

Finally, t_{setup}, is a parameter which represents the additional time needed by the picker to process the batch (either before starting the picking and/or after finishing it).

Once the model of the routing algorithm has been formalized, we introduce the models of the three studied problems, whose objective functions are defined next.

The objective function in (13) minimizes the total picking time needed to collect all orders arrived to the system. Notice that it avoids waiting times.

$$\min \sum_{j=1}^{m} T_j^{picking}. \tag{13}$$

The objective function in (14) minimizes the completion time, which is determined by the moment in the time when the picker delivers the last batch.

$$\min \max_{j \in \{1,\dots,m\}} \left(st_j + T_j^{picking} \right). \tag{14}$$

Finally, the objective function in (15) minimizes the maximum difference between the picking time used by the picker which works the most and the average picking time of all pickers in the system.

$$\min \max_{k \in \{1,\dots,l\}} \sum_{j=1}^{m} y_{jk} \cdot T_j^{picking} - \sum_{j=1}^{m} \frac{T_j^{picking}}{m}. \tag{15}$$

Next, we define the constraints that must be satisfied by any of the studied optimization problems.

Constraints in (16) guarantee that each order is assigned only to one batch:

$$\sum_{j=1}^{m} x_{ji} = 1, \quad \forall i \in \{1, \dots, n\}. \tag{16}$$

Constraints in (17) guarantee that each batch is assigned only to one picker:

$$\sum_{k=1}^{l} y_{jk} = 1, \quad \forall j \in \{1, \dots, m\}. \tag{17}$$

Constraints in (18) guarantee that the maximum capacity of a batch is not exceeded:

$$\sum_{i=1}^{n} w_i \cdot x_{ji} \le W, \quad \forall j \in \{1, \dots, m\}. \tag{18}$$

Constraints in (19) guarantee that the batch j starts to be collected only if there is at least one picker available:

$$st_j \geq \min_{k\in\{1,\dots,l\}} \max_{s\in\{1,\dots,j-1\}} y_{sk} \cdot \left(st_s + T_s^{picking}\right), \quad \forall\, j \in \{2,\dots,m\}. \tag{19}$$

Constraints in (20) guarantee that the batch j starts to be collected, once the collection of the batch $j-1$ has already begun:

$$st_j \geq st_{j-1}, \quad \forall\, j \in \{2,\dots,m\}. \tag{20}$$

Constraints in (21) guarantee that the collection of batch j do not start before the orders assigned to that batch have arrived to the system:

$$st_j \geq ar_i \cdot x_{ji}, \quad \forall\, i \in \{1,\dots,n\},\ \forall\, j \in \{1,\dots,m\}. \tag{21}$$

Constraints in (22) state that st_j cannot be negative:

$$st_j \geq 0, \quad \forall j \in \{1,\dots,m\}. \tag{22}$$

Finally, constraints in (23) state that the variables x_{ji} are binary:

$$x_{ji} \in \{0,1\}, \quad \forall\, j \in \{1,\dots,m\},\ \forall\, i \in \{1,\dots,n\}. \tag{23}$$

4. Algorithmic proposal

In this section, we describe our algorithmic proposal to tackle the OOBPMP. Particularly, we describe the strategy proposed for each of the aforementioned subproblems within the OOBPMP. Notice that in this paper we focus our attention on the batching task, while we use several well-known/straight-forward strategies to handle the rest of the subproblems. Initially, it is necessary to introduce a method in charge of the synchronization of all involved processes (see Section 4.1) that happen in the picking process during the considered time horizon.

4.1. General schema of the MS-VND

In this section, we describe a general orchestration method which organizes all processes involved in the problem and its associated relationships. This method is in charge of the simulation of the picking process and determines the different aspects involved, such as: the arrival of orders, the batching operation, the selection of the next batch to collect, the routing operation, the assignation of batches to the pickers, the picking of orders, and the update of the associated data structures. However, since the main contribution of this paper is focused on the batching task, we have used the strategies used for this task (multistart procedure and VND) to denote the algorithm as MS-VND.

In Algorithm 1 we introduce the proposed orchestration method MS-VND. The algorithm receives the list of orders ($ordList$) to collect as an input parameter. Notice that, in a real scenario, orders arrive online, so it means that not all orders are available at the beginning of the process. However, to illustrate the online behavior, we consider that the input list of orders contains not only the orders but also the moment in time when those orders arrive to the system within the considered time horizon. It is also important to remark that the schedule of the arrival of orders is calculated following a Poisson Point process distribution, as we will describe in Section 5, which will scatter the arrival of the orders within the considered time horizon, approximately.

The method is continuously running until all orders have been collected. It starts by initializing several data structures: $pendingOrd$ contains the orders which have already arrived to the system but that have not been collected yet; $partialSol$, contains a partial temporary solution (i.e., batches conformed with the orders in $pendingOrd$) calculated by the `batchingAlgorithm` in the current iteration; $bestPartialSol$ contains the best $partialSol$ found with the orders in $pendingOrd$. Notice that new partial solutions are being calculated (steps 6–11) until the `waitingAlgorithm` determines that at least a picker can start the picking operation of the next available batch; $finalSol$ contains an ordered list of batches with the orders already collected.

Once all variables have been initialized, the orchestration method in Algorithm 1 updates the list of $pendingOrd$ (step 8) and calculates a partial solution with the batching algorithm (step 9). Then, it updates (if necessary) the $bestPartialSol$ found with the current $partialSol$ obtained (step 10). The update consists of replacing the current $bestPartialSol$ with $partialSol$. Notice that this update is produced when either $partialSol$ is better than $bestPartialSol$ in terms of quality, or when $partialSol$ contains new orders arrived in the system. This process is repeated while the $bestPartialSol$ is empty (i.e., there are not orders in the system awaiting to be collected) or if there exists a waiting strategy which determines that the picker should wait for the arrival of new orders, before starting the picking operation (step 11).

Once there is at least one picker is available (step 13), the `selectingAlgorithm` chooses a batch within the $bestPartialSol$ (step 14) and the `routingAlgorithm` calculates the route to follow for collecting the orders in the selected batch (step 15). In step 16, the `assigningAlgorithm` determines if the selected $batch$ and its associated $route$ can be assigned to the selected available $picker$. Notice that depending on the objective pursued, the assignation might not be straight forward (i.e., an available picker can be the one with the larger workload and the method might determine to wait until another picker becomes available). If the assignation is performed, the picking operation starts (step 17), the collected orders are removed from $bestPartialSol$ and from the list of $pendingOrd$ (step 18). Finally, the collected $batch$ is added to the $finalSol$ (step 19). The whole process is repeated until the list of input orders becomes empty and there are not orders pending to be collected in the system.

Algorithm 1 Orchestration method MS-VND

```
1: function MS-VND(ordList)
2:     pendingOrd ← ∅
3:     partialSol ← ∅
4:     bestPartialSol ← ∅
5:     finalSol ← ∅
6:     do
7:         do
8:             pendingOrd ← pendingOrd ∪ getOrdersArrived(ordList)
9:             partialSol ← batchingAlgorithm(pendingOrd)
10:            bestPartialSol ← update(bestPartialSol, partialSol)
11:        while waitingAlgorithm(pendingOrd)||(bestPartialSol = ∅)
12:        for each picker ∈ getAvailablePickers() do
13:            batch ← selectingAlgorithm(bestPartialSol)
14:            route ← routingAlgorithm(batch)
15:            if assigningAlgorithm(picker, batch, route) then
16:                collect(picker, batch, route)
17:                remove(bestPartialSol, pendingOrd, batch)
18:                add(finalSol, batch)
19:            end if
20:        end for
21:    while (ordList ≠ ∅) || (pendingOrd ≠ ∅)
22:    return finalSol
23: end function
```

There are five remarkable methods within the MS-VND procedure presented in Algorithm 1. The batching algorithm (`batchingAlgorithm`), which conforms the batches to be collected, is described in Section 4.2. The waiting algorithm (`waitingAlgorithm`), which determines the time window available to update the current partial solution, is described in Section 4.3. Notice that this algorithm determines whether a picker can start the picking or must wait to improve the current partial solution. The selection algorithm (`selectingAlgorithm`), which determines the next batch of the partial solution to be assigned to an available picker, is described in Section 4.4. The assigning algorithm (`assigningAlgorithm`), which determines if the selected batch is assigned to an appropriate picker by following a workload balance criterion, is described in Section 4.5. Finally, the routing algorithm (`routingAlgorithm`), which determines the route that a picker must follow to collect a batch, is described in Section 4.6.

4.2. Batching algorithm

In this paper, we focus our attention on the batching algorithm as one of the key procedures in the context of any variant of the OOBP. The batching procedure consists of grouping all orders received at the warehouse in clusters, named batches. Each batch has a maximum predefined capacity, and all orders in the same batch are collected together. We propose a batching algorithm (which is introduced in Algorithm 2) based on the combination of a constructive procedure (step 2 in Algorithm 2), which is in charge of constructing feasible solutions for the OOBPMP, and an improvement procedure (step 2 in Algorithm 2) which is in charge of reaching a local optimum within the neighborhood of the solution previously constructed. Notice that this procedure is being continuously run following a multistart strategy (steps 7 to 11 in Algorithm 1) while the stopping criterion is not met. In each iteration of those steps, a single call to the batching algorithm (Algorithm 2) is performed, considering a single partial solution at the same time. This solution might contain one or more batches.

Algorithm 2 Batching Algorithm

1: **function** BATCHINGALGORITHM($pendingOrd$)
2: $initialSolution \leftarrow$ Constructive($pendingOrd$)
3: $improvedSolution \leftarrow$ VND($initialSolution$)
4: **return** $improvedSolution$
5: **end function**

The constructive method is inspired by the ideas presented within the Greedy Randomized Adaptive Search Procedure (GRASP) methodology (Feo & Resende, 1995). Particularly, it uses a greedy function to build a feasible solution step by step, combining its greediness with the randomization of several decisions. The pseudocode of the constructive method is presented in Algorithm 3. The method starts from an empty solution (step 2) and it adds a new order to the solution in each iteration (steps 5 to 11). The selection of the order to be added in each iteration is based in two principles: a greedy function which helps to select a set of promising candidate orders to be added to the solution, and a random function which selects one order within the list of promising candidate orders. Initially, all available orders are inserted in the Candidate List (CL) (step 3) and a random value α (step 4) is calculated. The greedy function ($\mathtt{f}$) evaluates the orders in the CL by measuring the weight of each order in such a way that all candidate orders are sorted in a descending way based on that weight. A percentage of the heaviest orders in the CL are included in a Restricted Candidate List (RCL), at step 7, by using the threshold th previously calculated in the step 6. The value of th is based on the maximum ($\arg\max \mathtt{f}(CL)$) and minimum ($\arg\min \mathtt{f}(CL)$) weight of any order in the CL, and the previous random value $\alpha \in U[0,1]$. Those orders with a weight over the threshold are inserted in the RCL. Finally, an order is selected at random from the RCL (step 8), added to the solution (step 9) and removed from the CL (step 10).

This procedure determines the sequence in which the orders will be added to the solution, however it is also necessary to determine the receiver batch for each particular order. Specifically, the algorithm starts with an empty batch and it adds the current selected order to the first batch with enough available space to handle the order. If none of the previously created batches have enough space, then a new empty batch is added to the solution.

The improvement phase of this multistart algorithm is based on a metaheuristic procedure. Particularly, instead of using a simple local search procedure, we propose the use of a Variable Neighborhood Descent (VND) method. Therefore, in each iteration, the method constructs an efficient solution, and this solution is further improved with a VND procedure. This schema is repeated until the method runs out of time, or the maximum number of iterations is reached.

The Variable Neighborhood Descent is a variant of the well-known Variable Neighborhood Search (VNS) methodology which was proposed by Mladenović and Hansen (1997) with the main idea of changing the neighborhood structure during the search to reach different

Algorithm 3 Constructive procedure

1: **function** CONSTRUCTIVE($orders$)
2: $solution \leftarrow \emptyset$
3: $CL \leftarrow orders$
4: $\alpha \leftarrow$ randomValue()
5: **while** $CL \neq \emptyset$ **do**
6: $th \leftarrow \arg\max \mathtt{f}(CL) - \alpha(\arg\max \mathtt{f}(CL) - \arg\min \mathtt{f}(CL))$
7: $RCL \leftarrow$ buildRCL(th, CL)
8: $order \leftarrow$ randomSelection(RLC)
9: add($solution, order$)
10: $CL \leftarrow CL \setminus \{order\}$
11: **end while**
12: **return** $solution$
13: **end function**

local optima. VNS has been used as the main procedure to solve many optimization problems. The original idea has been extended with many different variants. The classical ones are: Reduced VNS (RVNS), Variable Neighborhood Descent (VND), Basic VNS (BVNS), General VNS (GVNS), Skewed VNS (SVNS), and Variable Neighborhood Decomposition Search (VNDS) (Hansen & Mladenović, 2001; Hansen, Mladenović, & Moreno-Pérez, 2010; Mladenović & Hansen, 1997). Other recent approaches include: Parallel Variable Neighborhood Search (PVNS) (Duarte, Pantrigo, Pardo, & Sánchez-Oro, 2016; Menéndez, Pardo, Sánchez-Oro et al., 2017), Variable Formulation Search (VFS) (Pardo, Mladenović, Pantrigo, & Duarte, 2013), and Multi-Objective Variable Neighborhood Search (MO-VNS) (Duarte, Pantrigo, Pardo, & Mladenovic, 2015).

VNS methodology has been previously used in the context of order batching problems. On one hand, it has been used to tackle several offline variants of the problem. As far as we know, the first use of this methodology to tackle the OBP was proposed by Albareda-Sambola, Alonso-Ayuso, Molina, and De Blas (2009), where the authors minimized the picking time through the use of a VND. This objective function was also studied using BVNS and GVNS in Menéndez, Pardo et al. (2017), Menéndez, Pardo, Duarte, Alonso-Ayuso, and Molina (2015). Also in an offline context, the minimization of the tardiness was studied by Henn (2015) with VND and GVNS. This variant was again tackled by Menéndez, Bustillo et al. (2017) with a GVNS and, an extension of the problem with the same objective function, was also studied with a VND by Scholz et al. (2017). Finally, a Parallel VNS was presented in Menéndez, Pardo, Sánchez-Oro et al. (2017) for the minimization of the maximum picking time of a batch. On the other hand, VNS has been also used in an online context by Gil-Borrás et al. (2020b). In that paper, the authors proposed a VND to minimize the completion time and the maximum turnover time.

The key idea behind the VNS methodology is the definition and exploration of different neighborhood structures. These neighborhoods have been used for different tasks in the context of order batching problems: move orders among batches (Albareda-Sambola et al., 2009; Menéndez, Pardo et al., 2017), sort the batches to establish a sequence to collect them (Henn, 2015; Scholz et al., 2017), or assign orders/batches to pickers (Henn, 2015; Scholz et al., 2017). In this paper we focus on the task of batching orders. Therefore, we next review the neighborhood structures previously proposed to handle this task. The most common neighborhood structure used in the literature is the one defined by the swap operation, consisting of interchanging two orders belonging to different batches. This neighborhood has been previously studied in Albareda-Sambola et al. (2009), Gil-Borrás et al. (2020b), Henn (2015), Menéndez, Bustillo et al. (2017), Menéndez, Pardo et al. (2017), Menéndez et al. (2015), Menéndez, Pardo, Sánchez-Oro et al. (2017), Scholz et al. (2017). The second most studied neighborhood in the literature is the one defined by the insert operation, which consists of removing an order from its current batch

and assigning it to a new batch. This neighborhood structure has been previously used in: Albareda-Sambola et al. (2009), Gil-Borrás et al. (2020b), Henn (2015), Menéndez, Bustillo et al. (2017), Menéndez, Pardo et al. (2017), Menéndez et al. (2015), Scholz et al. (2017). The extension of this neighborhood structure, consisting of removing two orders from a batch and inserting them in another batch was presented in Albareda-Sambola et al. (2009). Similarly, another extension consists of removing two orders from a batch and interchanging them with one order from another batch. This neighborhood was used by Gil-Borrás et al. (2020b), Menéndez, Pardo et al. (2017). Other complex neighborhoods have also been explored in previous research, such as: two consecutive swap/insert operations in chain (Albareda-Sambola et al., 2009; Menéndez, Pardo et al., 2017; Menéndez, Pardo, Sánchez-Oro et al., 2017); the insertion of two orders from the same batch in two different batches (Albareda-Sambola et al., 2009); the insertion of two orders from different bathes into one batch (Albareda-Sambola et al., 2009).

The VND strategy proposed in this paper is designed to systematically explore three different neighborhood structures. The obtained result from the VND procedure is therefore a local optimum with respect to all three neighborhood structures considered.

In Algorithm 4 we present the pseudocode of the VND procedure used in this paper for the OOBPMP. The initial solution received by the VND as a starting point is calculated with the constructive procedure previously presented. This solution is explored by three local search procedures which follow a first improvement strategy. Each local search explores a different neighborhood and then determines if an improvement has been made (steps 13 to 18 in Algorithm 4) or not. If a local search procedure is not able to improve the current solution by exploring its neighborhood, then the method jumps to the next available neighborhood, otherwise it returns to the first local search procedure. The process is repeated until not further improvements are made with any of the local search methods considered.

Notice that the function used to evaluate the quality of the solution (`eval` in Algorithm 4) determines if a new solution visited can be considered as an improvement during the search. Therefore, different `eval` functions might guide the search to different areas of the solution space. In this paper we have explored two different `eval` functions: the workload balance and the picking time, obtaining two different search strategies that are compared in Section 5. However, to compute any of the objective functions considered, it is previously necessary to compute the picking route using a routing algorithm (see Section 4.6).

The VND in Algorithm 4 is configured with the following local search procedures:

- The `LocalSearchInsert1x0` is based on an insert neighborhood, which considers all possible solutions reached by the insertion of any order in the solution in all available batches.
- The `LocalSearchSwap2x1` is based on an interchange neighborhood, which considers all possible solutions reached by the exchange of every pair of two orders within the same batch, with any single order in any other batch.
- The `LocalSearchSwap1x1` is based on an interchange neighborhood which considers all possible solutions reached by the interchange of any pair of orders assigned to a different batch in the solution.

Notice that it is mandatory that the resulting batches after any move within the proposed local search procedures do not violate the maximum capacity restriction on a batch. Otherwise, the solution obtained is discarded.

Neighborhood structures are typically sorted depending on its size, which is usually related to the time needed to explore them, being the fastest to be explored considered firstly and the slowest considered at the end. However, this is an empirical rule that can be also tested when considering a specific set of neighborhoods for a particular optimization problem. In this case, the order of the local search procedures and its associated neighborhoods presented in Algorithm 4 has been empirically determined.

Algorithm 4 Variable Neighborhood Descent

```
1: function VND(solution)
2:     k ← 1
3:     k_max ← 3
4:     best ← solution
5:     repeat
6:         if k == 1 then
7:             solution' ← LocalSearchInsert1x0(best)
8:         else if k == 2 then
9:             solution' ← LocalSearchSwap2x1(best)
10:        else if k == 3 then
11:            solution' ← LocalSearchSwap1x1(best)
12:        end if
13:        if eval(solution') < eval(best) then
14:            best ← solution'
15:            k = 1
16:        else
17:            k = k + 1
18:        end if
19:    until k > k_max
20:    return best
21: end function
```

As a final remark to this section, we would like to remark the relevance of the strategies chosen to handle the batching task. First, the use of a multistart procedure is devoted to the online nature of the problem, since each construction in a multistart strategy can consider new orders arrived at the system, in addition to the ones available in the previous iteration. Among the different multistart algorithmic methodologies, GRASP was selected since it contributes to construct diverse solutions which make room in the batches constructed, due to the randomization nature of the procedure. This fact simplifies the task for local search procedures. Finally, as it was previously reviewed, VNS methodology resulted a very successful methodology in the past, when tackling other variants of the OBP. This fact has conducted researchers to very simple and successful ideas in the definition of neighborhood structures that can be easily adapted to the OOBPMP. Finally, GRASP and VNS have been widely combined in the past when tackling hard optimization problems by typically using VND as the local search phase of GRASP. This is the reason why VND is selected among the different variants of VNS.

4.3. Waiting algorithm

The waiting method is in charge of determining the time window that a picker must wait before starting a new route. Notice that, in some situations and depending on the objective function considered, it must be worthy to wait until more orders are available in the system before composing the batches. This is due to the fact that a larger number of orders might allow the batching algorithm to configure batches which retrieval is more efficient. In this paper, we follow the most naive waiting algorithm used in the literature of the OBP, which establishes that a picker can start its route as soon as it becomes available and there is at least one batch ready to be collected.

4.4. Selecting algorithm

Depending on the number of orders arrived to the system, the solution provided by the batching algorithm might have several batches. In this case, the selection method is in charge of determining which is the next batch that must be collected. The method proposed for this task

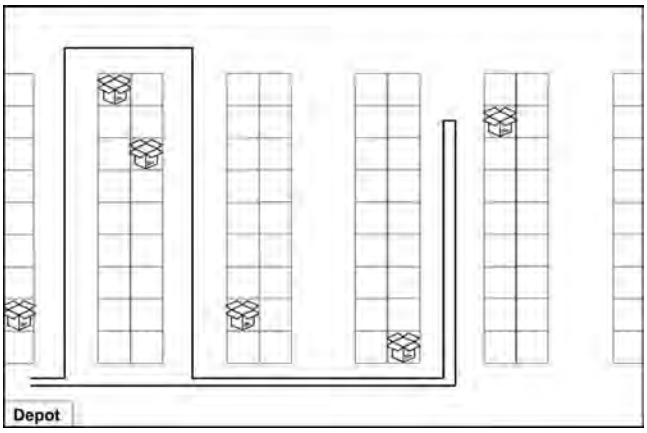


Fig. 3. Example of the route obtained with the S-Shape routing algorithm to collect the boxes in the figure.

sorts all batches in the current partial solution in a descending way with respect to its weight. Then, the method selects the heaviest batch as the next one to be collected. In the case that more than one batch achieves the largest weight, the tie is broken by selecting the batch which can be collected faster according to the routing method.

4.5. Assigning algorithm

In warehouses with multiple pickers, the assigning algorithm is in charge of determining who, among the pickers, will be assigned to the next waiting batch chosen by the selecting algorithm. This method is very relevant when the objective is to balance the workload among the pickers. In this case, we use the method proposed by Zhang et al. (2017) which assigns the next available batch to the picker that has traveled less until that moment.

4.6. Routing algorithm

Once there is a waiting batch assigned to an available picker, the picker must follow an efficient route within the warehouse to collect all orders in the assigned batch. This route is calculated by the routing algorithm. Routing algorithms for the OBP have been widely studied in the literature and it is possible to find exact and heuristic approaches. In this paper, we propose the use of the S-Shape routing algorithm (Hall, 1993) which has also been referred to as Traversal in the literature. This method has been widely used due to its simplicity for the pickers and its fast calculation process. In Fig. 3 we show an example of this algorithm. In this case, the depot is placed at the front cross aisle at the leftmost corner. The route starts and ends at the depot and all parallel aisles with at least an item to be collected are fully traversed except for the case when the number of aisles with items to collect is odd. In that situation, the last aisle is entered from the front cross aisle and the picker performs a U-turn when the last item is reached, to return to the front cross aisle, as this is the case of the example depicted in Fig. 3.

5. Computational results

The algorithmic proposals presented in this paper have been deeply evaluated over data sets of instances previously reported in the literature. In Section 5.1, we describe in detail the characteristics of the warehouses, orders, and arrival times which compose the instances used in our experiments.

To test our proposals, we have carried out a wide empirical study which can be divided in two main parts: on one hand, we have evaluated the performance of our methods, by considering three different objective functions: completion time, picking time, and workload balance, and comparing our results with the previous methods in the state of the art (see Sections 5.2, 5.3, and 5.4, respectively). For each objective function, we varied the number of pickers and the time horizon considered. All results were analyzed using statistical tests. Particularly, we used the Friedman Rank Test (Friedman, 1937) for comparing the results of multiple algorithms, and the Wilcoxon Test (Wilcoxon, 1992) for the comparison of pairs of algorithms. On the other hand, we have performed an additional analysis by studying the influence of the number of pickers (see Section 5.5) and the influence of the congestion in the arrival of orders (see Section 5.6), in the performance of the methods compared when handling the same scenario.

All methods, including the ones in the state of the art, were coded in Java 8 and run in an Intel (R) Core (TM) 2 Quad CPU Q6600 2.4 Ghz machine, with 4 GB DDR2 RAM memory. The operating system used was Ubuntu 18.04.1 64 bit LTS.

5.1. Instances

We have selected two widely used data sets of instances previously reported in the state of the art of different variants of the Order Batching family of problems. One data set was introduced by Albareda-Sambola et al. (2009) and the other one was presented by Henn (2012). The specifications and main characteristics of each data set are summarized in Table 3. The data sets were originally designed and used in the context of the OBP (the offline version of this problem) and used in the related literature (De Koster, Van der Poort, & Wolters, 1999; Koch & Wäscher, 2016; Menéndez, Bustillo et al., 2017; Menéndez, Pardo et al., 2017; Menéndez et al., 2015; Menéndez, Pardo, Sánchez-Oro et al., 2017; Scholz et al., 2017; Scholz & Wäscher, 2017). Later, those instances have been adapted for the online multipicker version of the OBP. In this sense it is necessary to determine two additional aspects: an arrival time for each order within the observed time horizon, and the number of pickers. Notice that these instances and the straightforward adaptation to the online multipicker context has also been used in previous research papers (Alipour et al., 2020; Gil-Borrás et al., 2020b; Pérez-Rodríguez, Hernández-Aguirre, & Jöns, 2015; Zhang et al., 2017; Žulj, Kramer, & Schneider, 2018).

Since the experiments in the context of the OOBPMP are performed in real time, the execution time of the study might be extremely large when considering many instances. To avoid this drawback, we have used the selection of a representative subset of instances proposed by Menéndez, Pardo et al. (2017) and later used in Gil-Borrás et al. (2020b), in order to have a reduced data set which can be handled in a reasonable amount of time. Particularly, the first reduced data set is composed by 80 instances, while the second reduced data set is composed by 64 instances. It is important to notice that the authors in Menéndez, Pardo et al. (2017) demonstrated that the selection of instances they made resulted in a representative data set, avoiding the necessity of using a larger number of instances from the same origin.

In addition to the characteristics of the warehouse, it is also necessary to take into consideration other parameters which affect the time needed to perform the picking operation. Particularly, in Table 4 we report: the travel speed of the pickers, once they have started their routes; the extraction speed of items, once the picker is placed at the picking position; and the batch setup time, which contains the necessary time to prepare the picking cart before starting the next route.

Finally, the arrival of orders to the warehouse in the context of the OOBPMP is distributed through a particular time horizon. In this sense, it is necessary to configure a simulation environment, which provides the orders to the system prior calculating the batch configuration and picking route. The time horizon of the arrival of orders to the warehouse has been set between 1 and 4 h, depending on the experiment.

To simulate the arrival time for each order, we follow a Poisson point process. Since the time horizon is set between 1 and 4 h (t =

Table 3
Warehouse characteristics of the data set introduced in Albareda-Sambola et al. (2009) and Henn (2012).

	Albareda-Sambola et al. (2009)				Henn (2012)
	W1	W2	W3	W4	W5
Storage policy	Random/ABC				Random/ABC
Depot position	Center/Left corner				Center
Order size	U(1,7)	U(2,10)	U(5,25)	U(1,36)	U(5,25)
Item weight	1	1	1	U(1,3)	1
Batch capacity (weight)	12	24	150	80	30 / 45 / 60 / 75
Number of parallel aisles	4	10	25	12	10
Number of items per aisle	2 × 30	2 × 20	2 × 25	2 × 16	2 × 45
Number of items	240	400	1250	384	900
Parallel aisle length	50 m	10 m	50 m	80 m	45 m
Center distance between two consecutive parallel aisles	4.3 m	2.4 m	5 m	15 m	5 m
Number of instances	20	20	20	20	64

Table 4
Configuration parameters proposed in Henn (2012) involved in the evaluation of the solutions.

Travel speed	48 LU/min
Extraction speed	6 items/min
Batch setup time	3 min

1,2,3,4). The number of events in the interval of length t is a Poisson random variable $X(t)$ with mean $E[X(t)] = \lambda * t$. The λ value is selected depending on the number of orders considered in the experiment. In this case, the λ values chosen for our experiments are compiled in Table 5. It is important to remark that the time horizon defined for the arrival of orders in each experiment, is also the time that all evaluated methods are running.

5.2. Empirical study of the completion time

In this section, we evaluate the performance of the algorithms when considering the minimization of the completion time, one of the most reported objective functions used in the state of the art of the OOBPMP. The completion time indicates the elapsed time since the start of the experiment until the moment in time when the last order, among the considered ones in the instances, has been collected and handled in the depot. This time includes the time needed for any activity in the warehouse: batching, waiting, selecting, assigning, and routing, and it also depends on the number of pickers simultaneously working.

In all experiments of this section, we compare the two previously proposed methods in the state of the art (Alipour et al., 2020; Zhang et al., 2017) with the Multistart VND (MS-VND) proposed in this paper. Notice that we have tested two variants of the MS-VND. Both methods are equal, but they use a different evaluation function to guide the search (i.e., the function used to determine which solution is more promising). The MS-VND-1 uses the workload balance function to compare two solutions, while the MS-VND-2 uses the picking time function. The rationale behind this strategy is that just varying the evaluation function which guides the search, lets the method to explore different areas of the space search.

First, we have evaluated the impact of the time horizon studied on the performance of the methods, by considering 2 h time horizon (see Table 6) and 4 h time horizon (see Table 7). For each considered time horizon, we have also studied the influence of varying the number of pickers (2, 3, 4, and 5) for each experiment. Each configuration has been executed over the 144 instances previously selected and the results per configuration have been reported separated by the data set and all together. For each configuration, we report the averaged values of the objective function (Avg. (s)) and the deviation to the best value obtained in the experiment (Dev. (%)). In each table, we have highlighted in bold type font the results obtained by the best algorithm.

We observe that MS-VND-1 is the best overall method in terms of completion time, since it achieves the smaller deviation to the best solution in the experiment in most of the studied scenarios. To confirm the relevance of the results found, we have performed a Friedman rank test (see the row "2 h" in Table 10). The obtained p-value of 0.000 indicates that the differences among the compared methods are significant with MS-VND-1 ranking in the first position. Also, we corroborated that there were significant differences when comparing only our best variant MS-VND-1 with any of the two methods from the state of the art in isolation. To that aim, we carried out two Wilcoxon tests. The obtained p-values of 0.000 in both comparisons also indicate that the differences between the compared methods are significant.

The results of the 4 h context presented in Table 7 are very similar to the 2 h context, resulting again the MS-VND-1, the best method among the compared ones. The significance of the results was also corroborated by the Friedman rank test (see row "4 h" in Table 10), and again confirmed by the Wilcoxon test when comparing MS-VND-1 with each of the previous proposals separately.

We have also studied the behavior of the compared methods with respect to the completion time, when fixing the number of pickers and varying the number of available hours for the arrival of orders. In Table 8 we report the results for 2 pickers and, in Table 9, for 5 pickers. Again, in both scenarios MS-VND-1 was the best compared method and the differences found with other methods were statistically significant (see rows "2 pickers" and "5 pickers" in Table 10).

Analyzing and summarizing our findings about the study of the completion time, first we would like to highlight the importance of the use of the completion time as objective function. This is because minimizing the completion time results in an overall benefit for the customers since it contributes to reduce the delivery time of products. Increasing the time horizon for the schedule of the arrival of orders results in an increase in the completion time. Also, an increase in the number of pickers reduces the completion time. This point is verified by all tested methods. However, increasing the number of pickers results more beneficial for the completion time, with an upper bound of having a picker available for each order arriving at the system, which clearly results in unacceptable operational costs. However, this explains why using the workload balance as the guiding function, which usually provides solutions with a larger number of batches, helps to minimize the completion time. Also, it is important to notice that in this paper we do not consider blocking situations, which would also deteriorate the solution when increasing the number of pickers.

5.3. Empirical study of the picking time

In this section, we have performed the empirical comparison between our proposals and the methods from the state of the art when considering the picking time as objective function. In this scenario, we have performed the same set of experiments described in Section 5.2, but reporting the picking time, instead of the completion time. The picking time indicates the sum of the times spent by the pickers in the picking task, avoiding any waiting time. Notice that the picking task includes the setup time (time needed for the pickers prior to starting a new route to set up the picking cart and to review the assigned route),

Table 5
Values of the Poisson parameter λ, used to distribute the arrival of different number of orders in a particular time horizon.

		#orders						
		40	60	80	100	150	200	250
Time horizon	**1 h**	0.667	1.000	1.334	1.667	2.500	3.334	4.167
	2 h	0.334	0.500	0.667	0.834	1.250	1.667	2.084
	3 h	0.223	0.334	0.445	0.556	0.834	1.112	1.389
	4 h	0.167	0.250	0.334	0.417	0.625	0.834	1.042

Table 6
Comparison of the completion time with the state-of-the-art methods over several 2 h scenarios with different number of pickers.

Instances	#pickers	Completion time (With 2 h)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**2**	32600	9.43%	35321	17.59%	30391	0.67%	**30251**	**0.47%**
	3	22775	8.87%	24872	18.10%	21300	0.77%	**21228**	**0.72%**
	4	18067	8.35%	19804	18.18%	16864	**0.68%**	**16857**	0.93%
	5	15377	7.76%	16802	16.35%	14437	**1.03%**	**14412**	1.08%
Henn (64)	**2**	15609	8.95%	15239	5.64%	**14386**	**0.43%**	14467	0.93%
	3	11416	7.45%	11105	4.41%	**10662**	**0.81%**	10692	1.06%
	4	9658	6.28%	9417	3.35%	**9169**	**1.04%**	9210	1.40%
	5	8885	5.99%	8636	2.77%	**8458**	**0.86%**	8483	1.15%
All (144)	**2**	25048	9.22%	26395	12.28%	23278	**0.56%**	**23236**	0.68%
	3	17727	8.24%	18754	12.01%	16572	**0.79%**	**16545**	0.87%
	4	14329	7.43%	15187	11.59%	**13444**	**0.84%**	13458	1.14%
	5	12492	6.90%	13172	10.24%	11780	**0.89%**	**11777**	1.05%

Table 7
Comparison of the completion time with the state-of-the-art methods over several 4 h scenarios with different number of pickers.

Instances	#pickers	Completion time (With 4 h)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**2**	34700	7.52%	36107	9.99%	32546	0.52%	**32460**	**0.43%**
	3	25905	5.92%	26784	7.82%	**24490**	**0.43%**	24496	0.55%
	4	22096	5.61%	22448	5.62%	**21001**	**0.61%**	21020	0.86%
	5	19987	5.19%	20266	5.02%	**19068**	**0.41%**	19123	0.86%
Henn (64)	**2**	18235	6.56%	17767	3.45%	**17164**	**0.43%**	17207	0.62%
	3	15968	5.41%	15401	1.48%	**15206**	**0.33%**	15278	0.77%
	4	15552	5.60%	14867	0.79%	**14803**	**0.38%**	14817	0.48%
	5	15441	5.60%	14691	0.35%	**14655**	**0.11%**	14687	0.32%
All (144)	**2**	27382	7.09%	27956	7.08%	25709	**0.48%**	**25681**	0.52%
	3	21489	5.70%	21725	5.00%	**20364**	**0.38%**	20399	0.65%
	4	19187	5.61%	19079	3.47%	**18246**	**0.51%**	18263	0.69%
	5	17967	5.37%	17789	2.94%	**17106**	**0.27%**	17151	0.62%

Table 8
Comparison of the completion time with the state-of-the-art methods over several 2 pickers scenarios, varying the number of hours for the arrival of orders.

Instances	#Hours	Completion time (With 2 Pickers)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**1**	32221	11.33%	34795	20.47%	29895	0.68%	**29751**	**0.57%**
	2	32600	9.43%	35321	17.59%	30391	0.67%	**30251**	**0.47%**
	3	33472	8.85%	35895	14.45%	31210	**0.57%**	**31166**	0.58%
	4	34700	7.52%	36107	9.99%	32546	0.52%	**32460**	**0.43%**
Henn (64)	**1**	15206	9.48%	15779	12.50%	**13931**	**0.19%**	14049	1.05%
	2	15609	8.95%	15239	5.64%	**14386**	**0.43%**	14467	0.93%
	3	16547	7.74%	16114	4.39%	15439	0.78%	**15419**	**0.49%**
	4	18235	6.56%	17767	3.45%	**17164**	**0.43%**	17207	0.62%
All (144)	**1**	24659	10.51%	26343	16.93%	22800	**0.46%**	**22772**	0.78%
	2	25048	9.22%	26395	12.28%	23278	**0.56%**	**23236**	0.68%
	3	25950	8.36%	27104	9.98%	24201	0.67%	**24168**	**0.54%**
	4	27382	7.09%	27956	7.08%	25709	**0.48%**	**25681**	0.52%

Table 9
Comparison of the completion time with the state-of-the-art methods over several 5 pickers scenarios, varying the number of hours for the arrival of orders.

Instances	#Hours	Completion time (With 5 Pickers)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**1**	14052	10.76%	15141	18.00%	13055	**1.11%**	**13040**	1.34%
	2	15377	7.64%	16802	16.21%	14437	**0.91%**	**14412**	0.96%
	3	17546	5.96%	18332	8.33%	16667	**0.82%**	**16631**	0.86%
	4	19987	5.19%	20266	5.02%	**19068**	**0.41%**	19123	0.86%
Henn (64)	**1**	7079	8.96%	7400	13.00%	**6534**	**0.85%**	6598	1.64%
	2	8885	5.99%	8636	2.77%	**8458**	**0.86%**	8483	1.15%
	3	11977	5.64%	11461	0.92%	11441	0.77%	**11422**	**0.62%**
	4	15441	5.60%	14691	0.35%	**14655**	**0.11%**	14687	0.32%
All (144)	**1**	10953	9.96%	11700	15.78%	**10156**	**0.99%**	10177	1.47%
	2	12492	6.90%	13172	10.24%	11780	**0.89%**	**11777**	1.05%
	3	15071	5.82%	15279	5.03%	14344	0.80%	**14316**	**0.75%**
	4	17967	5.37%	17789	2.94%	**17106**	**0.27%**	17151	0.62%

Table 10
Friedman rank test for different test scenarios, when studying the completion time.

	Friendman rank test — Completion time				
	Alipour et al. (2020)	Zhang et al. (2017)	MS-VND-1 (Workload Balance)	MS-VND-2 (Picking Time)	Sig. (p-value)
2 h	3.47	3.16	**1.66**	1.71	**0.000**
4 h	3.67	2.69	**1.79**	1.85	**0.000**
2 pickers	3.47	3.33	**1.56**	1.65	**0.000**
5 pickers	3.45	2.97	**1.75**	1.81	**0.000**

the traveling time (time needed by the pickers to reach each picking position), the time needed to accelerate or decelerate the picking cart, and the extraction time of the items from the shelves. Particularly, in Tables 11 and 12 we study the behavior of the algorithms over 2 and 4 h scenarios, respectively, varying the number of pickers (2, 3, 4, or 5). Then, in Tables 13 and 14 we fixed the number of pickers (2 or 5) and varied the number of available hours (1, 2, 3, or 4) for the arrival of orders.

In this case, we have also studied the statistical significance of the results obtained. In Table 15, we report the obtained rank values for the Friedman tests for each of the studied scenarios. In all cases, the p-values obtained were 0.000, which indicate significant differences among the methods. However, this time our best variant was MS-VND-2, which resulted ranked in the first position in two out of the four scenarios. On the other hand, the proposal by Alipour et al. (2020) resulted in the first position in the other two scenarios. Observing the scenarios where the MS-VND-2 performed better, we can conclude that MS-VND-2 is the best method when the warehouse presents a higher congestion of orders. This might be due to the existence of fewer pickers working at the studied moment or numerous orders currently in the system. On the other hand, the method in Alipour et al. (2020) performed better when dealing with scenarios with low congestion. The results between the two best variants of the experiments (MS-VND-2 and Alipour et al. (2020)) were also observed to be statistically significant (p-value = 0.000) in three out of the four studied scenarios (4 h, 2 pickers and 5 pickers) when compared with the Wilcoxon test. However, in the 2 h scenario, the obtained p-value = 0.061 indicates no significant differences between the methods.

Analyzing and summarizing our findings about the study of the picking time, the optimization of the picking time is usually related to the energy saving and the enlargement of the life of the machinery. The larger the number of pickers, the higher the picking time. The reduction in the picking time is mainly due to the existence of fewer batches with less empty space. This situation is more suitable when the number of pickers is small, so there is more time to complete the batches with new arrived orders. Therefore, introducing waiting times between each departure might result in a reduction of the picking time, since there are more full batches. The same observation can be derived from a different perspective, i.e., a larger congestion rate usually results in better picking times. Also, we noticed that the use of the same objective function being minimized (picking time) as the guiding function to determine the search direction is more beneficial than using the workload balance.

5.4. Empirical study of workload balance

In this section, we have performed the empirical comparison between our proposals and the methods from the state of the art when considering the workload balance as the objective function. In this scenario, we have performed the same set of experiments described in Sections 5.2 and 5.3, but reporting the workload balance. The workload balance indicates the difference between the maximum picking time needed by a picker to complete its assigned tasks, with respect to the average picking time reported by all pickers. Particularly, in Tables 16 and 17 we study the behavior of the algorithms over 2 and 4 h scenarios, respectively, varying the number of pickers (2, 3, 4, or 5). Then, in Tables 18 and 19 we fixed the number of pickers (2 or 5) and varied the number of available hours (1, 2, 3, or 4) for the arrival of orders.

In this case, we have also studied the statistical significance of the results obtained (see Table 20). In all cases, the p-values obtained indicate significant differences among the methods. This time our best variant (MS-VND-1) resulted ranked in the first position in all scenarios. The differences found in the results obtained when comparing MS-VND-1 and the best algorithm in the state of the art (Zhang et al., 2017) were also observed to be statistically significant when compared with the Wilcoxon test, with a p-value of 0.000 in all comparisons performed.

Analyzing and summarizing our findings about the study of the workload balance among different workers, we found that it is a key criterion for maintaining a safe and healthy work environment. This objective function has not been deeply studied in the literature and previous methods usually fail when dealing with it. As observed, the increase in the number of pickers also increases the hardness of finding a better workload balance. On the other hand, higher congestion rates increase the chance of finding a more balanced solution. Again, we observed that the use of the same objective function being minimized (workload balance) as the guiding function to determine the search direction is more beneficial than using the picking time.

Table 11
Comparison of the picking time with the state-of-the-art methods over several 2 h scenarios with different number of pickers.

Instances	#pickers	Picking time (With 2 h)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**2**	61755	4.15%	67020	15.50%	60284	5.29%	**59558**	**3.58%**
	3	62507	**2.88%**	68045	16.90%	62828	11.07%	**61400**	7.49%
	4	**63291**	**2.11%**	69259	19.01%	65323	16.74%	63613	12.64%
	5	**64166**	**1.58%**	70469	20.64%	68553	23.18%	66372	18.47%
Henn (64)	**2**	28635	2.95%	29585	6.59%	28231	2.68%	**27930**	**1.32%**
	3	**29407**	**1.35%**	31278	9.73%	30643	8.82%	29859	5.47%
	4	**30397**	**0.43%**	33684	14.36%	33827	16.24%	32560	11.23%
	5	**31554**	**0.17%**	36616	20.58%	37252	23.80%	35653	18.04%
All (144)	**2**	47035	3.62%	50382	11.54%	46038	4.13%	**45501**	**2.57%**
	3	47796	**2.20%**	51704	13.71%	48524	10.07%	**47382**	6.59%
	4	**48671**	**1.36%**	53448	16.94%	51325	16.52%	49812	12.01%
	5	**49672**	**0.95%**	55423	20.61%	54641	23.46%	52719	18.28%

Table 12
Comparison of the picking time with the state-of-the-art methods over several 4 h scenarios with different number of pickers.

Instances	#pickers	Picking time (With 4 h)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**2**	62887	**2.65%**	68410	18.95%	63497	14.06%	**62589**	11.51%
	3	**64648**	**1.32%**	72511	25.30%	69108	24.54%	67483	20.27%
	4	**66357**	**0.97%**	76255	31.53%	75349	35.08%	72766	29.37%
	5	**67975**	**0.99%**	80295	36.60%	81054	43.01%	77856	36.73%
Henn (64)	**2**	**29892**	**0.82%**	33046	14.75%	32497	14.40%	31884	11.53%
	3	**32097**	**0.10%**	38529	24.84%	39536	29.34%	37810	23.06%
	4	**33923**	**0.01%**	43621	33.92%	45425	39.84%	42880	32.22%
	5	**34934**	**0.00%**	47236	41.14%	49013	46.22%	46763	40.08%
All (144)	**2**	**48223**	**1.84%**	52693	17.08%	49719	14.21%	48943	11.52%
	3	**50181**	**0.78%**	57408	25.10%	55965	26.68%	54295	21.51%
	4	**51942**	**0.54%**	61751	32.59%	62049	37.20%	59483	30.64%
	5	**53290**	**0.55%**	65602	38.62%	66814	44.44%	64037	38.22%

Table 13
Comparison of the picking time with the state-of-the-art methods over several 2 pickers scenarios, varying the number of hours for the arrival of orders.

Instances	#Hours	Picking time (With 2 Pickers)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**1**	61628	5.30%	65206	12.28%	59418	1.94%	**58720**	**0.57%**
	2	61755	4.15%	67020	15.50%	60284	5.29%	**59558**	**3.58%**
	3	62201	**3.34%**	68234	17.86%	61534	9.49%	**60886**	7.56%
	4	62887	**2.65%**	68410	18.95%	63497	14.06%	**62589**	11.51%
Henn (64)	**1**	28474	3.97%	29941	8.60%	27570	1.18%	**27365**	**0.28%**
	2	28635	2.95%	29585	6.59%	28231	2.68%	**27930**	**1.32%**
	3	**29183**	**1.76%**	30893	9.38%	29966	7.44%	29496	5.20%
	4	**29892**	**0.82%**	33046	14.75%	32497	14.40%	31884	11.53%
All (144)	**1**	46893	4.71%	49533	10.64%	45263	1.61%	**44784**	**0.44%**
	2	47035	3.62%	50382	11.54%	46038	4.13%	**45501**	**2.57%**
	3	47526	**2.64%**	51638	14.09%	47504	8.58%	**46935**	6.51%
	4	48223	**1.84%**	52693	17.08%	49719	14.21%	**48943**	11.52%

5.5. Influence of the number of pickers

Our next experiment is devoted to observe the influence of the number of pickers on the performance of the algorithms compared over the three objective functions studied. In Fig. 4 we report the averaged Completion time of all methods for 2 and 4 h scenarios, when varying the number of pickers. Similarly, in Figs. 5 and 6 we report the results for the averaged picking time and the averaged workload balance for the same scenarios.

As expected, we can observe a similar influence of the number of pickers on the completion time (i.e., the larger the number of pickers, the shorter the completion time) for any of the compared methods. On the other hand, the performance of the methods in terms of picking time benefits from higher congestion (i.e., orders arrive in shorter times). This is due to the fact that a larger number of orders in the system lets the batching algorithms to conform batches, which retrieval times are shorter (i.e., the retrieval of each picking route is more efficient). Finally, when observing the workload balance, all

Table 14
Comparison of the picking time with the state-of-the-art methods over several 5 pickers scenarios, varying the number of hours for the arrival of orders.

Instances	#Hours	Picking time (With 5 Pickers)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**1**	62640	**2.72%**	69096	15.47%	63832	11.43%	**61723**	6.27%
	2	**64166**	**1.58%**	70469	20.64%	68553	23.18%	66372	18.47%
	3	**66219**	**1.35%**	75242	28.24%	75068	34.91%	72064	28.56%
	4	**67975**	**0.99%**	80295	36.60%	81054	43.01%	77856	36.73%
Henn (64)	**1**	**29290**	**1.03%**	31075	7.57%	30740	8.57%	29792	4.60%
	2	**31554**	**0.17%**	36616	20.58%	37252	23.80%	35653	18.04%
	3	**33690**	**0.05%**	42948	33.36%	44502	38.33%	42344	31.96%
	4	**34934**	**0.00%**	47236	41.14%	49013	46.22%	46763	40.08%
All (144)	**1**	**47818**	**1.97%**	52198	11.96%	49125	10.16%	47531	5.53%
	2	**49672**	**0.95%**	55423	20.61%	54641	23.46%	52719	18.28%
	3	**51762**	**0.77%**	60889	30.51%	61483	36.43%	58855	30.07%
	4	**53290**	**0.55%**	65602	38.62%	66814	44.44%	64037	38.22%

Table 15
Friedman rank test for different test scenarios, when studying the picking time.

	Friendman rank test — Picking time				
	Alipour et al. (2020)	Zhang et al. (2017)	MS-VND-1 (Workload Balance)	MS-VND-2 (Picking Time)	Sig. (p-value)
2 h	1.96	3.44	2.92	**1.68**	**0.000**
4 h	**1.45**	3.12	3.35	2.08	**0.000**
2 pickers	2.38	3.70	2.43	**1.49**	**0.000**
5 pickers	**1.51**	3.03	3.40	2.05	**0.000**

methods except the one introduced by Alipour et al. (2020) were able to maintain the maximum workload difference when increasing either the time or the number of pickers. However, the method by Alipour performed worse with a larger number of pickers. This is due to the fact that its algorithm assigns the next available batch to the first available picker, instead of balancing its assignment.

As a general conclusion, the number of pickers influences the objective functions studied in different ways. Particularly, we observed that increasing the number of pickers results in shorter completion times and lower workload for each picker, however, finding a balance among the work performed by each picker results more complicated and the overall picking time increases, since pickers collect fewer items in each picking tour. Also, we can conclude that less pickers than necessary might result in delays in the completion time, while an excessive number of pickers might result in more dead time in the activity, larger costs, and a deterioration in the picking time.

5.6. Influence of the congestion rate

This last experiment is devoted to observe the influence of congestion in the arrival of orders on the performance of the compared algorithms over the three objective functions studied. Particularly, the scenarios considered range from lower congestion rates (larger number of available pickers and/or larger arrival time horizons) to higher congestion rates (fewer pickers and/or shorter arrival time horizons).

In Fig. 7 we report the averaged completion time in several scenarios. Similarly, in Figs. 8 and 9 we report the results for the averaged picking time and the averaged workload balance for the same scenarios.

As expected, in Fig. 7 we can observe that larger time horizons imply larger completion times, since the arrival of orders is more scattered. Also, we can observe that the completion time, when two pickers are available (Fig. 7(a)) is much larger than the case with five pickers available (Fig. 7(b)). On the other hand, we observe that when only two pickers are available, the increase in the number of hours does not produce as large deterioration in the completion time as it is the case of the five pickers configuration. We can conclude that the benefit of using these algorithms with low congestion rates, in terms of completion time, is more limited than using them in scenarios with larger congestion rates.

As far as the picking time is concerned, we observe in Fig. 8, that a larger number of pickers in the same time horizon implies worse picking times. This is due to the fact that the batches conformed are less compact (i.e., they have more available space in the batch), and therefore there are more batches. Consequently, a larger number of batches implies more picking routes and therefore larger picking times. The same effect can be observed when focusing on scenarios with the same number of pickers, but increasing the time horizon arrival. Larger time for the arrival of orders implies a larger number of batches and consequently larger picking times. Additionally, among the compared algorithms, we can highlight that the approach by Alipour performs better in very low congestion rate scenarios (5 pickers and 4 h time horizon). Again, we can conclude that the benefit of using these algorithms with low congestion rates, in terms of picking time, is more limited than using them in scenarios with larger congestion rates.

Finally, considering the workload balance reported in Fig. 9, we observe that a larger number of pickers difficult an egalitarian distribution of the work. On the other hand, increasing the number of hours for the arrival of orders does not seem to influence the balance of the workload, except for the method of Alipour in the 5 pickers scenario. We can conclude that the behavior of the proposed algorithms, in terms of workload balance, is very similar either with low congestion rates or with larger congestion rates.

To summarize our findings, we observed in our experiments that the guiding function has a large impact on the obtained results. Particularly, in this paper, we studied the optimization of three different objective functions in the context of the OOBPMP: the workload balance, the picking time, and the completion time. To handle the previous task, we proposed two methods: MS-VND-1 and MS-VND-2. The MS-VND-1 uses the workload balance function to compare two solutions and therefore to guide the search, while the MS-VND-2 uses the picking time function for the same task. As it is intuitively expected, using the objective function being minimized as a guiding function results in a better performance (i.e., MS-VND-1 performed better when minimizing the workload balance, while MS-VND-2 performed better when minimizing the picking time). Finally, when minimizing the completion time, we observed that the strategy of using the workload balance as the guiding function performed better than using the picking time. This is explained due to the differences in the structure of the solutions obtained with the two different guiding functions. Solutions obtained with MS-VND-2 presented fewer number of batches (usually with less empty space) than the solutions obtained with MS-VND-1.

In brief, MS-VND-1 is the best method overall when considering any of the objective functions, however it is closely followed by the MS-VND-2. The only exception to the previous assertion occurs in the

Table 16
Comparison of the workload balance with the state-of-the-art methods over several 2 h scenarios with different number of pickers.

Instances	#pickers	Workload balance (With 2 h)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**2**	448	2845%	387	2446.63%	**15**	**0.00%**	284	1769.37%
	3	798	2112%	621	1620.24%	**36**	**0.00%**	518	1333.94%
	4	1110	810%	718	489.00%	**122**	**0.00%**	610	399.91%
	5	1413	486%	671	178.27%	**241**	**0.00%**	611	153.63%
Henn (64)	**2**	296	686%	246	553.70%	**38**	**0.00%**	282	648.57%
	3	570	324%	393	192.23%	**134**	**0.00%**	410	204.90%
	4	935	271%	480	90.55%	**252**	**0.00%**	497	97.35%
	5	1257	274%	533	58.51%	**336**	**0.00%**	502	49.35%
All (144)	**2**	380	1410%	325	1188.04%	**25**	**0.00%**	283	1024.16%
	3	697	773%	519	551.10%	**80**	**0.00%**	470	488.64%
	4	1032	474%	613	240.73%	**180**	**0.00%**	560	211.39%
	5	1344	374%	609	115.12%	**283**	**0.00%**	563	98.64%

Table 17
Comparison of the workload balance with the state-of-the-art methods over several 4 h scenarios with different number of pickers.

Instances	#pickers	Workload balance (With 4 h)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**2**	691	2124.84%	445	1334.25%	**31**	**0.00%**	325	947.84%
	3	1438	1003.44%	572	338.73%	**130**	**0.00%**	496	280.83%
	4	2430	1041.98%	644	202.61%	**213**	**0.00%**	616	189.60%
	5	3230	926.22%	698	121.69%	**315**	**0.00%**	653	107.59%
Henn (64)	**2**	507	439.67%	286	204.78%	**94**	**0.00%**	266	182.99%
	3	1748	597.61%	382	52.40%	**251**	**0.00%**	400	59.66%
	4	3001	783.68%	430	26.73%	**340**	**0.00%**	414	21.88%
	5	4226	1006.96%	440	15.16%	**382**	**0.00%**	457	19.76%
All (144)	**2**	609	932.33%	375	534.98%	**59**	**0.00%**	299	406.59%
	3	1576	757.56%	487	165.25%	**184**	**0.00%**	454	146.83%
	4	2684	897.14%	549	103.99%	**269**	**0.00%**	526	95.55%
	5	3673	965.98%	583	69.22%	**345**	**0.00%**	566	64.34%

Table 18
Comparison of the workload balance with the state-of-the-art methods over several 2 pickers scenarios, varying the number of hours for the arrival of orders.

Instances	#Hours	Workload balance (With 2 Pickers)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**1**	444	3575.43%	404	3248.44%	**12**	**0.00%**	311	2473.98%
	2	448	2844.79%	387	2446.63%	**15**	**0.00%**	284	1769.37%
	3	502	1715.66%	379	1271.05%	**28**	**0.00%**	303	998.54%
	4	691	2124.84%	445	1334.25%	**31**	**0.00%**	325	947.84%
Henn (64)	**1**	283	782.75%	292	813.62%	**32**	**0.00%**	267	733.62%
	2	296	686.46%	246	553.70%	**38**	**0.00%**	282	648.57%
	3	422	378.83%	245	177.86%	**88**	**0.00%**	202	129.05%
	4	507	439.67%	286	204.78%	**94**	**0.00%**	266	182.99%
All (144)	**1**	372	1677.74%	355	1593.93%	**21**	**0.00%**	291	1291.37%
	2	380	1409.74%	325	1188.04%	**25**	**0.00%**	283	1024.16%
	3	466	754.91%	319	485.40%	**55**	**0.00%**	258	373.66%
	4	609	932.33%	375	534.98%	**59**	**0.00%**	299	406.59%

picking time with low congestion rates, where the method proposed by Alipour et al. (2020) performs better than our approaches. These conclusions can be easily observed in Figs. 4 to 9, where lower values indicate a better performance, with MS-VND-1 appearing to be the best choice in most of the figures.

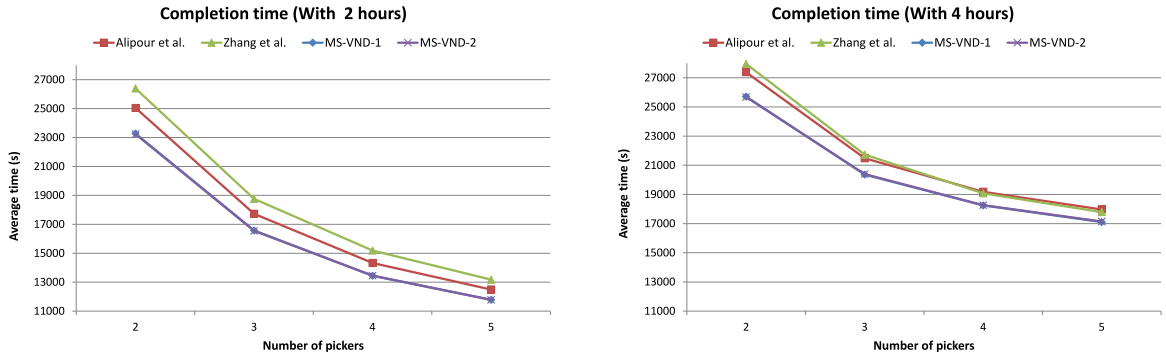
As a general conclusion, the congestion rate depends on the arrival of orders in a particular time horizon, and on the pickers available at the warehouse. Also, we can conclude that there exists a relationship between this rate and the studied objective functions. Studying the congestion rate could be used in modern and flexible warehouses to determine the number of pickers needed per shift. Higher congestion rates help to find a better workload balance and result in smaller picking times, while lower congestion rates favor the completion time, since it increases the possibilities of having pickers available as soon as an order arrives to the system.

6. Conclusions

In this paper we have studied the Online Order Batching Problem with Multiple Pickers. This problem is one of the most realistic

Table 19
Comparison of the workload balance with the state-of-the-art methods over several 5 pickers scenarios, varying the number of hours for the arrival of orders.

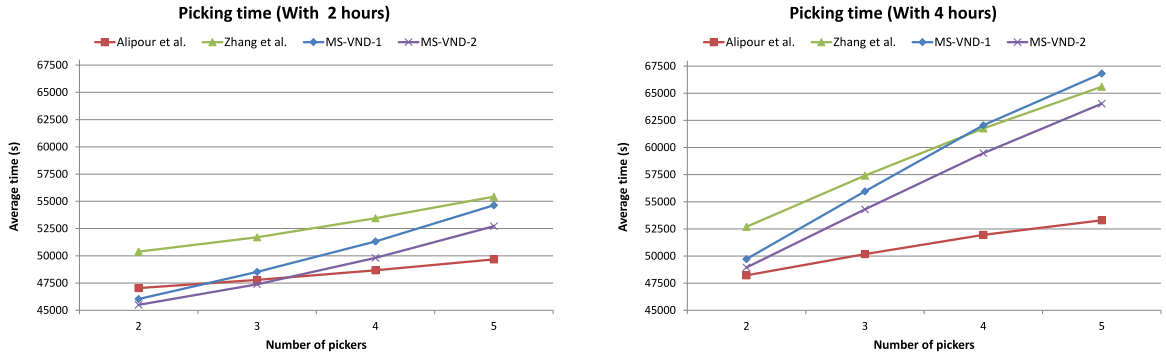
Instances	#Hours	Workload balance (With 5 Pickers)							
		State of the art				Multistart VND			
		Alipour et al. (2020)		Zhang et al. (2017)		MS-VND-1 (Workload balance)		MS-VND-2 (Picking time)	
		Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)	Avg. (s)	Dev. (%)
Albareda (80)	**1**	889	533.45%	765	444.73%	**140**	**0.00%**	674	380.14%
	2	1413	486.32%	671	178.27%	**241**	**0.00%**	611	153.63%
	3	2169	623.70%	691	130.61%	**300**	**0.00%**	668	122.92%
	4	3230	926.22%	698	121.69%	**315**	**0.00%**	653	107.59%
Henn (64)	**1**	684	201.68%	576	153.94%	**227**	**0.00%**	514	126.70%
	2	1257	274.09%	533	58.51%	**336**	**0.00%**	502	49.35%
	3	2721	568.12%	469	15.14%	**407**	**0.00%**	482	18.23%
	4	4226	1006.96%	440	15.16%	**382**	**0.00%**	457	19.76%
All (144)	**1**	798	346.39%	681	280.77%	**179**	**0.00%**	603	237.24%
	2	1344	374.40%	609	115.12%	**283**	**0.00%**	563	98.64%
	3	2414	594.75%	592	70.47%	**348**	**0.00%**	585	68.39%
	4	3673	965.98%	583	69.22%	**345**	**0.00%**	566	64.34%



(a) Averaged completion time for 2 hours scenario.

(b) Averaged completion time for 4 hourss cenario.

Fig. 4. Performance of the algorithms, in terms of completion time, when increasing the number of pickers for 2 and 4 h scenarios.

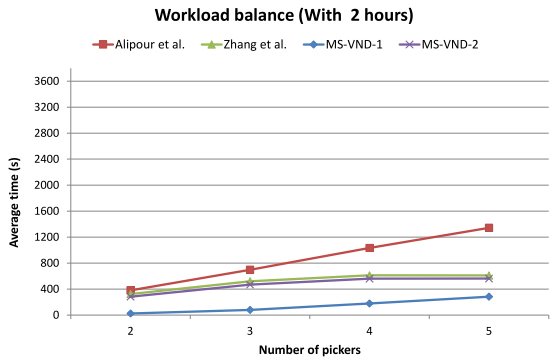


(a) Averaged picking time for 2 hours scenario.

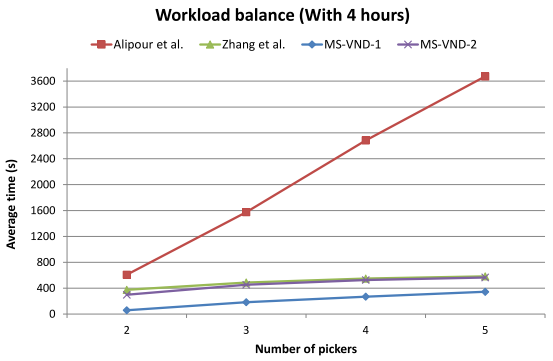
(b) Averaged picking time for 4 hours scenario.

Fig. 5. Performance of the algorithms, in terms of picking time, when increasing the number of pickers for 2 and 4 h scenarios.

variants of the Order Batching family of problems, since it considers the existence of several pickers in the warehouse at the same time, and the online arrival of orders to the system. We have classified the studied variant and identified all previous methods in the state of the art. We noticed that several objective functions had been studied for the problem, but not all previous papers report all of them. We have compiled the most relevant objective functions for the problem and empirically analyzed the behavior of the previous methods in the state of the art for all of them. Particularly, we have studied the minimization of the completion time, the minimization of the picking time, and the minimization of the differences in workload among the pickers. Then we have proposed two heuristic approaches based on a multistart VNS to tackle the problem considering all identified objective functions for the problem. Our approaches have been compared favorably with the previous methods and the differences have been found to be significant when using statistical tests. All experiments were performed over previously reported instances in the literature. Finally, we have studied the influence of the increase in the number of pickers available in the

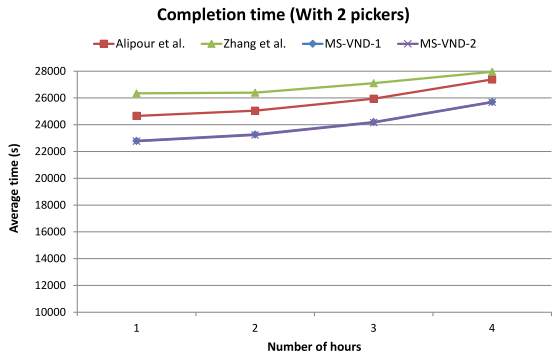


(a) Averaged workload balance for 2 hours scenario.

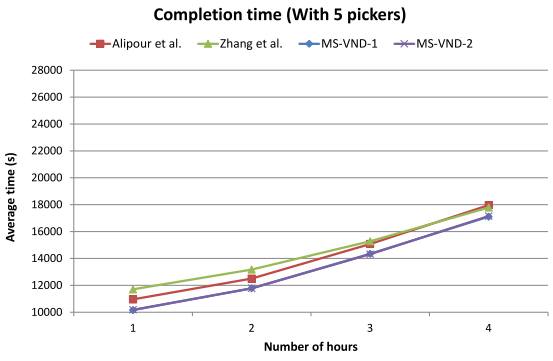


(b) Averaged workload balance for 4 hours scenario.

Fig. 6. Performance of the algorithms, in terms of workload balance, when increasing the number of pickers for 2 and 4 h scenarios.

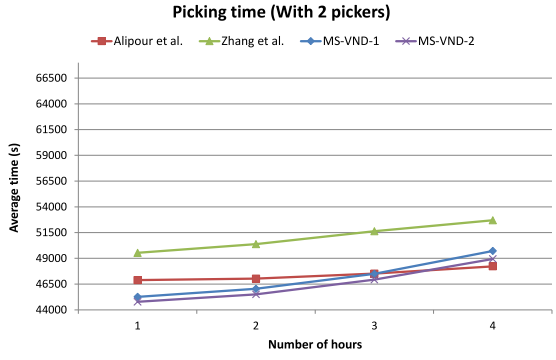


(a) Averaged completion time for 2 pickers scenario.

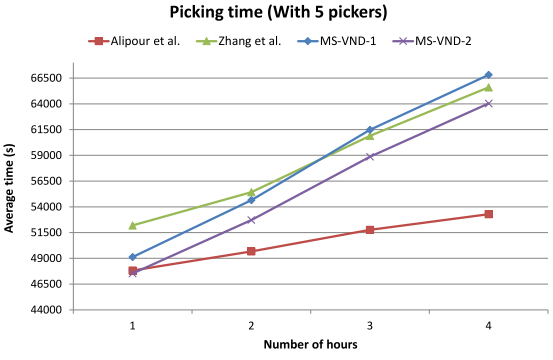


(b) Averaged completion time for 5 pickers scenario.

Fig. 7. Performance of the algorithms, in terms of completion time, when increasing the number of hours for the arrival of orders, for 2 and 5 pickers scenarios.



(a) Averaged picking time for 2 pickers scenario.



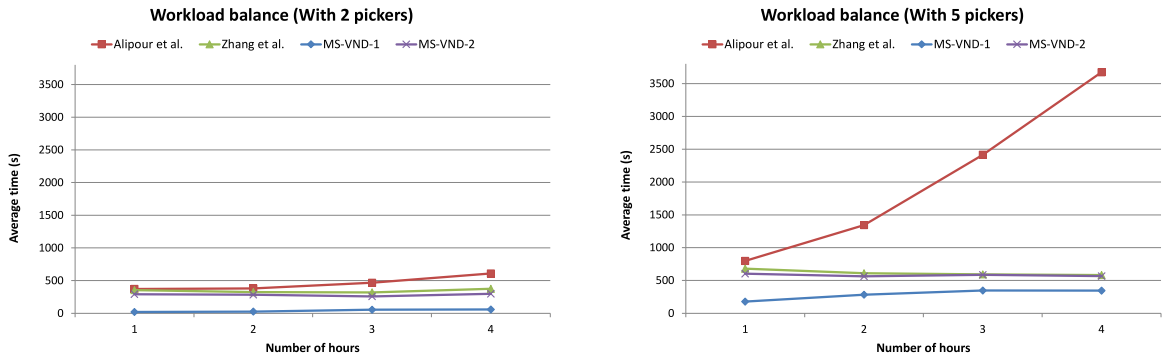
(b) Averaged picking time for 5 pickers scenario.

Fig. 8. Performance of the algorithms, in terms of picking time, when increasing the number of hours for the arrival of orders, for 2 and 5 pickers scenarios.

system and the influence of the congestion rate on the arrival of orders in several scenarios. Next, we expose our conclusions derived from the analysis of the obtained results.

Search procedures usually use the objective function being optimized to guide the search looking for better solutions. However, this approach is harder to follow when studying more than one objective function at the same time. Analyzing the search strategies proposed in this paper, we observed, as expected, that using the objective function being optimized as the guiding function usually results in a better performance in the optimization of that specific objective function. In the case of the OOBPMP, we studied two different guiding functions and we observed that either using the workload balance or the picking time results in a reasonable guiding function to reduce the completion time.

The number of pickers available and the congestion rate in the arrival of orders highly influence in the efficiency of the warehouse and both factors are closely related. On one hand, increasing the number of pickers results in shorter completion times and lower workload for



(a) Averaged workload balance for 2 pickers scenario.

(b) Averaged workload balance for 5 pickers scenario.

Fig. 9. Performance of the algorithms, in terms of workload balance, when increasing the number of hours for the arrival of orders, for 2 and 5 pickers scenarios.

Table 20
Friedman rank test for different test scenarios, when studying the workload balance.

	Friendman rank test — Workload balance				
	Alipour et al. (2020)	Zhang et al. (2017)	MS-VND-1 (Workload Balance)	MS-VND-2 (Picking Time)	Sig. (*p*-value)
2 h	3.37	2.72	**1.27**	2.64	**0.000**
4 h	3.66	2.48	**1.49**	2.37	**0.000**
2 pickers	3.24	2.88	**1.25**	2.64	**0.000**
5 pickers	3.65	2.48	**1.45**	2.42	**0.000**

each picker, however, it is harder to find a better balance among the work performed by each picker, and the overall picking time increases (since pickers collect fewer items in each picking tour). This fact might result in an increase in the energy consumption. On the other hand, the study and segmentation of the congestion rate in the arrival of orders should be used in modern and flexible warehouses to determine the number of pickers needed per shift. Fewer pickers than necessary might result in delays in the completion time. On the contrary, more pickers than necessary might result in more dead time in the activity and a deterioration in the picking time.

From a managerial point of view, the optimization of the objective functions studied in this paper might result in an increase of the benefits. Particularly, the reduction in the picking time reduces the energy consumption, while the reduction of the completion time results in a faster service of products for the customers. On the other hand, the balance of the workload results in a healthier and safer work environment and prevents the overload of the machinery.

Since real scenarios look for an increase in the benefits, future research should focus on multiobjective optimization problems including the objective functions studied in this paper and discovering others. Optimizing several objective functions at the same time will provide companies with non-dominated solutions that can result useful in different scenarios.

CRediT authorship contribution statement

Sergio Gil-Borrás: Conceptualization, Investigation, Data curation, Methodology, Software. **Eduardo G. Pardo:** Conceptualization, Investigation, Validation, Writing – original draft. **Antonio Alonso-Ayuso:** Supervision, Formal analysis, Writing – reviewing & editing. **Abraham Duarte:** Supervision, Writing – reviewing & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partially funded by the projects: RTI2018-094269-B-I00, and PGC2018-095322-B-C22 from Ministerio de Ciencia, Innovación y Universidades (Spain); by Comunidad de Madrid (Spain) and European Regional Development Fund (European Union), grant ref. P2018/TCS-4566; and by Programa Propio de I+D+i de la Universidad Politécnica de Madrid (Spain) (Programa 466A).

References

Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., & De Blas, C. S. (2009). Variable neighborhood search for order batching in a warehouse. *Asia-Pacific Journal of Operational Research*, *26*(5), 655–683.

Alipour, M., Mehrjedrdi, Y. Z., & Mostafaeipour, A. (2020). A rule-based heuristic algorithm for on-line order batching and scheduling in an order picking warehouse with multiple pickers. *RAIRO-Operations Research*, *54*(1), 101–107.

Chen, F., Wang, H., Qi, C., & Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers & Industrial Engineering*, *66*(1), 77–85.

Chen, F., Wang, H., Xie, Y., & Qi, C. (2016). An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse. *Journal of Intelligent Manufacturing*, *27*, 389–408.

Chen, F., Wei, Y., & Wang, H. (2018). A heuristic based batching and assigning method for online customer orders. *Flexible Services and Manufacturing Journal*, *30*(4), 640–685.

De Koster, R. B. M., Van der Poort, E. S., & Wolters, M. (1999). Efficient order batching methods in warehouses. *International Journal of Productions Research*, *37*(7), 1479–1504.

Duarte, A., Pantrigo, J. J., Pardo, E. G., & Mladenovic, N. (2015). Multi-objective variable neighborhood search: an application to combinatorial optimization problems. *Journal of Global Optimization*, *63*(3), 515–536.

Duarte, A., Pantrigo, J. J., Pardo, E. G., & Sánchez-Oro, J. (2016). Parallel variable neighbourhood search strategies for the cutwidth minimization problem. *IMA Journal of Management Mathematics*, *27*(1), 55–73.

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

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2020a). Fixed versus variable time window warehousing strategies in real time. *Progress in Artificial Intelligence*, *9*, 315–324.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2020b). GRASP with variable neighborhood descent for the online order batching problem. *Journal of Global Optimization*, *78*(2), 295–325.

Hahn, S., Scholz, A., et al. (2017). *Order picking in narrow-aisle warehouses: A fast approach to minimize waiting times*. (Tech. Rep. 170006), Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, FEMM Working Papers.

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

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, *130*(3), 449–467.

Hansen, P., Mladenović, N., & Moreno-Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, *175*(1), 367–407.

Henn, S. (2012). Algorithms for on-line order batching in an order picking warehouse. *Computers & Operations Research*, *39*(11), 2549–2563.

Henn, S. (2015). Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. *Flexible Services and Manufacturing Journal*, *27*(1), 86–114.

Henn, S., Koch, S., Doerner, K. F., Strauss, C., & Wäscher, G. (2010). Metaheuristics for the order batching problem in manual order picking systems. *Business Research*, *3*(1), 82–105.

Henn, S., Koch, S., & Wäscher, G. (2012). Order batching in order picking warehouses: a survey of solution approaches. In *Warehousing in the global supply chain* (pp. 105–137). Springer.

Ho, Y.-C., & Tseng, Y.-Y. (2006). A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. *International Journal of Productions Research*, *44*(17), 3391–3417.

Hojaghania, L., Nematian, J., Shojaiea, A. A., & Javadi, M. (2019). Metaheuristics for a new MINLP model with reduced response time for on-line order batching. *Scientia Iranica*.

Koch, S., & Wäscher, G. (2016). A grouping genetic algorithm for the order batching problem in distribution warehouses. *Journal of Business Economics*, *86*(1–2), 131–153.

Menéndez, B., Bustillo, M., Pardo, E. G., & Duarte, A. (2017). General Variable Neighborhood Search for the Order Batching and Sequencing Problem. *European Journal of Operational Research*, *263*(1), 82–93.

Menéndez, B., Pardo, E. G., Alonso-Ayuso, A., Molina, E., & Duarte, A. (2017). Variable neighborhood search strategies for the order batching problem. *Computers & Operations Research*, *78*, 500–512.

Menéndez, B., Pardo, E. G., Duarte, A., Alonso-Ayuso, A., & Molina, E. (2015). General variable neighborhood search applied to the picking process in a warehouse. *Electronic Notes in Discrete Mathematics*, *47*, 77–84.

Menéndez, B., Pardo, E. G., Sánchez-Oro, J., & Duarte, A. (2017). Parallel variable neighborhood search for the min-max order batching problem. *International Transactions in Operational Research*, *24*(3), 635–662.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097–1100.

Öncan, T. (2015). MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem. *European Journal of Operational Research*, *243*(1), 142–155.

Pardo, E. G., Mladenović, N., Pantrigo, J. J., & Duarte, A. (2013). Variable formulation search for the cutwidth minimization problem. *Applied Soft Computing*, *13*(5), 2242–2252.

Pérez-Rodríguez, R., & Hernández-Aguirre, A. (2015). An estimation of distribution algorithm-based approach for the order batching problem. *Research in Computing Science*, *93*(1), 141–150.

Pérez-Rodríguez, R., Hernández-Aguirre, A., & Jöns, S. (2015). A continuous estimation of distribution algorithm for the online order-batching problem. *International Journal of Advanced Manufacturing Technology*, *79*(1), 569–588.

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

Rubrico, J. I. U., Higashi, T., Tamura, H., & Ota, J. (2011). Online rescheduling of multiple picking agents for warehouse management. *Robotics and Computer-Integrated Manufacturing*, *27*(1), 62–71.

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. *European Journal of Operational Research*, *263*(2), 461–478.

Scholz, A., & Wäscher, G. (2017). Order batching and picker routing in manual order picking systems: the benefits of integrated routing. *Central European Journal of Operations Research*, *25*(2), 491–520.

Tang, L. C., & Chew, E. P. (1997). Order picking systems: Batching and storage assignment strategies. *Computers & Industrial Engineering*, *33*(3), 817–820, Selected Papers from the Proceedings of 1996 ICC&IC.

Van Der Gaast, J. P., Jargalsaikhan, B., & Roodbergen, K. J. (2018). Dynamic batching for order picking in warehouses. In *15th IMHRC proceedings - progress in material handling research: 2018* (pp. 1–8). Savannah, Georgia. USA: Digital Commons Georgia Southern.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics* (pp. 196–202). Springer.

Yu, M., & De Koster, R. B. M. (2009). The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, *198*(2), 480–490.

Zhang, J., Wang, X., Chan, F. T. S., & Ruan, J. (2017). On-line order batching and sequencing problem with multiple pickers: A hybrid rule-based algorithm. *Applied Mathematical Modelling*, *45*(Suppl. C), 271–284.

Zhao, D. G., Jiang, Y., Bao, J. W., Wang, J. Q., & Jia, H. (2019). Study on batching and picking optimization of marine outfitting pallets. In *MATEC web of conferences ICFMCE 2018 (Vol. 272)* (p. 01015). EDP Sciences.

Žulj, I., Kramer, S., & Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, *264*(2), 653–664.