Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Online Order Batching Problem: a heuristic approach for single and multiple pickers

Tesis Doctoral

D. Sergio Gil Borrás
Máster en Ciberseguridad
por la Universidad Carlos III

2022

Departamento de Sistemas Informáticos

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Online Order Batching Problem: a heuristic approach for single and multiple pickers

D. Sergio Gil Borrás
Máster en Ciberseguridad
por la Universidad Carlos III

Supervised by

Dr. D. Eduardo García Pardo
Doctor en Sistemas Telemáticos e Informáticos
por la Universidad Rey Juan Carlos.

Dr. D. José Ernesto Jiménez Merino
Doctor en Lenguajes y Sistemas Informáticos e Ingeniería de Software
por la Universidad Rey Juan Carlos.

2022

POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

ACTA DE DEFENSA DE TESIS DOCTORAL

(Art. 30 del Reglamento de Elaboración y Evaluación de la Tesis Doctoral, aprobado el 21 de diciembre de 2011)

En Madrid, a ___ de _____ del 2022, en la **Escuela Técnica Superior de Ingeniería de Sistemas Informáticos Universidad de la Universidad Politécnica de Madrid**, se reúne y constituye el Tribunal encargado de juzgar la tesis doctoral de D. **Sergio Gil Borrás** con Documento de Identidad Nº. **46843187-F** Ingeniero de Computadores, por la Universidad Politécnica de Madrid.

El título de la tesis es "**Online Order Batching Problem: a heuristic approach for single and multiple pickers**" y ha sido dirigida por D. **Eduardo García Pardo** y por D. **José Ernesto Jiménez Merino** dentro del **programa de doctorado en Ciencias y Tecnologías de la Computación para *Smart Cities***, adscrito a la **Escuela Técnica Superior de Ingeniería de Sistemas Informáticos Universidad de la Universidad Politécnica de Madrid,** Departamento de Sistemas Informáticos, con código de AGORA _____ y según Real Decreto **(*) R.D. 99/2011**

El Tribunal, aprobado por la Comisión de Doctorado de la Universidad Politécnica de Madrid en su reunión del _____ , está integrado por los siguientes doctores:

Presidente
D/Dª _____

Vocales
D/Dª _____
D/Dª _____
D/Dª _____

Secretario
D/Dª _____

Suplentes
D/Dª _____
D/Dª _____

El Sr. Presidente significa que se ha dado cumplimiento a todos y cada uno de los trámites previstos en el R.D. 99/2011, de 28 de enero, y en el Reglamento de Elaboración y Evaluación de la Tesis Doctoral de la U.P.M.

El Tribunal declara abierta la sesión pública compareciendo ante él D. **Sergio Gil Borrás**

El Secretario del tribunal da lectura a los Artículos 29 y 30 del Reglamento de Elaboración y Evaluación de la Tesis Doctoral de la U.P.M.

Siendo las _____ horas, y en aplicación de dichos preceptos, el doctorando da comienzo a su actuación, consistente en la exposición de la labor realizada, la metodología, el contenido y las conclusiones, con una especial mención a sus aportaciones originales.

(*) R.D. 1393/2007 o R.D. 99/2011

ACTA DE DEFENSA DE TESIS DOCTORAL

(Art. 30 del Reglamento de Elaboración y Evaluación de la Tesis Doctoral, aprobado el 21 de diciembre de 2011)

Expuestas las opiniones de los miembros del Tribunal sobre la tesis leída, y oídas las respuestas del doctorando a las cuestiones y objeciones formuladas por aquellos, el Presidente invita a los doctores presentes en la sala a que formulen las cuestiones y objeciones que consideren oportunas.

Posteriormente, el Tribunal invita al doctorando y al público a que se ausenten de la sala y, reunido en sesión privada, comienza su deliberación, para lo cual todos y cada uno de los miembros del tribunal exponen su criterio con respecto a la actuación del alumno en defensa de su tesis doctoral.

Concluye la deliberación del Tribunal y, previa votación en sesión secreta, a las _____ horas se acuerda otorgar a la tesis doctoral la calificación de:

☐ No apto
☐ Aprobado
☐ Notable
☐ Sobresaliente

Cada miembro del tribunal emite un voto secreto proponiendo si procede la obtención de la mención "cum laude". Estos votos son introducidos en un sobre, que queda cerrado y firmado en la solapa por todos los miembros del tribunal.

Certificación de Mención Internacional (solo en caso necesario)

El Secretario del Tribunal certifica:

☐ Que al menos un experto perteneciente a alguna institución de educación superior o centro de investigación no española, con el título de doctor, y distinto del responsable de la estancia realizada por el doctorando, ha formado parte del tribunal evaluador de la tesis.
☐ Que parte de la tesis doctoral, al menos el resumen y las conclusiones, se ha redactado y se ha presentado en una de las lenguas habituales para la comunicación científica en su campo de conocimiento, distinta a cualquiera de las lenguas oficiales en España. Esta norma no será de aplicación cuando las estancias, informes y expertos procedan de un país de habla hispana.

Por tanto, y teniendo en cuenta el visto bueno de la Comisión de Doctorado de la UPM,

☐ Procede otorgar la mención de "Doctor Internacional"
☐ No procede otorgar mención

Madrid, a _____ de _____ de 20__

El/La Presidente/a,

D/Dª.

El/La Secretario/a,

D/Dª

Vocal,

D/Dª

Vocal,

D/Dª

Vocal,

D/Dª

Acknowledgement

First of all, I thank Professor Ph.D. Eduardo García Pardo for his work as the first supervisor of this doctoral thesis, in which he has shown not only great knowledge, but also understanding and empathy throughout all these years, without which the work carried out here would not have been possible. He encouraged me to tackle this difficult path and, thanks to him, this doctoral thesis reflects all the effort made during all these years. Thank you very much! In the same way, I would like to thank Professor Ph.D. José Ernesto Jiménez Merino as the second director and tutor for his support and unconditional help at all times.

I also want to thank Professors Ph.D. Antonio Alonso Ayuso, Abraham Duarte, and Kenneth Sörensen, who have collaborated in the research and preparation of the different publications comprised of this doctoral thesis. I also want to thank Professor Kenneth Sörensen for the opportunity he gave me to carry out a research stay together with his research group at the University of Antwerpen in Belgium. I also want to thank the rest of the research group colleagues for the good reception they gave me. Thank you all!

I also thank the Vicerrectorado de Investigación, Innovación y Doctorado for their financial support through the "*Programa Propio de la UPM*", which has made it possible for me to dedicate myself completely to the preparation of this doctoral thesis.

I want to thank the other people who have helped me complete this doctoral thesis: my parents, my girlfriend and the rest of my family for their infinite patience and good advice, because the life of a doctoral student is full of pressure, difficulties, as well as good and bad times. Each goal achieved is a small step along the way, but one that fills us with great satisfaction.

Finally, I would like to thank all the professors, classmates, and friends who have helped to make this thesis a reality today.

"Only those who will risk going too far can possibly find out how far one can go."

Thomas Stearns Eliot

Abstract

The recent expansion of electronic commerce has promoted the development of numerous sectors around it. Among these sectors are those related to the supply chain management. Within the supply chain, logistic warehouses play a key role, being responsible for receiving, storing, and collecting products, which must be delivered to other warehouses or final customers. Generally, the objective of the operations in a warehouse are devoted to reducing delivery times. To that aim it is important to have efficient storage and collection strategies.

There are multiple problems within logistic warehouses that need to be solved. Many of these problems can be defined as optimization problems. Among them, problems related to the order picking process stands out. Order picking can be tackled with different picking policies. Among the best-known order picking policies, we can find "Strict Order Picking" and "Order Batching". The former one is characterized by picking each order individually, i.e. the picker starts a new collection route each time a new order needs to be picked. On the other hand, the Order Batching policy is characterized by the fact that orders are grouped into batches, and the collection of all the orders associated with the same batch is carried out on the same picking route, normally by a single picker.

This Doctoral Thesis focuses on solving several optimization problems belonging to the family of Order Batching Problems (OBP), which appear when a batch collection policy is used in the picking process of a warehouse. More specifically, this Doctoral Thesis focuses on solving the task of determining the grouping of orders into batches, commonly named as "batching". The resolution of a problem belonging to the OBP family implies also addressing other tasks such as: determining the next batch to be picked, selecting the order picker who will carry out the picking, establishing the picking route within the warehouse, determining the time that a picker must wait before starting a new route, etc. These tasks vary depending on the variant of the problem studied. A possible classification of the variants would divide them into Offline/Online, depending on the availability of information regarding to the arrival of new orders. It is also possible to classify them as single picker/multiple pickers depending on whether there are one or several pickers collecting the orders simultaneously. Variants of the problem can also be identified according to the objective function studied.

This Doctoral thesis focuses on the online variants of the problem, which are characterized by being dynamic optimization problems where orders arrive at the system continuously, i.e., while the collection process is undergo. In this context, there is no information about the next order that will arrive at the system. This type of scenario could be considered the most realistic nowadays, given that the online sale of products is continuously happening, as e-Commerce platforms operate 24 hours a day. Among the different online variants of the OBP family, in this Doctoral Thesis, the batching task is studied both: single picker context, and multiple pickers context. In addition, the task that determines the time window is also highlighted and studied. This task occurs only in online variants of the problem and it has been little studied in the literature, but it has a great impact on the obtained results. Determining the time window consists of determining the best time for the picker to leave and collect the generated batch. Delaying the start of the picking route means that new orders may arrive at the system, which could benefit a better composition of the remaining batches.

The resolution of the batching task belongs to the $\mathcal{NP}$-hard computational complexity class, so it is not possible to efficiently determine the optimal solution to the problem in a reasonable amount of time, when the size of the problem is large, as it is the case in most real situations.

Heuristic and metaheuristic techniques are used to address the problems described above, since these techniques provide high-quality solutions for $\mathcal{NP}$-Hard problems in short computing times. Heuristics are used both to generate an initial solution to the problem (constructive heuristics) and to search for better-quality solutions in the neighborhood about a given solution (search heuristics). The latter present the difficulty of being trapped in local optima, that is, in the best possible solution within a specific neighborhood. Metaheuristics are high-level heuristics capable of escaping from a local optimum and reaching others belonging to different neighborhoods. In the case of this Doctoral Thesis, different construction and local search algorithms have been proposed to solve the problems, together with the use of the Variable Neighborhood Search and Greedy Randomized Adaptive Search Procedure metaheuristics. The algorithms proposed for each of the different variants studied have been able to improve, in their respective contexts, the existing algorithms in the state of the art. Finally, we point out that the most relevant results obtained have been published in prestigious international scientific forums.

Resumen

La reciente expansión del comercio electrónico ha impulsado el desarrollo de numerosos sectores a su alrededor, entre los que se encuentran todos aquellos relacionados con la cadena de suministro. Los almacenes logísticos juegan un papel clave en dicha cadena, siendo los responsables de la recepción, almacenaje y posterior recogida de productos, que deben ser servidos a otros almacenes o a clientes finales. De manera general, se persigue el objetivo de reducir los tiempos de entrega, para lo que es relevante disponer de estrategias de almacenaje y recogida eficaces.

Dentro de los almacenes logísticos, deben resolverse múltiples problemas que pueden ser enunciados como problemas de optimización. Entre ellos, destacan los problemas relacionados con la recogida de pedidos, que puede seguir diferentes estrategias o políticas de recogida. Entre las más conocidas se encuentran las políticas *Strict Order Picking* y *Order Batching*. La primera, está caracterizada por realizar una recogida individual de los pedidos, es decir, el operario inicia una nueva ruta de recogida cada vez que recoge un nuevo pedido. Por otro lado, la política *Order Batching* se caracteriza porque los pedidos son agrupados en lotes y la recogida de todos los pedidos asociados al mismo lote se realiza en una misma ruta, normalmente por un único operario.

Esta Tesis Doctoral se centra en la resolución de algunos problemas de optimización pertenecientes a la familia denominada *Order Batching Problems* (OBP), que aparecen cuando se emplea una política de recogida por lotes. De manera más concreta, esta Tesis Doctoral está centrada en la resolución de la tarea consistente en determinar la agrupación de pedidos en lotes, comúnmente denominada *batching*, si bien, la resolución de un problema perteneciente a la familia OBP implica abordar también otras tareas tales como: determinar el siguiente lote a recoger, seleccionar el operario que realizará la recogida, establecer la ruta de recogida dentro del almacén, determinar el tiempo que un operario debe esperar antes de iniciar una nueva ruta, etc. Estas tareas varían según la variante del problema estudiada. Una posible clasificación de las variantes las dividiría en *Offline*/*Online*, según la disponibilidad de la información relativa a la llegada de nuevos pedidos. También es posible clasificarlas como *Single Picker*/*Multiple Pickers* según se disponga de uno o varios operarios recogiendo los pedidos simultáneamente. Las variantes también pueden diferenciarse según la función objetivo que se estudie. Esta Tesis Doctoral se centra en las variantes *Online* del problema, que se caracterizan por ser problemas de optimización dinámica donde los pedidos llegan al sistema de manera continuada, es decir, mientras que el proceso de recogida está en marcha. En este contexto, no se tiene ninguna información del siguiente pedido que llegará al sistema. Este tipo de escenario podría considerarse como el más realista hoy en día, dado que la venta de productos *online* es continua, al estar las plataformas de comercio electrónico 24h al día operativas. Entre las diferentes variantes *Online* del OBP, en esta Tesis Doctoral se estudia la tarea de *batching* tanto en contextos con un único operario, como en contextos con múltiples operarios. Además, se destaca la identificación y el estudio de la tarea que determina el tiempo de ventana. Esta tarea solo se da en las variantes *Online* del problema y ha sido muy poco estudiada en la literatura, pero tiene un gran impacto sobre los resultados obtenidos. Determinar el tiempo de ventana consiste en determinar el mejor momento de salida del operario para realizar la recogida del lote

generado. Retrasar el tiempo de salida del operario produce que puedan llegar nuevos pedidos al sistema, lo que podría beneficiar una mejor composición de los lotes restantes.

La resolución de la tarea de *batching* tiene una complejidad computacional $\mathcal{NP}$-Difícil, de manera que no es posible determinar de manera eficiente la solución óptima al problema en un tiempo razonable, cuando el tamaño del problema es grande, como es el caso de la mayoría de las situaciones reales. Para abordar los problemas descritos anteriormente se emplean técnicas heurísticas y metaheurísticas, ya que estas técnicas son capaces de ofrecer soluciones de gran calidad, en un corto periodo de tiempo, para problemas $\mathcal{NP}$-Difíciles. Las heurísticas son empleadas tanto para generar una solución inicial al problema (a través de heurísticas constructivas) como para buscar soluciones de mejor calidad en la vecindad de una solución dada (heurísticas de búsqueda). No obstante, estas presentan la dificultad de quedar atrapadas en óptimos locales, es decir, en la mejor solución posible dentro de una vecindad concreta. Las metaheurísticas son heurísticas de nivel superior que complementan a las heurísticas de búsqueda, siendo capaces de escapar de un óptimo local y alcanzar otros pertenecientes a distintas vecindades. En el caso de esta Tesis Doctoral se han propuesto diferentes algoritmos constructivos y de búsqueda para la resolución de los problemas anteriormente mencionados, junto con la utilización de las metaheurísticas *Variable Neighborhood Search* y *Greedy Randomized Adaptive Search Procedure*. Los algoritmos propuestos para cada una de las diferentes variantes estudiadas han sido capaces de mejorar, en sus respectivos contextos, a los algoritmos existentes en el estado del arte. Por último, hay que destacar que los resultados más relevantes obtenidos durante el desarrollo de esta Tesis Doctoral han sido publicados en foros científicos internacionales de reconocido prestigio.

Acronym List

» **ACO**: Ant Colony Optimization

» **ALNS**: Adaptive Large Neighborhood Search

» **AMRs**: Autonomous mobile robots

» **AS/RS**: Automated Storage and Retrieval System

» **BVNS**: Basic Variable Neighborhood Search

» **C&W**: Clarke and Wright

» **CL**: Candidate List

» **EDD**: Earliest Due Date

» **ESD**: Earliest Start Date

» **FCFS**: First Come, First Served

» **GA**: Genetic Algorithm

» **GRASP**: Greedy Randomized Adaptive Search Procedure

» **GVNS**: General Variable Neighborhood Search

» **ILS**: Iterated Local Search

» **JCR**: Journal Citation Reports

» **MDP**: Markov Decision Process

» **MO-VNS**: Multi-Objective Variable Neighborhood Search

» **OBP**: Order Batching Problem

» **OOBP**: Online Order Batching Problem

» **P/D**: Pick-up and Delivery

» **PSO**: Particle Swarm Optimization

» **PVNS**: Parallel Variable Neighborhood Search

» **RCL**: Restricted Candidate List

» **RVND**: Random Variable Neighborhood Descent

» **RVNS**: Reduced Variable Neighborhood Search

» **SJR**: Scimago Journal & Country Rank

» **SVNS**: Skewed Variable Neighborhood Search

» **TS**: Tabu Search

» **TSP**: Travelling Salesman Problem

» **UML**: Unified Modeling Language

» **VFS**: Variable Formulation Search

» **VND**: Variable Neighborhood Descent

» **VNDS**: Variable Neighborhood Decomposition Search

» **VNS**: Variable Neighborhood Search

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This Doctoral Thesis addresses the study of several variants of the Online Order Batching Problem (OOBP). The OOBP is an optimization problem that occurs during the picking of orders in logistic warehouses. Logistic warehouses are devoted to manage goods that need to be received, stored, and picked for a later delivery. They are an essential part of the supply chain, and optimization of its management plays an important role in the reduction of costs and improvement of the quality of service offered to the final customer. Among the activities that occur within a warehouse, order picking is one of the most important and requires a large amount of resources.

In this chapter, we first contextualize the research carried out and explain the motivations that led us to do this research. Then, we present the general concept of an optimization problem. In addition, we examine different types of optimization problems and review the techniques used to solve them. Later, we focus on the optimization problems within the supply chain. We continue by presenting the scientific methodology that we follow in this Doctoral Thesis, as well as the approximate optimization methodology used to solve the problem studied here. Finally, we propose the hypothesis and objectives of this Doctoral Thesis and summarize the structure of the rest of the document.

1.1 Context and motivation

Nowadays, there is a boom in electronic commerce (e-Commerce) where new customers continuously increase the demand for products that are sold online. This increase means that companies must improve their logistic processes to provide customers with their products as soon as possible and at a lower cost. Specifically, consumers want to receive new products purchased in a very short period of time, and this behavior is becoming more prevalent in society. The way to reduce delivery time is to develop new systems and improve current procedures in supply chain management. Amazon[1] is one of the best examples, since it has become one of the leaders in the world of e-Commerce retail in the last few years. Amazon, among others, has very fast delivery times, with some products delivered even in less than 12 hours. Amazon has become a leader due to the early digitalization of its production system and the associated reduction in operating expenses [78]. This fact has forced other companies involved in retail commerce to invest a large amount of resources to improve their processes. Enhancing the process is the only way to remain competitive in an increasingly globalized market.

E-Commerce retail is still a growing sector compared to traditional retail [86]. This trend can be seen in Figure 1.1, where the results of the last few years and a forecast for the next few years are shown. This figure includes total sales in trillions of dollars, the percentage of

[1] Amazon.com Inc.

e-Commerce sales over total retail sales, and the percentage of change from e-Commerce sales at the retail level in the period 2017 to 2025. The data presented in this figure are consolidated up to 2021 and are an estimate from that point on. As we can observe, it is expected that in a few years, worldwide e-Commerce will reach around 24% of total retail sales.

The strong growth of the market has led us to place our interest in the optimization of e-commerce processes related to the supply chain. In particular, we focus on finding technological solutions that improve productivity in supply chain management and, therefore, in retail e-Commerce.

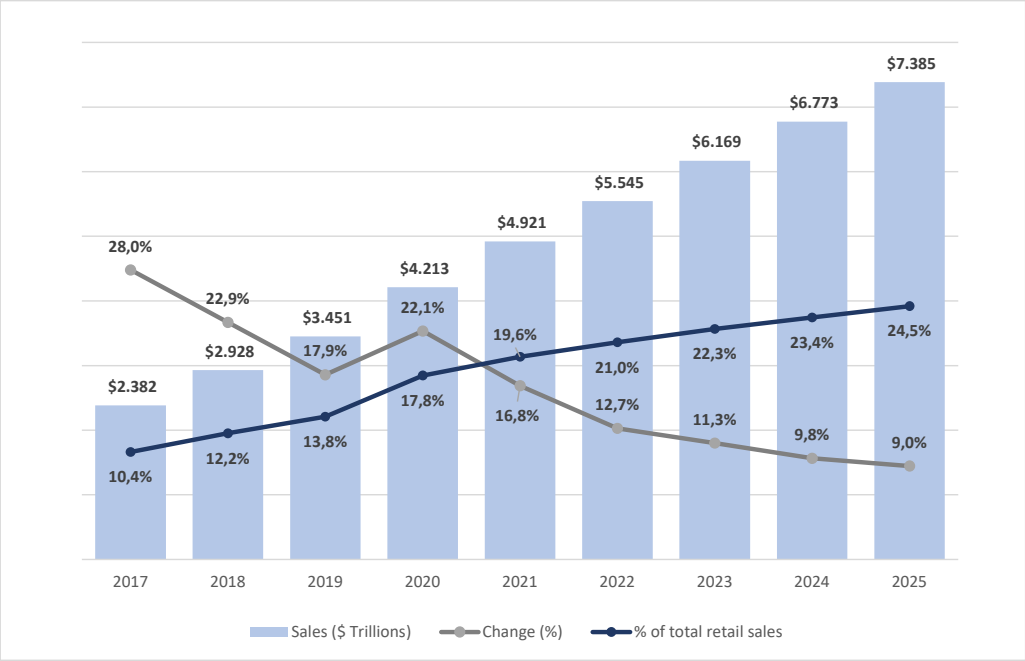


Figure 1.1: Global online retail e-commerce growth (2017-2025). Data source: Insider Intelligence.

There are many processes within supply chain management that need to be optimized with new solutions, leading to an overall improvement in cost and/or time. If we pay attention to Amazon, we can see a strong expansion of its logistic centers in all its areas of influence, as well as a strong investment in digitalizing the processes in logistic warehouses [166]. We often find many optimization problems in the supply chain, so we want to participate in the search for new solutions to help the global market. But there are many problems and decisions to be solved in the supply chain [114, 194]. These decisions are classified according to its execution deadlines, resulting in three levels of decision (operational, tactical, and strategic) in the management of the supply chain and, more specifically, in the logistic of a warehouse [54]. Here, we review some necessary decisions in the logistic of a warehouse which are related to order picking.

The sets of decisions at each decision level can be very different, and only one decision can include several optimization problems. Strategic decisions are decisions that are executed in the long term (3 to 5 years). Tactical decisions are decisions that are executed in the medium term (1 to 3 years). Finally, operational decisions are decisions that are executed in the short term (up to 1 year). Among strategic decisions, we can find: the design of the warehouse layout and its location, the level of mechanization and the system used, or the origin of the products to be processed in the warehouse (frozen, large volumes, or weights), among others. Among tactical decisions, we can find the placement of items in the warehouse, the type and capacity of the picking systems, the number of pickers, assigned tasks and their work shifts, or the number of the pick-up and delivery (P/D) points (also known as depot) and its location, among others.

Finally, operational decisions typically correspond to the set of tasks (Routing, Batching, among others) that we need to address when solving the order-picking process. All decisions that affect the order picking for each level of management can be found in [94], this paper is attached to this Doctoral Thesis in Section 3.1.

In this Doctoral Thesis, we focus on an operational-level problem. Specifically, the order picking problem, because there are many studies in the state of the art where it is confirmed that it is the most important operation in cost, for example, according to the author, Colin Drury affirmed in 1988, [59] the order picking task supports 65% of the total operational costs for a typical warehouse. Other authors, Robert G. Coyle et al. in 1996, [45], detailed that between 50% and 75% of the total operational costs of a warehouse are attributed to picking operations. Years later, Edward Frazelle in 2002 [82] identified order picking as the most expensive operational task in manual and automatic warehouses, because it represents around 55% of overall expenses. Subsequently, James A. Tompkins et al. in 2003 and 2010 [270] published talking from the cost perspective that order picking tasks represent approximately 55% of the total operational costs. Later, in 2006, Alan Rushton et al. [245] wrote that the order picking process accounts for about 50% of the direct labor costs of a warehouse. In the literature, much research on the importance of the order picking process in logistic warehouses has been conducted, motivating us to contribute new solutions to the problem. We can find more information on the cost of collect and storage orders in [246, 257].

The problems studied in this Doctoral Thesis are order picking, when the pick policy is by batches. In this case, the problem is known as order batching. But other decisions at all levels directly affect the order batching problem. Some of these decisions define the variant of the problem to be addressed, and other decisions are part of problem instances. In addition, we need to solve several operational decisions, also known as tasks, to address the order batching problem. We can find detailed information on the different variants of the problem and its instances in Chapter 2, as well as the details of the problem that we address.

1.2 Optimization problems

Operational research is a field of mathematics in which optimization problems are studied. Optimization problems are found in multiple application fields, such as economics, finance, biology, or engineering, among others. Since the 17^{th} century, scientists and mathematicians have tried to develop new methods to address these types of problem. However, operational research did not begin to be studied as a branch of applied mathematics until the first half of the 20^{th} century. The term linear programming was also coined at the same time. Since then, linear programming has defined some particular optimization problems in which the objective function can be expressed as a linear function. Later, in the middle of the 20^{th} century, George Dantzig published the well-known Simplex algorithm [159] to solve linear programming problems [253, 260].

Optimization problems consist of finding the best value, maximum, or minimum in a real function. This value is known as the optimal value, and the function is known as the objective function. In addition, problems have different constraints that are applied to the variables that represent the solution. The target of the problem is to find the optimal value in the function domain. In this sense, the search can be performed in the entire function domain or only in part of it. The selected domain is limited by the constraints of the problem. Among all possible sets of solutions, those which satisfy the constraints of the problem are named feasible solutions.

Optimization problems can be defined from a mathematical perspective. The problem consists of one or more objective functions (f). Having $f : A \to \mathbb{R}$ and $A \subseteq \mathbb{R}^n$, the domain A of the function f is known as a set of solutions. The set of solutions is delimited by a set of

constraints that the solutions in A must satisfy. Each constraint is an equation or inequation that limits A. All elements of A that satisfy the set of all constraints of the problem are known as feasible solutions or candidate solutions. When the optimization problem minimizes f, it consists of finding the element x_0, such that $x_0 \in A : f(x_0) \leq f(x), \forall\, x \in A$. Similarly, in the case where the optimization problem maximizes f, consists of finding the element x_0, we can define it as $x_0 \in A : f(x_0) \geq f(x), \forall\, x \in A$. The standard form to define a continuous optimization problem is:

$$\begin{aligned} &\textbf{minimize/maximize} && f(x) \\ &\textbf{subject to} && g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & && h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

where:

- $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimized / maximized in the n-variable vector x,
- $g_i(x) \leq 0$ are called inequality constraints
- $h_j(x) = 0$ are called equality constraints, and
- $m \geq 0$ and $p \geq 0$.

There are many types of optimization problems. A possible classification can be derived from the nature of the problem. Each optimization problem has a different and specific way of solving it. However, there are other possible classifications, so the same problem can often be classified into different categories, and thus different approaches can be used to solve it. Next, we review the most common categories of optimization problems. The best known is the convex programming problem [16]. The main characteristic of convex programming is that every local optimal is a global minimum. Other types of optimization problems are linear programming problems [253, 260]. The characteristics of identifying a linear programming problem are that the objective function and the constraints are linear. Among the optimization problems, we can also find geometric programming problems [17, 64]. The characteristics of identifying a problem of geometric programming are that the objective function is a polynomial [2] and the constraints are posynomials and monomials. Other types of optimization problems are semidefinite programming problems [83]. In this kind of problem, the variables in the problem are vectors and the objective function is linear, but the constraints on real variables are replaced by constraints on semidefinite matrices. Also, one should note that some variants of the optimization problems reviewed can be framed in convex programming. Additionally, linear integer programming problems, classified within the linear programming family [253, 260], are characterized by restricting the variables of the problem to be integer numbers. Furthermore, there is a special case within linear integer programming problems, denoted as linear binary programming [46, 211] (0-1 programming). In this case, linear binary programming problems are characterized because some problem variables are restricted to be 0 or 1. Other types of optimization problems are non-linear programming problems [13]. In non-linear programming, the objective function is non-linear, and the constraints can be linear and/or non-linear. Among

[2] Note that the term posynomial is not equivalent to the term polynomial. Particularly, polynomial's exponents must be non-negative integers, but its independent variables and coefficients can be arbitrary real numbers. On the other hand, the polynomial's exponents can be arbitrary real numbers, but its independent variables and coefficients must be positive real numbers. [64].

them, quadratic programming problems [112] are those non-linear programming problems in which the objective function is quadratic and the constraints can be linear and/or quadratic. Other types of optimization problems are stochastic programming problems [14, 258]. The characteristic of identifying a problem as stochastic is that some variable in the objective function or constraints of the problem are random or stochastic variables. Other types of optimization problems are known as combinatorial optimization problems. The characteristic of identifying an optimization problem as combinatorial is that the set of feasible solutions is discrete or can be reduced to a discrete set. Another family of optimization problems is the multi-objective optimization problem [20, 68]. In multi-objective optimization problems, there is more than one objective function to optimize, and the objective functions considered are opposed to each other. In addition, there are a large number of optimal solutions. The set of optimal solutions is known as the Pareto frontier [173]. Then, the Pareto frontier can be defined as an optimal solution set that contains solutions that are not dominated by any other feasible solution.

As we have seen before, an optimization problem can be classified into different categories at the same time. The problem studied in this Doctoral Thesis is classified as a combinational optimization problem, because the feasible solution set is a discrete set. But the problem can also be mathematically expressed as to be classified as a binary linear programming problem; because the objective function and constraints are linear, and some variables can only have binary values (0 and 1). In this Doctoral Thesis, we tackle different variants of the problem, which are denoted as online. The characteristic of online variants is that the arrival of orders to the system is dynamic. Then, it is normal to model arrival orders with a random variable and to assume that the random variable follows a Poisson distribution [228]. This problem can then also be classified as a stochastic programming problem because some constraint depends on a random variable. But in this case, we are not going to solve the problem by following stochastic programming techniques. In this Doctoral Thesis, we will use simulation techniques [195] to solve different variants of the problem. Simulation techniques consist of simulating the values of the random variable in the problem, so we can solve it as a non-stochastic problem and use the same techniques applied to solve offline variants of the problem.

Generally, optimization problems can be solved in different ways. There are exact and approximate methods. The exact methods are able to return the optimal solution, but they can only solve small problems or some easy variants of the problem. However, many optimization problems are classified as $\mathcal{NP}$-hard. In these cases, these problems cannot be optimally solved in polynomial time [295], so the use of approximate methods is required. Approximate methods are capable of returning a solution close to the optimal one, and can find a solution to problems of all sizes, in a reasonable amount of time [153, 294]. Furthermore, we can find an approximate method with an error bound (ϵ) [154]. Only this type of approximate method with an error bound can be applied in some variants of optimization problems.

In this Doctoral Thesis, we use approximate methods. Among the approximate methods, we have used heuristic and metaheuristic methods to solve the problems studied. These techniques are detailed in depth in the following sections.

1.3 Optimization problems in logistic warehouses

There are many different tasks to tackle in logistic warehouses, such as managing and processing articles distributed in the warehouse. Among these tasks, we find the reception of products, the management of their storage and inventory, the collection of orders, the packaging of orders, the product returns, the generation of routes within the warehouse, etc. [54, 149]. Solving each task might involve tackling one or more optimization problems. Furthermore, different factors affect the correct management of the task, such as: the number of workers, the layout

of the warehouse [194], or the type of product. This Doctoral Thesis is mainly focused on the order-picking task, which results in most of the operational costs in a logistic warehouse. All tasks and factors involved in order picking are compiled and classified in [94]. This article is attached in Section 3.1 of this Doctoral Thesis.

There are several policies that can be followed to deal with order picking. The order-by-order approach, also known as strict order picking, is probably the most common and simple picking policy. It consists of collecting all items associated to a single order in the same picking tour. However, we can also use an article-by-article approach, where a determined number of items of the same type belonging to one or more orders can be collected in the same picking tour, or a batch approach. The batch approach consists of generating groups of orders that will be picked together. This approach can also generate batches of articles, or batches of articles/orders by picking zone [222]. In this Doctoral Thesis, we study the order picking policy that considers batching of orders, since it is considered one of the most efficient ones [69].

Solving the order-picking process implies handling a set of tasks, which might vary depending on the variant of the problem tackled. Particularly, some tasks such as batch generation or route generation in the warehouse, are common to all order batching problems. However, other tasks depend on the number of pickers. In particular, if the number of pickers is larger than one, the task of assigning batches to pickers appears. Similarly, the sequencing task is only needed when dealing with some objectives, such as balancing the workload or reducing the tardiness associated with delivery.

Most of the tasks can be solved individually; however, there are dependencies among them, and if we solve some of these tasks together, then we can obtain better results. In [52] it is stated that the effort to tackle batching and routing tasks together can represent savings of up to 35% of the cost compared to handling batching and routing tasks separately. In Section 2.1, we review the tasks that we need to handle within the online order batching problems and the relationships among them.

When solving an order batching problem, we can look for the optimization of different objectives with commercial or managerial interest, such as economic savings, quality of service, or the welfare of workers, among others. Each objective is represented as an objective function in an optimization problem. In this Doctoral Thesis, we study several objective functions related to the Online Order Batching Problem. For each variant of the problem studied, we optimize only one objective simultaneously (i.e., we do not solve multi-objective optimization problems). However, in some cases we report the evaluation of the solutions using more than one function, which helps us to analyze the relation between the objectives. In Section 2.3, we review the main objective functions related to order picking problems.

Finally, we would like to highlight that the variants of the problem tackled in this Doctoral Thesis are dynamic optimization problems. Among dynamic optimization problems, the online category corresponds to problems where the instance changes over time. In this case, orders arrive at the system continuously. Then, we do not have any information about the next orders that will arrive at the system in a determinate moment. As we explained before, we solve this type of problem by simulation. Specifically, we simulate the orders that come into the system. This type of problem is more difficult to solve than the static variants, but is also more realistic.

1.4 Methodology

In this section, we detail the scientific methodology followed in this Doctoral Thesis, together with the main techniques and methods used to achieve the established goals. The scientific findings of this Doctoral Thesis have been obtained following the scientific method, which is explained in Section 1.4.1. As a result, several papers have been published. To complement the

scientific methodology, we used several heuristic optimization methods to solve the different optimization problems tackled in this Doctoral Thesis. These methods are reviewed in Section 1.4.2.

1.4.1 Scientific methodology

In Figure 1.2 we graphically present the scientific methodology followed in this Doctoral Thesis. The figure details the flow and steps followed since the beginning of an investigation until the end of it, when the research is published as a scientific article. In this process, each problem tackled is identified and the state of the art of the problem is studied. Then, a hypothesis is stated and an algorithm proposal is performed to validate the hypothesis. This process is repeated until the hypothesis is validated and the results are published. It is worth mentioning that this process has been followed iteratively for each of the articles published as a result of the research carried out in this Doctoral Thesis.

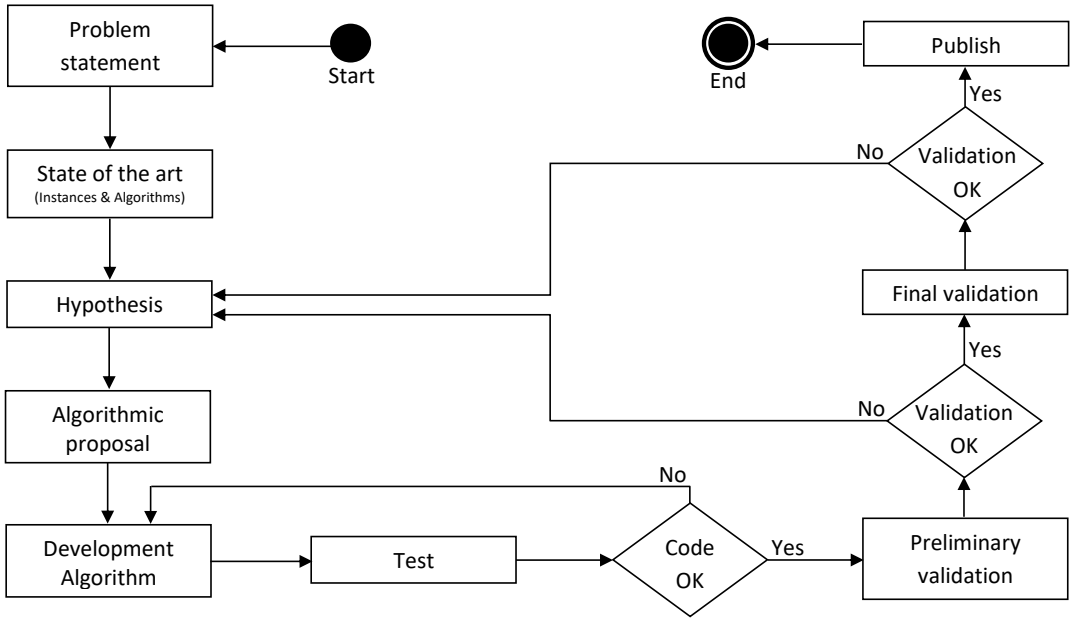


Figure 1.2: Methodological process to obtain the results for the publication of this Doctoral Thesis.

Taking a closer look, in this Doctoral Thesis we have tackled several variants of the OOBP following the next steps:

- Perform a detailed study of the main variants related to order-picking systems in logistic warehouses.
- Select an identified variant of the problem from the previous step. The variant is discriminated according to their practical interest, previous publications, availability of material, etc.
- Study the selected variant from an optimization point of view: objective function, constraints, and instances used.
- Reproduce the previous algorithms in the state of the art to compare with our future proposals.

- Propose efficient heuristic methods to build initial high-quality solutions and improve them.
- Choose the more suitable metaheuristic methodology for the problem to reach different local optima in each instance.
- Compare the algorithms proposed with the state-of-the-art methods over the reference data sets.
- Elaborate and publish the results of the research in high-impact international journals.

1.4.2 Heuristic optimization methodologies

In this Doctoral Thesis, we used an approximated optimization methodology based on heuristic and metaheuristic algorithms. Generally speaking, an approximate optimization method does not guarantee that a globally optimal solution can be found. Within approximate optimization methods, we can find heuristics [113] and metaheuristics [110, 206]. The main characteristics of heuristics are: each heuristic is characterized to provide a solution to one kind of optimization problem; heuristics are not generic strategies suitable for different problems, i.e. we must develop a new heuristic for each problem; heuristics are usually deterministic algorithms, i.e., they always return the same result for the same instance; heuristics return very good solutions in very short time for some kinds of problem, sometimes better than metaheuristics methods. Heuristic methods can be further classified into constructive heuristics (i.e., heuristics that generate an initial solution) or search heuristics (i.e., the heuristics which try to improve a given feasible solution). On the other hand, we can find metaheuristic methods. Metaheuristics are made up of one or more heuristic components. The main characteristics of the metaheuristics are: they are generic strategies for different problems, i.e., these can be adapted to work in each kind of problem; they are based on efficiently exploring the search space in order to find near–optimal solutions; metaheuristics are usually non-deterministic algorithms because they use a random process to explore the search space; metaheuristics use very different techniques to find the best solution to an optimization problem, so it is very common to combine different intensification techniques with diversification techniques to escape from a local optimal, or with complex learning processes, i.e., all these strategies guide the search process in the solution space. In addiction, the results obtained when using metaheuristics are usually better than those obtained just with heuristics in most scenarios. There is a wide variety of metaheuristic methods that can be classified into a taxonomy. Specifically, there are several articles that define different taxonomies for metaheuristics algorithms. One of the last reviews [263] the most important published taxonomies of the past few years [79, 199, 261].

As we can observe by the references, metaheuristic methods can be classified according to different factors. Among them, we highlight the method proposed to solve the problem (which can be classified into population-based methods, trajectory-based methods, naturally-inspired methods, or local search methods, among others. In these classifications, some methods belong to several categories. For example, the Ant Colony Optimization algorithm can belong to population methods, multistart methods, and local search methods. In Table 1.1, we present the main characteristics of some of the best known metaheuristic algorithms in the literature. This table was inspired by the different taxonomies published in [58].

Metaheuristic	Ref.	Acronym	Naturally Inspired	Population	Trajectory	Multi-Start	Evolutionary	Constructive	Local Search	Probabilistic Technique	Memory
Particle Swarm Optimization	[67, 156]	PSO	✔	✔					✔		
Ant Colony Optimization	[43, 57]	ACO	✔	✔		✔	✔	✔	✔		✔
Scatter Search	[106, 107]	SS		✔			✔		✔		✔
Path Relinking	[44, 109]	PR		✔					✔		✔
Genetic Algorithm	[291, 292]	GA	✔	✔			✔				
Genetic Programming	[157, 162]	GP	✔	✔			✔				✔
Evolution Strategy	[262, 289]	ES	✔	✔			✔				
Evolutionary Programming	[80, 81]	EP	✔	✔			✔				
Differential Evolution	[230, 264]	DE	✔	✔			✔				
Tabu Search	[104, 105]	TS			✔			✔	✔		✔
Greedy Randomized Adaptive Search P.	[76, 77]	GRASP			✔	✔		✔	✔		
Variable Neighborhood Search	[118, 197]	VNS			✔				✔		
Adaptive Large Neighbourhood Search	[238, 267]	ALNS			✔				✔		
Simulated annealing	[158, 280]	SA	✔		✔				✔	✔	
Estimation of Distribution Algorithm	[168, 169]	EDA					✔			✔	
Attributed Based Hill Climbing	[167, 293]	ABHC			✔			✔	✔		✔
Guided Local Search	[285, 286]	GLS							✔		✔
Iterated Local Search	[55, 180]	ILS			✔				✔		
Stochastic Local Search	[142, 143]	SLS	✔		✔				✔	✔	
Iterated Greedy	[47, 48]	IG			✔			✔	✔		
Memetic Algorithm	[200, 231]	MA	✔	✔			✔		✔		

Table 1.1: Characteristics of some of the most relevant metaheuristics.

In this Doctoral Thesis, we used different metaheuristic methods to solve the different variants of the problem we tackled. The two methods used in this work are the Greedy Randomized Adaptive Search Procedure and the Variable Neighborhood Search. They are classified in Table 1.1 as memoryless trajectory-based methods with local search procedures. In addition, the Greedy Randomized Adaptive Search Procedure is considered a constructive multi-start method. Both methods are thoroughly reviewed in the next section.

Greedy Randomized Adaptive Search Procedure

Greedy Randomized Adaptive Search Procedure (GRASP) was proposed by Thomas A. Feo and Mauricio G. C. Resende in 1995 [76]. It is a metaheuristic consisting of a multi-start algorithm composed of two phases: a construction phase and an improvement phase. The construction phase hybridizes two components: a greedy selection and a randomization. The percentage in which each component contributes to the solution is defined by a search parameter named alpha (α). The α parameter must be adjusted for each problem to improve the performance of the method. In some cases, the α parameter can be adjusted to be a new random value for each iteration of the method, which, sometimes, yields a better solution in comparison to fix its value to a predetermined value. The improvement phase is devoted to improve the solution provided by the construction phase, and it is originally based on a local search procedure (i.e., a method that results in a local optimum with respect to a particular neighborhood structure), but it can also be hybridized with another improvement algorithm (i.e., another metaheuristic).

In Figure 1.3, we present a diagram that illustrates the GRASP steps.

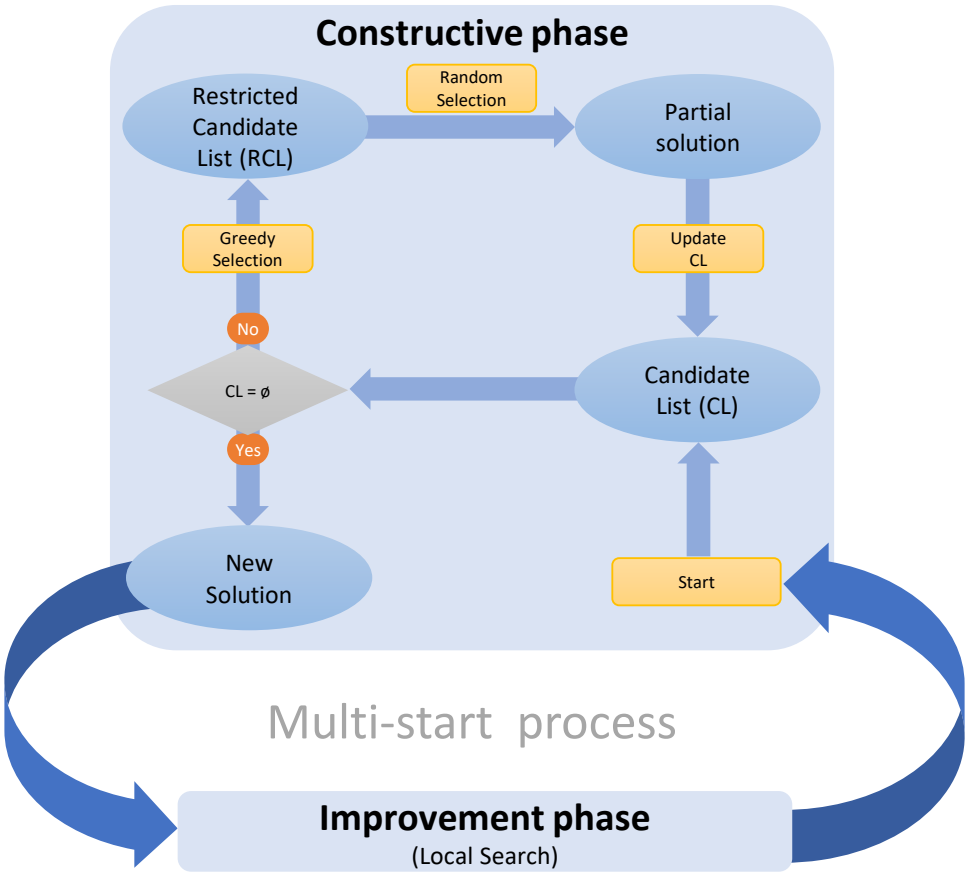


Figure 1.3: Diagram of GRASP process.

As we can observe in the figure, the construction phase is one of the key components of GRASP and the multi-start process is repeated for a number of iterations and it is considered a search parameter. We have used this GRASP constructive phase to tackle several of the problems studied in this Doctoral Thesis. To complement this figure, the pseudocode of the GRASP constructive phase is detailed in Algorithm 1. In particular, this pseudocode represents one iteration of the constructive phase. The constructive method starts with an empty solution (Step 2), a Candidate List (CL) with all the items available to be part of a solution (Step 3) and an alpha value (α) that is generated randomly, where $\alpha \in U[0, 1]$ (Step 4). Then, it initiates a loop that is repeated until CL is empty (Step 5). In the loop, first, it calculates a threshold (th) value (Step 6) based on the maximum and minimum values obtained from evaluating each candidate in CL with a greedy function ($\arg\max f(CL)$) and ($\arg\min f(CL)$) respectively and the value of the search parameter α. To continue, the procedure constructs a new Restricted Candidate List (RCL) which is created with a percentage of the best candidates from CL (Step 7) selected as those elements that qualify over th when being evaluated with f. Then, it selects a new item of RCL at random (Step 8). The selected item is then added to the solution (Step 9). Finally, the selected order is removed from CL (Step 10), and the loop is repeated until CL becomes empty.

The GRASP methodology has been used to solve the Order Batching Problem in some articles in the literature [49, 92, 93, 269]. In particular, we propose an adaptation of the GRASP constructive to handle OBP, consisting of initializing the Candidate List (CL) with all available orders in the system. Then defining a greedy function $f(CL)$ based on the size of the order and, therefore, the maximum value ($\arg\max f(CL)$) would correspond to the largest order and

Algorithm 1 Constructive GRASP

```
1: Function ConstructiveGRASP(instance)
2: solution ← ∅
3: CL ← instanceItems
4: α ← getRandomValue()
5: while CL ≠ ∅ do
6:     th ← arg max f(CL) − α(arg max f(CL) − arg min f(CL))
7:     RCL ← buildRestrictedCandidateList(th, f(CL))
8:     item ← randomOrderSelection(RLC)
9:     insertItem(solution, item)
10:    CL ← CL \ {item}
11: end while
12: return solution
```

$\arg\min f(CL)$ to the smallest order. In this case, the randomly selected order from the RCL is inserted into the first batch that has the available space to perform the insertion. If there is no space in any of the currently created batches, a new batch is created to perform the insertion. This method tries to create a compact batch solution with minimum free space in batches, but the randomization included by GRASP allows the method to produce diverse solutions in different iterations.

Variable Neighborhood Search

Variable Neighborhood Search (VNS) is a metaheuristic methodology proposed in 1997 by Pierre Hansen and Nenad Mladenovic. The VNS methodology includes a set of algorithms that follows a main idea: performing changes in the neighborhood structure during the search to access better solutions. Based on this concept, VNS algorithms have been able to successfully solve many optimization problems known to be $\mathcal{NP}$-hard. The different algorithmic schemes that compose VNS provide different strategies for any kind of optimization problem. Most of these VNS algorithms can be reviewed in [120, 124].

To understand the VNS methodology, we first need to understand some concepts that we will detail next. A neighborhood structure is the set of solutions that can be reached from any initial solution with a particular movement. A movement is defined as a change provoked in a solution by applying a particular operator, which results in a different (usually feasible) solution. Some common movements when dealing with combinatorial optimization problems are the insertion or the interchange/swap of elements in the solution. Within the VNS methodology, a neighborhood change typically occurs when a solution reaches a local optimum with the aim of escaping from that basin of attraction, allowing the method to carry on the search. To this aim, the VNS includes a disturbance or shake procedure. VNS also considers a deterministic improvement process based on a local search procedure, which explores one neighborhood until the local optimum is reached. Furthermore, some variants of the methodology consider deterministic exploration of more than one neighborhood, reaching a point that is locally optimal with respect to several neighborhoods.

With the previous concepts at hand, the VNS methodology is a versatile and open methodology, and many variants based on the same idea have emerged and evolved. Next, we detail the most important VNS variants introduced in the literature.

- **Variable Neighborhood Descent (VND)**: This method is characterized by the deterministic exploration of several neighborhood structures during the search. The explored neighborhood structures are visited sequentially and in descendant way, and the solution

provided is locally optimal with respect to several neighborhoods. This algorithm can be used as an improvement method within another VNS method or with other different methodologies, but it can also be used directly to solve the problem [62, 118].

- **Basic VNS (BVNS)**: This algorithm follows the classic structure of VNS and alternates between a disturbance phase (shake) that performs random changes in the neighborhood and an improvement phase with a single neighborhood structure that performs a deterministic exploration of the neighborhood [119, 196].
- **General VNS (GVNS)**: This algorithm follows the classic structure of BVNS but replaces the improvement phase of one neighborhood with an improvement VND procedure, which explores several neighborhood structures. [122, 198].
- **Random VND (RVND)**: This algorithm is similar to the VND, but in this case, the neighborhood structures are visited in a random way, compared to the classic VND, where the neighborhood structures are visited sequentially and in a descendant way [62, 122].
- **Reduced VNS (RVNS)**: This algorithm follows the classic structure of VNS, but it only includes the random exploration of the neighborhood through the use of a shake procedure without using a local search procedure to perform a deterministic exploration. It is used in scenarios where the neighborhood structures are very large and the exploration of all neighbors results impractical [119, 196].
- **Skewed VNS (SVNS)**: This algorithm follows a multi-start structure together with VND. In the VNS structure, the disturbance method is replaced by a generator of new solutions for each loop. The improvement method is usually a VND with several neighborhood structures [123, 196].
- **Variable Neighborhood Decomposition Search (VNDS)**: This algorithm follows the classic structure of the VNS, but extends the BVNS into a two-level VNS scheme based on the decomposition of the problem. It acts on the structure of the neighborhood in the shake and improvement phases to decompose the problem [121].
- **Variable Formulation Search (VFS)**: This algorithm follows the classic structure of VNS, but considers alternative changes in the objective function being explored, to discriminate the best solutions to carry on the search. This variant of VNS is useful when the problem presents a flat landscape with many different solutions that achieve the same value of the objective function [123, 216].
- **Parallel Variable Neighborhood Search (PVNS)**: This algorithm follows the classic structure of VNS, but includes several strategies to run the shake and improvement methods in parallel. Then, it keeps the best solution of the set on each execution [60, 190].
- **Multi-Objective Variable Neighborhood Search (MO-VNS)**: This algorithm follows the classic structure of VNS, but the disturbance and improvement methods have been modified to adapt them to multi-objective scenarios [61].

Among all these VNS schemes, three of them have been used in this Doctoral Thesis to solve the different variants of OOBP. In particular, the variants used have been the following: BVNS, VND, and GVNS.

The pseudocode of the Basic VNS scheme is introduced in Algorithm 2. As we can observe, the method includes three main steps (`Shake`, `Improvement`, and `NeighborhoodChange`). The `Shake` performs a stochastic exploration of the neighborhood and is used to escape from the basis

of attraction, the `Improvement` is devoted to reach a local optimum starting from a particular solution and with a predefined neighborhood. Finally, the `NeighborhoodChange` determines the next neighborhood to be explored, expressed as the value of the search parameter k.

Algorithm 2 Basic Variable Neighborhood Search

```
1: function BVNS(solution, k_max, t_max)
2:     repeat
3:         k ← 1
4:         while k ≤ k_max do
5:             S' ← Shake(solution, k)
6:             S'' ← Improvement(S')
7:             k ← NeighborhoodChange(solution, S'', k)
8:         end while
9:     until t < t_max
10:    return solution
11: end function
```

Once we have understood the Basic VNS process, we can jump into the deterministic exploration of more than one neighborhood structure in a deterministic way at the same time. In general, the use of different neighborhood structures occurs in the improvement phase. We search for a local optimum common to all neighborhood structures explored. To that aim, VNS methodology proposes several algorithms to achieve this. The most well-known algorithm is Variable Neighborhood Descent (VND). The pseudocode of VND with these ideas can be found in Algorithm 3. The set of neighborhood structures is defined by $\mathcal{N}_1, ..., \mathcal{N}_i$ when i is the number of total neighborhood structures. The algorithm loops until one achieves a local optimal for each neighborhood structures. The method of visiting these neighborhood structures is descent and ordered, but there are other methods which also perform this task, such as the General VNS schema, which replaces the Local Search phase of BVNS with a VND. For the sake of simplicity, we do not include an additional pseudocode of GVNS, since the algorithmic scheme is equal to the BVNS introduced in Algorithm 2 but replacing the step 6 with a VND.

Algorithm 3 Variable Neighborhood Descent

```
1: function VND(solution, N_1, ..., N_i)
2:     k ← 1
3:     k_max ← i
4:     best ← solution
5:     repeat
6:         solution' ← LocalSearch(best, N_k)
7:         if eval(solution') < eval(best) then
8:             best ← solution'
9:             k = 1
10:        else
11:            k = k + 1
12:        end if
13:    until k > k_max
14:    return best
15: end function
```

For a better understanding of the GVNS process, in Figure 1.4 we illustrate a few steps of a search process within a GVNS. In this figure, we can see in a representation of the complete

process for two rounds of the algorithm when the value of the k parameter is 1 and 4. In the example, the improvement phase is a VND with three neighborhoods. For each value of k, we show four states. The first shows the initial solution. The second illustrates the random perturbation of the initial solution with a shake procedure. The third step illustrates the improvement of the solution to reach the local optimum with respect to the three neighborhoods considered. Finally, in the fourth step, the solution obtained in the previous step is compared with the initial solution and the new value k is updated. We can observe that when the k value is 1 the solution is caught at a local minimum and cannot escape there until the k value is set to 4. With the value k set to 4, the method is able to escape from the previous local optimum and reaches a new zone of the solution space. Again, the VND procedure explores with several local search methods the different neighborhoods until the method is caught again at a new local minimum.

1.5 Hypothesis and objectives

In this section, we propose our hypothesis from an earlier overhaul of the state of the art. The hypothesis is the basis for the research performed in this Doctoral Thesis and was used to establish the main objectives to achieve.

After a preview review of the state of the art, we identified the possibility of improving the process that involves the different variants of the online order batching problems in logistic warehouses. Particularly, we observed that order picking is one of the operational processes that has a greater impact on the performance of logistic warehouses in supply chain management [257]. The order-picking task involves optimization problems classified as $\mathcal{NP}$-hard, i.e., they cannot be solved with an exact method in a reasonable amount of time for real-size instances. Therefore, we believe that if we apply new heuristic methods to solve these problems, they could be solved more efficiently compared to the previous proposals in the state of the art. The most realistic and efficient process for order picking is known as online order batching. At this point, once we have identified the problem to be solved, we formulate a hypothesis based on the preview analysis.

Hypothesis: "*Order picking systems with dynamic arrival of orders and batching picking policy, can be improved with the use of new heuristic algorithms combined with the latest advances in metaheuristics, to obtain high-quality solutions in a short amount of time.*"

The main objective of this Doctoral Thesis is to design new heuristic algorithms to improve the state of the art for each of the variants of the Online Order Batching Problem addressed. To that aim, we have identified three optimization problems to study in this Doctoral Thesis: the online order batching problem with a single picker, reviewed in Chapter 4, the online order batching problem with multiple pickers, reviewed in Chapter 5, and the online order batching problem with time window, reviewed in Chapter 6.

The specific objectives to achieve the main objective of this Doctoral Thesis for each variant of the problem studied are detailed next.

» **Review and analyze the current state of the art of the studied problems.** First, we need to study the current state of the art of the identified problems and, specifically, the different techniques and algorithms used to solve the variant of the problem discussed.

» **Classify and position the problems identified in the literature.** We need to define unified criteria to classify the problems studied in relation to other problems of the same family in the literature.

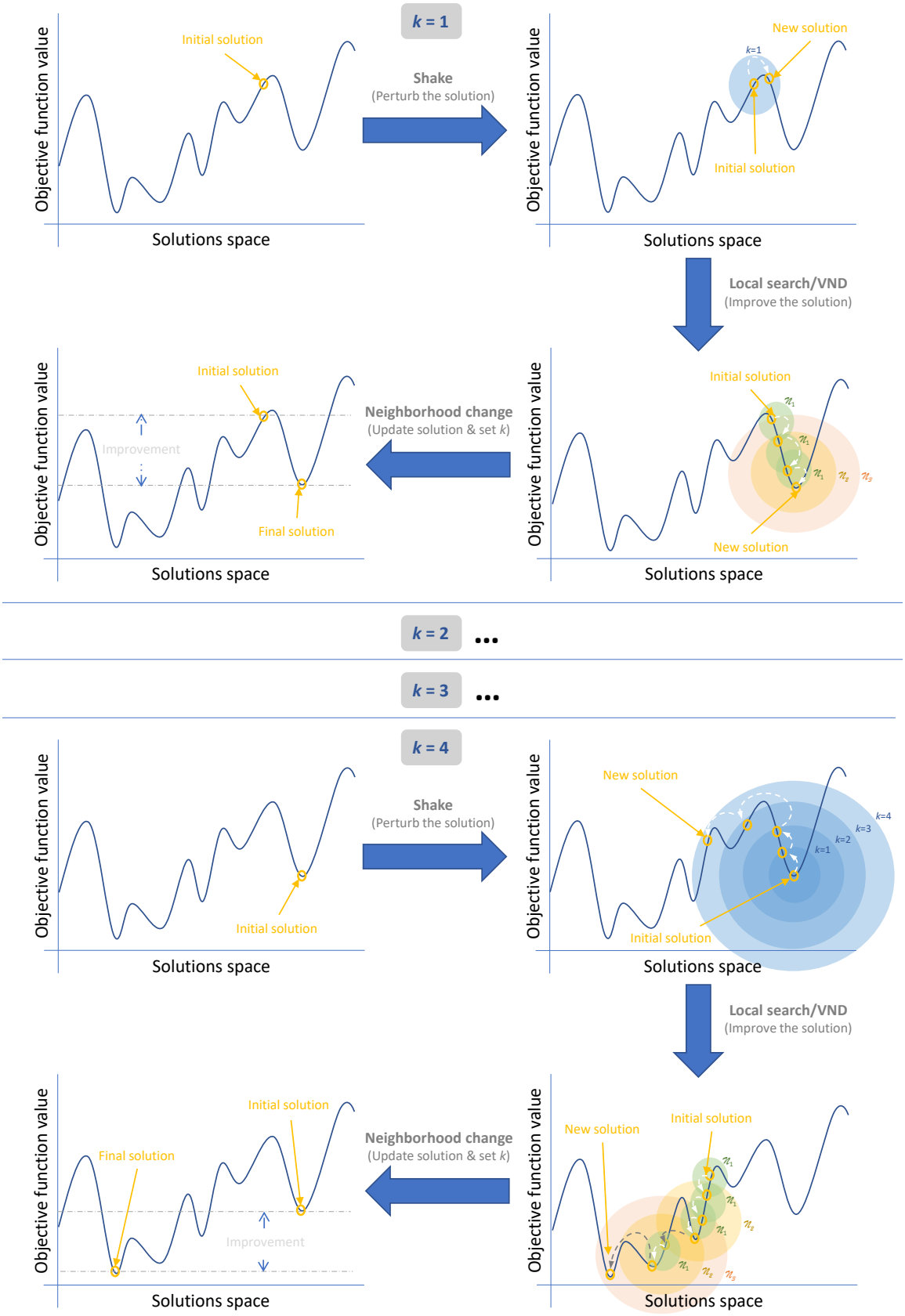


Figure 1.4: Illustration of several steps within a GVNS process.

» **Study and implement the most relevant algorithms in the current state of the art.** For comparison purposes, we need to obtain or implement any relevant previous

algorithm for each variant of the problem studied.

» **Propose, design and implement heuristic algorithms to tackle each problem.** For each variant, we need to propose and develop specific heuristic algorithms to tackle each task that must be handled within each problem.

» **Identify the most suitable metaheuristic schemes for the problem.** For each variant, the proposed heuristic needs to be combined within metaheuristic algorithms. For that purpose, we need to study the most suitable ones, depending on the characteristics of the problem.

» **Validate the algorithms created for each variant.** Once we have implemented all the algorithms proposed to solve the problem, we need to verify its performance with instances previously used in the literature.

» **Configure the parameters of developed algorithms.** The algorithms developed need to be configured in order to use the most suitable parameters to obtain the best possible results for the problem. To configure the parameters, it is necessary to perform some preliminary experiments.

» **Compare experimentally our proposals with previous algorithms.** Each algorithm proposed for any variant of the family of problems studied must be compared with previous algorithms in the state of the art for the problem.

» **Write and publish the results achieved.** The results obtained in the experiments must be disseminated by the publication of articles in journals and conferences.

Throughout this dissertation, we present the findings reached in this Doctoral Thesis to affirmatively respond to the previous hypothesis. Also, at the end of the document, we analyze the achievement of the proposed objectives.

1.6 Structure of the memory

In this dissertation, we detail the research carried out in this Doctoral Thesis, on the different variants of the Online Order Batching Problem. First, we present from a general point of view, the context in which the family of optimization problems is tackled, as well as the scientific methodology followed, and the main algorithms applied to tackle the problems. Then, we describe in detail the optimization problems tacked. Next, we review the state of the art of problems within this family, detailing the characteristics of each variant addressed and the results obtained. We continue by including a chapter for each of the three variants of the problem considered. Finally, we present the conclusions of each variant and the general conclusion of the Doctoral Thesis. In addition to this small previous summary, we structure this memory in the following chapters.

» **Chapter 1.** Introduces the concept of optimization problem in general and the optimization problems that occur within logistic warehouses. In addition, it describes the motivation to realize this Doctoral Thesis, as well as the hypothesis proposed in this work, the methodology, and the objectives achieved.

» **Chapter 2.** Presents the family of Online Order Batching Problems addressed in this Doctoral Thesis. Details the characteristics of each variant studied, as well as each of the tasks which are needed to be solved, to tackle the different OOBP variants. Also, it review the instances and main objective functions used in the context of the problem.

» **Chapter 3.** Addresses the state-of-the-art of OBP. It presents an article published that details the state of the art of the whole family of problems. Also, in this article, we present a new taxonomy of the problem and review all the factors and tasks involved in OBP. In addition, it contains all the bibliographic information about the published article.

» **Chapter 4.** It contains two associated publications related to the Online Order Batching Problem with a Single Picker. In addition, it presents a summary of each publication as well as all bibliographic information about the articles published.

» **Chapter 5.** It contains two associated publications related to the Online Order Batching Problem with Multiple Pickers. In addition, it presents a summary of each publication as well as all bibliographic information about the articles published.

» **Chapter 6.** It contains two associated publications related to the Online Order Batching Problem with Time Window. In addition, it presents a summary of each publication as well as all bibliographic information about the articles published.

» **Chapter 7.** Presents general conclusions about this Doctoral Thesis, as well as a particular conclusion about each problem addressed in this work, future works, and a summary of the publications obtained in this Doctoral Thesis.

Chapter 2

Online Order Batching Problem

Order Batching Problems are a family of optimization problems that belong to the operational level of warehouse management. In particular, this family compiles all problems which consist of determining an efficient picking operation when it follows the batching policy (i.e., orders are grouped in batches before being picked). These problems are characterized by several tasks that need to be addressed in order to solve them. The most common tasks considered when dealing with problems within this family are batching and routing tasks. The batching task consists of generating groups of orders (i.e., batches) of a maximum predefined size. On the other hand, the routing task consists of generating a route in the warehouse to collect the products assigned to a batch in such a way that all products from the same batch are collected in a single tour by the same picker. Other well-known tasks that are frequently studied are: sequencing, assigning, waiting, or sorting, among others. In Figure 2.1 we illustrate the sequence of tasks involved in order batching problems.

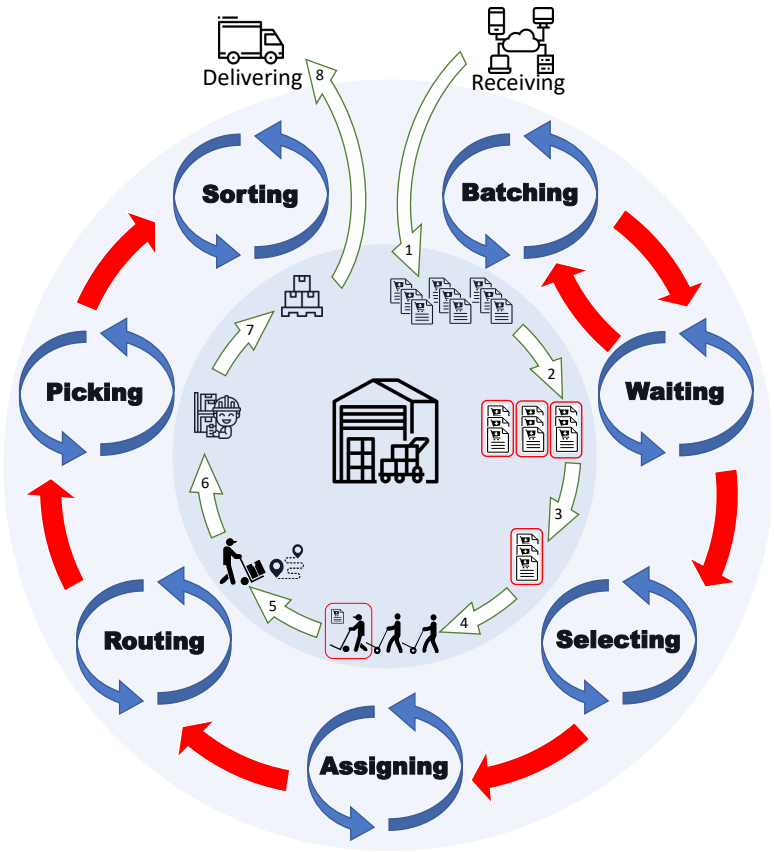


Figure 2.1: Sequence of tasks involved in order batching problems.

In addition to following an order-batching policy, any variant of the OBP must satisfy certain constraints, which might vary depending on the particular variant of the problem tackled. However, some general constraints are: batches cannot exceed certain limits of capacity, volume, or weight; orders cannot be divided in different batches; a batch cannot be modified when the picking tour has started. Other characteristics considered in the variants addressed in this Doctoral Thesis are:

- The sorting policy followed, if not specified, is the sort-while-pick policy. This means that the products are sorted in its corresponding order while the picking is performed. For instance, this can be done by carrying different bins in the picking cart.
- The picking process is picker-to-parts. This means that order pickers travel through the picking zone to collect all items from each batch.
- Customer order collection is done manually by the order pickers. However, sometimes pickers are assisted by automated systems, such as mechanical picking carts.
- Order pickers maintain a constant speed when traveling through the picking zone.
- Extracting each item from a picking position on the route takes a constant time.
- The depot operations at the beginning and ending of the picking route take a constant time.
- The warehouse layout is rectangular, and there is only one block and one depot.
- The picking routes start and end at the depot, which is placed on a crossing aisle, either in the leftmost corner or in the center of the aisle.

The OBP family is quite wide; we can find many variants of the problem. If we review the classification published in [94], we can find two large groups, based on the availability of information on the order to collect: offline variants and online variants. Offline variants are characterized by all the orders must be known before starting the collected process. Offline variants are the most well known and studied in the literature. In the other case, online variants are characterized by a lack of information about some orders that are not known before starting the collection process because they have not arrived at the system yet. Therefore, online variants are more difficult to solve. Furthermore, offline variants can be considered as a particular case of online variants. In addition, offline and online variants can be classified in turn by the number of order pickers used in the picking process into: single-picker variants and multiple-picker variants. In this case, single-picker variants can be considered as a particular case of multiple-picker variants. This classification is based on the extra constraints that are necessary to model the problem when it is online or with multiple pickers. In both cases, we need to include additional constraints in the model. In this Doctoral Thesis, we focus on different online versions of the problem.

When dealing with order-picking problems, we can look for the optimization of different and varying objectives. Some of the objectives of the problem can occur only in determinate variants. In Section 2.3 of this chapter, we study the main objectives functions, as well as their application context according to the classification previously proposed. Also, according to this classification and the objective function used, we find different numbers and types of tasks to solve. In the next section, we detail each task, the context in which we need to solve them, and the strategies followed by the proposals in the literature to address them. In addition, we detail the process to synchronize all tasks involved in the order picking.

2.1 Processes involved in Online Order Batching Problem

In this Doctoral Thesis, we address several variants of the OOBP with a single picker and with multiple pickers. As we have previously seen, the online variants do not have information on the orders that will be collected until each order arrives at the system. This fact means that the problem is closer to real order picking scenarios than offline variants. But the problem is also more complex to solve, and it is necessary to have an algorithm to synchronize all tasks when the order-picking process is simulated. The algorithm to synchronize the tasks involved in the process is also known as the orchestration algorithm. All these tasks and the orchestration algorithm are described in the following sections.

2.1.1 Orchestration algorithm

When we simulate the order picking process in a logistic warehouse, we have to synchronize the tasks involved in the process (Batching, Routing, Selecting, Sequencing, Assigning, Waiting, and Sorting). Therefore, we need a method to coordinate all these activities. This method is called the orchestration or synchronization algorithm. This algorithm is different depending on the variant of the problem solved because, as we previously reviewed, each variant involves a different number of tasks to handle. In Figure 1 in [93] (attached in Section 5.2) compiles a general activity diagram with all tasks involved in OOBP. This diagram represents the fundamentals of the different orchestration algorithms used in our publications. Next, we detail the pseudocodes designed for each variant of the problem addressed. In these pseudocodes, we can observe the sequence in which the tasks are sorted, but here we do not explore the different tasks, which are detailed in the following sections. It is important to note that since we are handling online versions of the OBP, the real systems are continuously running, but, for simulation purposes, we need to observe a particular time horizon. In this case, we have observed the arrival of orders in a chunk of 4 hours and its corresponding picking.

Orchestration algorithm for OOBP with a single picker

The pseudocode of the orchestration algorithm for OOBP with a single picker is presented in Algorithm 4 and was used in [89, 92]. This algorithm only considers batching, routing, and selecting tasks. The assignment task here makes no sense because there is only one picker. The waiting policy that we follow in this algorithm is the no-wait policy, which means that the picker leaves to collect the next batch as soon as he/she is available and there is at least one batch pending.

This orchestration Algorithm 4 receives two input parameters: the time horizon for the reception of orders ($maxTime$) and the list of pending orders to collect ($listOrders$) when the process starts. Notice that this list of orders contains the orders pending to be collected from the previous working day. Notice that in this Doctoral Thesis, we consider that all the orders of the previous day have already been collected. The process starts by initializing several data structures: $pendingOrd$ contains the orders arrived at the system but not collected yet; $partialSol$, contains a partial temporary solution (i.e., a list of batches generated with the orders in $pendingOrd$) updated by `batchingAlgorithm()` in the following iterations; and $bestPartialSol$, which contains the best $partialSol$ found with the orders in $pendingOrd$. Then, the algorithm enters into a loop (step 5) executed while the maximum execution time has not been reached and there are orders in the list of pending orders. Within the loop, the algorithm checks if there are new orders available and updates the list of pending orders (step 7). Then, the batching algorithm is executed (step 9) and the best partial solution found is updated (step 10), if needed. If there is a picker available (step 11) the most suitable batch of the best partial

solution (*bestSolution*) is selected (step 12) and the routing algorithm generates a route to collect the batch (step 13). Then the picker will collect all orders within the selected batch (step 14). Finally, the list of pending orders and the best partial solution are updated by removing the collected orders (step 15). The loop is repeated until the stop conditions are met. Note that the algorithm does not wait until the picker returns from its route but is continuously running to have the best possible solution with the newly arrived orders available as soon as the picker becomes available again.

Algorithm 4 Orchestration method for OOBP with a single picker

```
1: Procedure Orchestration(maxTime, listOrders)
2: pendingOrders ← listOrders
3: partialSol ← ∅
4: bestPartialSol ← ∅
5: do
6:     if (getCPUTime() < maxTime) then
7:         pendingOrders ← pendingOrders ∪ getNewOrders()
8:     end if
9:     partialSol ← batchingAlgorithm(pendingOrders)
10:    bestPartialSol ← update(bestPartialSol, partialSol)
11:    if isPickerAvailable() then
12:        batch ← selectBatchAlgorithm(bestPartialSol)
13:        route ← routingAlgorithm(batch)
14:        collect(batch, route)
15:        remove(bestPartialSol, pendingOrders, batch)
16:    end if
17: while (getCPUTime() < maxTime) || (pendingOrders ≠ ∅)
```

Orchestration algorithm for the OOBP with multiple pickers

The pseudocode of the orchestration algorithm for OOBP with multiple pickers is presented in Algorithm 5 and was used in [90, 93]. This new algorithm is an improved version of Algorithm 4 because it also includes the assigning task, to decide which picker collects each batch, and the waiting task, to decide the best time for a picker to start a new collection route.

The orchestration algorithm presented in Algorithm 5 receives two input parameters: the time horizon for the reception of orders (*maxTime*) and the list of pending orders to collect (*listOrders*) when the process starts. Notice that this list of orders is the orders pending to be collected from the previous working day, but, in this Doctoral Thesis, we consider that all the orders of the previous day have already been collected. Again, the method starts by initializing several data structures: *pendingOrd* contains the orders that arrived in the system but were not collected yet; *partialSol*, contains a partial temporary solution (i.e., a list of batches generated with the orders in *pendingOrd*) updated by `batchingAlgorithm()` in the following iterations; and *bestPartialSol*, which contains the best *partialSol* found with the orders in *pendingOrd*. The algorithm then enters the main loop (step 5), which is executed until the maximum execution time is reached, and the list of orders pending to be collected becomes empty. The main loop contains two inner loops. The first loop is run until `waitingAlgorithm()` decides that *bestPartialSol* is in accordance with some quality indicator, so the next batch can be collected (step 12). In this loop, first, it checks if new orders have reached the system to update the list of pending orders (step 8). Then, the batching algorithm is run (step 10) and the best partial solution found is updated (step 11) if needed. In the second loop (step 14),

for each available picker, the most suitable batch of the best partial solution (*bestSolution*) found is chosen by the selecting procedure (step 15). Then, the routing algorithm generates a route to collect that batch (step 16), and the assignment to the picker is performed (step 17). The assigning algorithm decides whether the *picker* and the *batch* selected for the generated *route* are suitable for continuing the process. Note that depending on the objective pursued, the assignment might not be straightforward (i.e., a currently available picker can be the one with the larger workload, and the method might determine to wait until another picker becomes available). Then, the picker collects all orders within the selected batch (step 18). Finally, the list of pending orders and the best partial solution are updated by removing the collected orders (step 19). The main loop is repeated until the stop conditions are met. Note that the algorithm does not wait until the picker returns from its route but is continuously running to have the best possible solution, including the newly arrived orders, when another picker becomes available again.

Algorithm 5 Orchestration method for OOBP with multiple pickers

```
1:  Procedure Orchestration(maxTime, listOrders)
2:  pendingOrd ← listOrders
3:  partialSol ← ∅
4:  bestPartialSol ← ∅
5:  do
6:      do
7:          if (getCPUTime() < maxTime) then
8:              pendingOrders ← pendingOrders ∪ getNewOrders()
9:          end if
10:         partialSol ← batchingAlgorithm(pendingOrd)
11:         bestPartialSol ← update(bestPartialSol, partialSol)
12:     while (waitingAlgorithm(bestPartialSol)
13:
14:     for each picker ∈ getAvailablePickers() do
15:         batch ← selectingAlgorithm(bestPartialSol)
16:         route ← routingAlgorithm(batch)
17:         if assigningAlgorithm(picker, batch, route) then
18:             collect(picker, batch, route)
19:             remove(bestPartialSol, pendingOrd, batch)
20:         end if
21:     end for
22: while (getCPUTime() < maxTime) || (pendingOrders ≠ ∅)
```

Once we know how the tasks are related among them and their execution time process, we can study each task individually. In the following sections, we review the routing, batching, selecting / sequencing, assigning, waiting, and sorting tasks.

2.1.2 Routing task

The generation of routes in a logistic warehouse, known as routing task, consists of finding the best route through the aisles of the warehouse to collect one or more items. In the context of this Doctoral Thesis, to collect all the items in a batch. This task is one of the main tasks that occurs in the order-picking process and appears in any variant of OBP online or offline. According to various authors in the literature [2, 234, 255], this task can be seen as a particular case of the Travel Salesman Problem (TSP) [170, 193]. The TSP is a well-known problem in

the literature, and many different strategies have been used to solve it, including solutions for large instances.

Different approaches have been used to solve the routing task in logistic warehouses. Specifically, we can differentiate among: exact methods, heuristic methods, and metaheuristic methods. We study each of these methods below.

Exact methods

This type of method generates an optimal solution for this task. We have found two different exact approaches in the literature to solve this task with an exact method: a mathematical model solved in a commercial solver (such as CPLEX [1] and GUROBI [2]); and a dynamic programming algorithm.

Among the models solved in the literature with commercial solvers, they can be classified depending on the number of blocks in the warehouse layout that are able to handle. The problem was solved for a single block in [251, 290] and for multiple blocks in [215, 242, 249, 272]. This type of method has two disadvantages: First, sometimes the routes generated are not easy to follow by order pickers, since pickers need to go back to some aisle to pick an item and after returning for the same path, which might result unnatural. Second, the execution time might be too long to be applied in real scenarios. For these reasons, exact methods are not used in practice.

The second type of exact approach is based on dynamic programming. Again, there are versions of the algorithm for a single block [234], and for two blocks [235]. These algorithms generate an optimal solution in less time than commercial solvers running a mathematical model. Then, they can be used together with metaheuristic algorithms to solve the batching task.

In Figure 2.2 we can see an example of a warehouse layout, several picking positions, and an optimal route to collect the items.

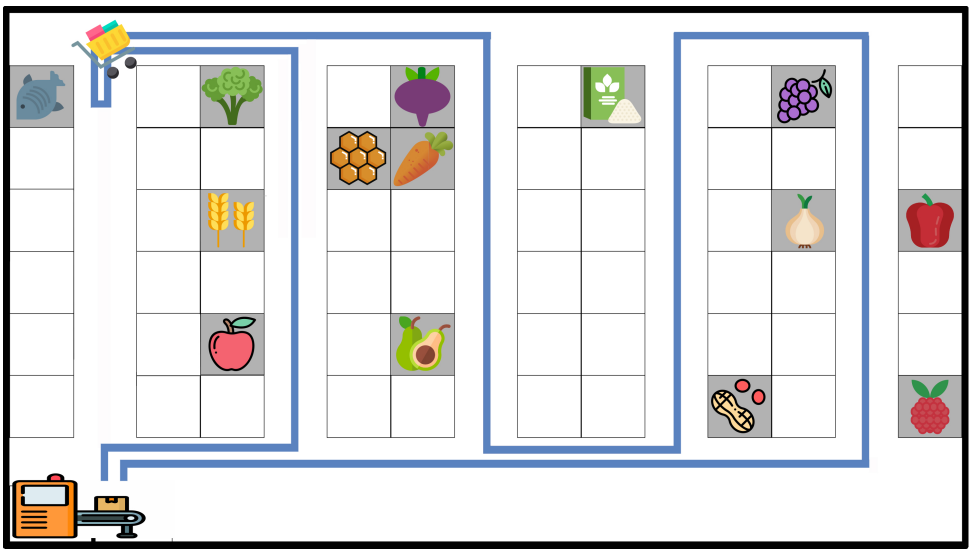


Figure 2.2: Example of an optimal route in a logistic warehouse.

Heuristics methods

This type of method is the most common in the literature related to OBP and its use is very extended. These algorithms are used to generate routes that are easy to follow for order pickers.

[1] https://www.ibm.com/products/ilog-cplex-optimization-studio
[2] https://www.gurobi.com

Additionally, the generated solutions are of very good quality and are fast to compute, so they are usually combined with metaheuristic algorithms used to solve the batching task. Among these methods, we can find S-Shape or traversal, Return, Mid-point, Largest-Gap, Composite, and Combined. Most of these methods were designed for warehouses with only one block [117]. Later, the methods were extended to warehouses with more than one block [237]. There are several articles in which these methods were compared among them [221, 223, 236], and also compared with exact methods. Among the previously described methods, the most used in the literature are: S-shape, Largest-Gap, and Combined, and they have also been used in different articles obtained as the result of this Doctoral Thesis. Next, we review some of the most common heuristic routing methods.

- **S-Shape o traversal:** In this method, the order picker crosses all parallel aisles with items to collect. If the number of aisles with items is odd, the picker has to turn around in the last aisle to go back to the depot, located in the front cross aisle. This method is the most widely used in the literature because of its simplicity and performance. In Figure 2.3, we can see an example of an S-shaped route in a logistic warehouse.

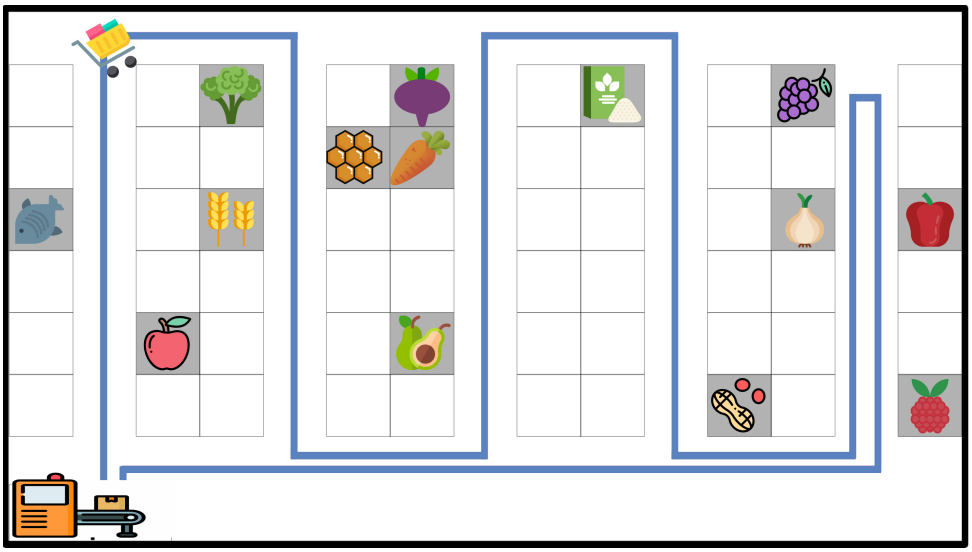


Figure 2.3: Example of an S-shape route in a logistic warehouse.

- **Return:** In this method, the order picker enters from the front cross-aisle into the parallel aisles with items to collect. In each parallel aisle with items, the picker travels through the aisle until the last item to collect, then turns around to come back to the front cross aisle, and so on. If there are no more items to collect in the following parallel aisles, the picker returns to the depot. In Figure 2.4, we can see an example of a return route in a logistic warehouse.

- **Mid-Point:** In this method, the order picker enters in the parallel aisles with items to collect from either the front and the back cross-aisles. The first and last parallel aisles with items are fully traversed to change from one cross aisle to the other. Items in the upper half of the warehouse are first collected from the back cross-aisle, and items from the lower half of the warehouse are collected from the front cross-aisle. In each parallel aisle, the picker turns around before the middle point of the aisle. In Figure 2.5, we can see an example of a Mid-Point route in a logistic warehouse.

- **Largest-Gap:** In this method, orders are collected entering in the parallel aisles from the front or back cross aisles. In any case, pickers avoid traversing the largest gap in the

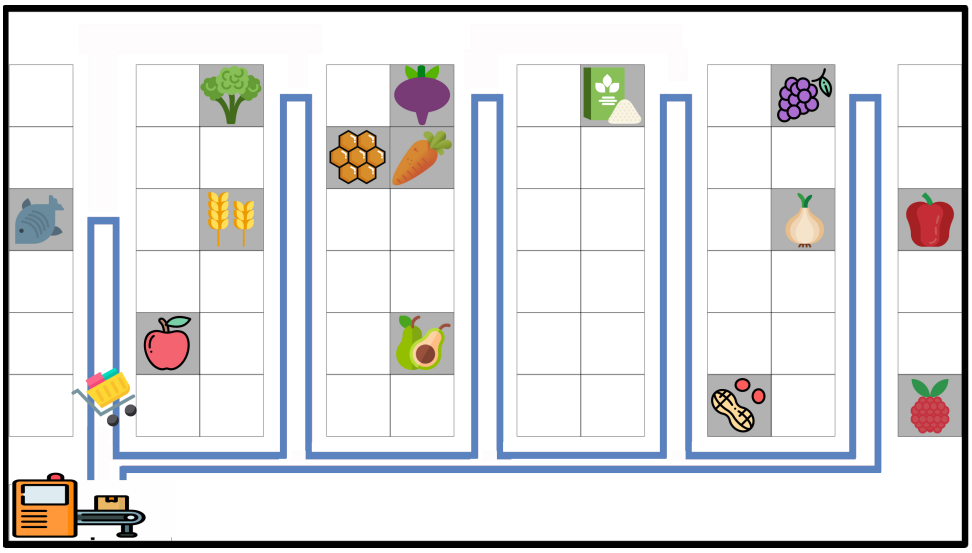


Figure 2.4: Example of an Return route in a logistic warehouse.

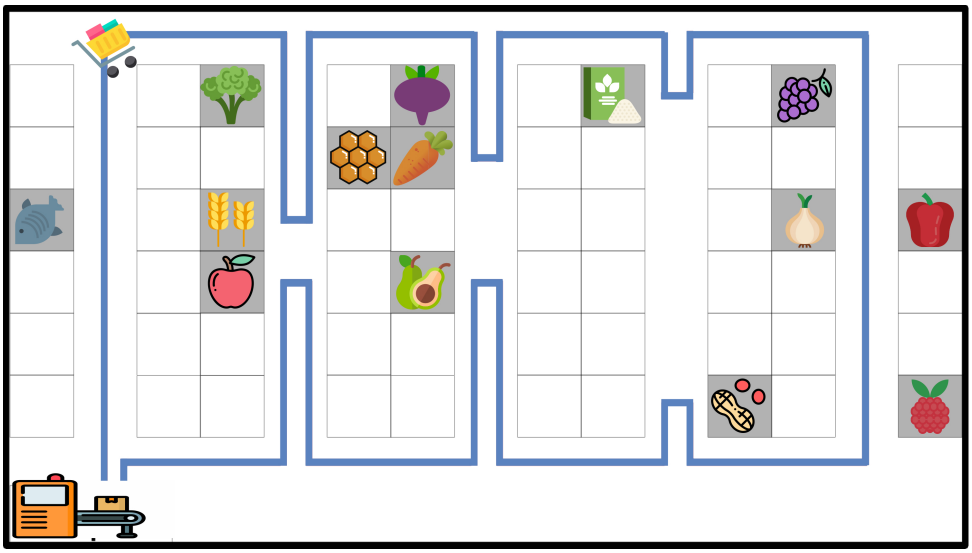


Figure 2.5: Example of an Mid-Point route in a logistic warehouse.

aisle without items to collect. A gap is defined as the distance between two consecutive items to collect or the distance between the first item to collect and its nearest cross-aisle. The first and last parallel aisles with items to collect are fully traversed to change from one cross aisle to another. The pickers turn around when they reach the point of the aisle where the largest gap starts. In Figure 2.6, we can see an example of a Largest-Gap route in a logistic warehouse.

- **Composite:** This method is a combination of the S-Shape and Return methods. For each aisle with items to collect, the algorithm decides which is the best method to use between Shape and Return. This method was developed in [24, 171, 220]. In Figure 2.7, we can see an example of a Composite route in a logistic warehouse.

- **Combined:** This method is a combination of the S-Shape and Largest-Gap methods. For each aisle with items to collect, the algorithm decides which is the best method to use. This method was developed in [51]. Later, this method was improved in [189]. In addition, this method is the best routing strategy among the heuristic ones. In Figure 2.8, we can see an example of a Combined route in a logistic warehouse.

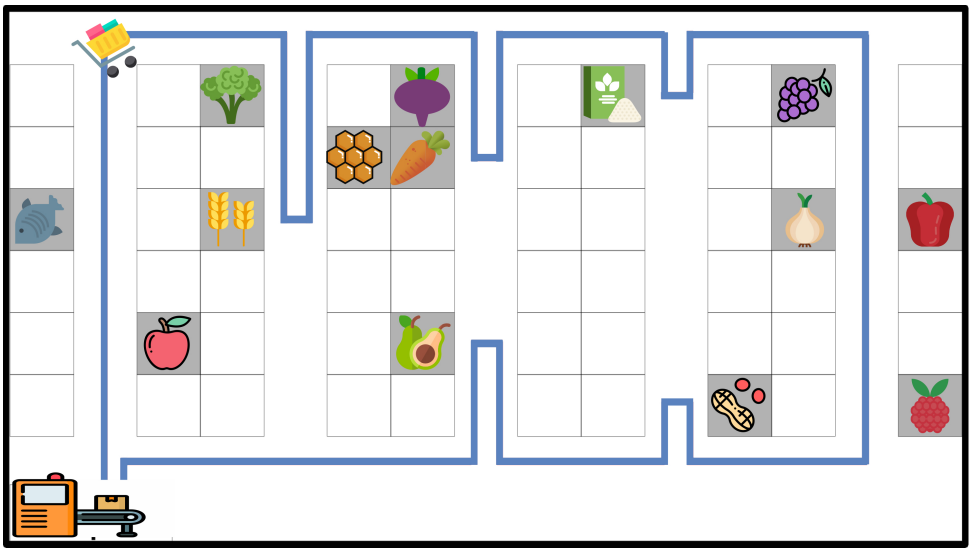


Figure 2.6: Example of an Largest-Gap route in a logistic warehouse.

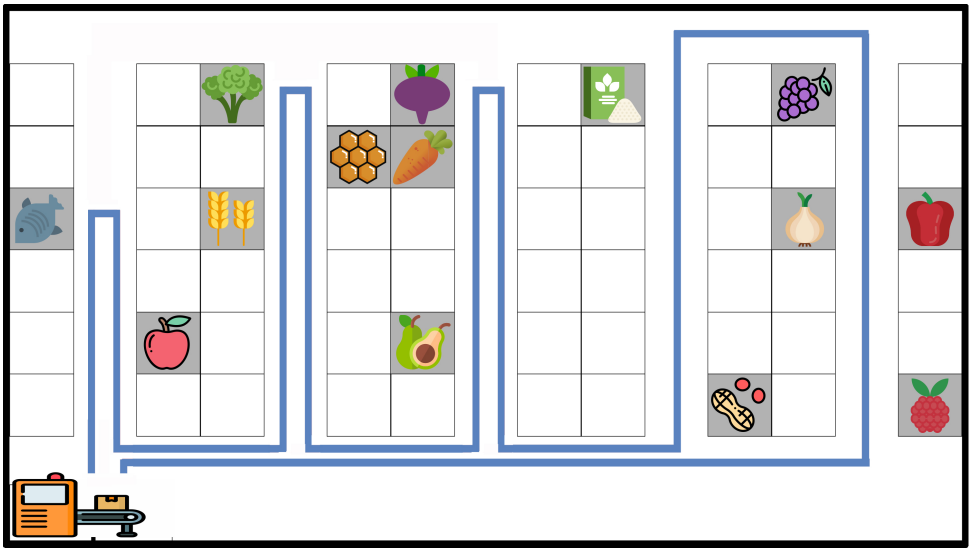


Figure 2.7: Example of an Composite route in a logistic warehouse.

All the heuristic algorithms reviewed above have been defined here for a rectangular-shaped single-block warehouse. However, most of them have an extended version for two or more blocks in the literature. We only reviewed the algorithms for a single block because this Doctoral Thesis only considers this warehouse layout.

Metaheuristic methods

These methods have also been used in the literature to solve the routing task in the context of the OBP. However, their performance presents very few advantages compared to the methods previously presented. On one hand, metaheuristic methods cannot compute, or at least confirm, that a solution is optimal in contrast to exact methods. On the other hand, these methods are slower than simple heuristics, and the quality of the solutions obtained do not use to be better, since the computing time is very short. The main advantage of these methods is that they can be easily adapted to any warehouse layout. Among the metaheuristic algorithms used in the literature, we highlight: Adaptive Large Neighborhood Search [33], Ant Colony Optimization [34, 35, 210], Genetic Algorithms [111, 256, 307], or Particle Swarm Optimization [111].

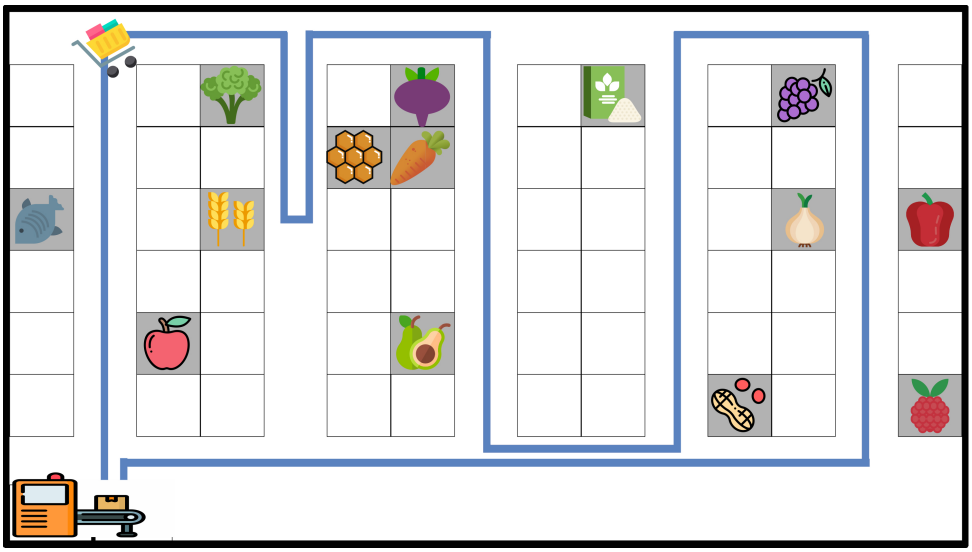


Figure 2.8: Example of an Combined route in a logistic warehouse.

2.1.3 Batching task

The batching task consists of the generation of batches of orders in a logistic warehouse. Specifically, it consists of grouping the orders available in the system in batches without exceeding the maximum predefined capacity. In the variants addressed in this Doctoral Thesis, orders cannot be split in different batches. The batching task is handled in all variants of OBP tackled in this Doctoral Thesis. To solve this task, there are heuristic algorithms, metaheuristic algorithms, and exact algorithms. Next, we review the most outstanding methods from each group.

Heuristic methods

Heuristic methods were the first type of method used to solve this task, but nowadays they are in disuse in favor of metaheuristic methods. Heuristic methods generate solutions very fast, but with a low quality for this task, in comparison with metaheuristics. This type of method is usually deterministic (i.e., for the same instance, always generate the same solution), and currently they are often used as constructive procedures for metaheuristic algorithms. Next, we can see a review of the most relevant heuristic algorithms in the literature.

- **First Come, First Served (FCFS)** [4, 125, 144]: This algorithm is widely used in different contexts and problems. In the order batching context, orders are grouped into batches according to their arrival time. In a compact extension of the algorithm, each new order attempts to fill each batch previously created before creating a new one. FCFS has been extensively used in the context of OBP as a baseline method.

- **Earliest Due Date (EDD)** [25, 279, 312]: This algorithm is also widely used in different contexts and problems. This is very similar to the FCFS algorithm. The difference is that orders have a predefined due date, and they are grouped into batches according to their due date. This algorithm is widely used in variants of the problem in which the objective function is to minimize the delay in the delivery of products.

- **Seed algorithms** [52, 70, 151]: These types of algorithms are widely used in the OBP literature. The algorithms belonging to this family consist of two parts that are repeated until all orders are processed. For each part, a different strategy can be applied. The first

part consists of selecting an order as a seed, and the second part consists of finding the most similar orders to the seed order and adding them to the same batch until the batch capacity is completed. Then, we need to repeat the process, select another seed order for a new batch, and so on.

- **Saving algorithms** [52, 140, 308]: This is a family of greedy algorithms in which orders are added to the batch in order of the evaluation performed over each order, such that the order that produces the largest improvement in the objective function is added next. There are different saving algorithms. The most used Saving algorithm in relation to OBP is the Clarke and Wright (C&W) method [42] and its extensions, despite the fact that it was created for other types of context.

Metaheuristic methods

Metaheuristic methods are the type of method most used for order batching because they generate very good quality solutions in a reasonable amount of time. This type of methods is used to be non-deterministic (i.e., for each execution of the same instance, they generate different solutions), because they usually have a stochastic component. Next, we review the most important metaheuristic algorithms used in the literature for OBP.

- **Iterated Local Search (ILS):** This algorithm was proposed for the first time in 1995 [284], although the algorithm has been studied by several authors [180, 265]. The algorithm is a local-search-based algorithm, which consists of two toggle phases. An improvement phase to reach a local optimum and a perturbation phase to diversify the search. The ILS algorithm has been used to solve the OBP in [125, 130, 229, 288].

- **Variable Neighborhood Search (VNS):** This algorithm was proposed in 1997 [197] and uses an exploration strategy based on changes in the neighborhood structure to escape from a local optimum. There are a large number of VNS variants based on this principle. All of these variants were studied in detail in a previous section of the methodology (see Section 1.4.2). Different variants of VNS have been used to solve the OBP [4, 90, 126, 252].

- **Greedy Randomized Adaptive Search Procedure (GRASP):** This algorithm was published in 1995 [76] as a multi-start algorithm. Each repetition of the loop is composed of two phases: a construction phase and an improvement phase. The construction phase is the core of GRASP and is based on a greedy heuristic component and a random component. This algorithm was studied in detail in a previous section of the methodology (see Section 1.4.2). The GRASP algorithm has been used to solve the OBP in [49, 92, 93, 269]

- **Genetic Algorithm (GA):** This algorithm was published in 1992 [138] and was later extended in 1994 [291]. The algorithm is based on the evolution of a population of individuals subjected to certain conditions of stochastic biological nature. The main operators of this type of algorithms are the mutation and genetic recombination. There are different variants of GA published in the literature that are used to solve OBP [146, 160, 214, 213].

- **Ant Colony Optimization (ACO):** This algorithm was published in 1991 [43, 56] and is classified as a bioinspired algorithm, since it is based on the natural behavior of ants when looking for food. When the day starts, the ants move randomly looking for food. In each movement, the ants leave a trail of pheromones that attract other ants to the surrounding area, but obtain the shortest route to find the food. Following these principles, the algorithm toggles between two phases: the diversification phase and the

intensification phase. There are different variants of ACO published and used to solve the OBP [6, 38, 130, 229].

- **Tabu Search (TS):** This algorithm was published in 1989 [104, 105, 108] and is based on memory structures to remember solutions previously explored. In addition, it uses a local search to improve the solution and perturbations of the solution to escape from the local optimum. There are different variants of TS published and used to solve the OBP [129, 165, 189, 219, 276, 308].

In addition to the metaheuristics that we have mentioned above, there are other metaheuristics in the literature that have been used to solve different variants of problems related to OBP. In this Doctoral Thesis, we used variants of the VNS y GRASP methodologies to tackle OOBP. To get more details about this methodology, visit Section 1.4.2.

Exact methods

This type of method requires a lot of time to complete and obtain the optimal solution. Due to the fact that the OBP family is known to be a $\mathcal{NP}$-Hard problem, the situation allows the method to be used only in small instances (approximately up to 20 orders). To solve these problems with exact methods, there are several mathematical models in the literature. These models are implemented as mixed-integer linear programming models. In addition, these are solved in the literature with commercial solvers such as CPLEX and GUROBI. The implementation of these models is varied; some of them simultaneously solve the routing and batching task [29, 272, 175]. In other cases, batching is solved optimally, but routing is solved following a well-known routing heuristic (e.g., S-Shape, Largest-Gap, among others)[204, 205].

In addition, there are other algorithms published in the literature that solve OBP optimally. Among them, we can highlight a Branch-and-Bound Pricing, based on a linear relaxation at each node of a branch-and-bound tree solved by Column Generation. It is used in [84, 276]; Branch-and-Cut, based on cutting planes, to improve the linear relaxation of the search tree. It is used in [21, 275]; Column Generation, based on linear relaxation in a small group of variables or columns to solve the problem. It is used in [21, 84, 202]; or the Markov Decision Process (MDP), based on an extension of Markov chains in which the difference is the addition of actions and rewards to the method. It is used in [23].

2.1.4 Selecting/sequencing task

Selecting and Sequencing are two different tasks, but closely related. Selecting consists of choosing the next batch to be collected among a set of batches ready to go. On the other hand, sequencing consists of finding an order (a priority) of all batches available. The use of selecting or sequencing tasks depends on the variant of the problem tacked and its objective function. Sometimes these tasks do not need to be addressed.

In the literature, simple heuristics have been used to address the selecting task. Among these heuristics, we can highlight the following:

- **FIRST:** This heuristic selects the batch that contains the order with the shortest arrival time. It is used in [125, 304].
- **SHORT:** This heuristic selects the batch that has the shortest service time. It is used in [125].
- **LONG:** This heuristic selects the batch that has the longest service time. It is used in [125].

- **SAV:** This heuristic selects the batch that contains the order with the largest savings with respect to collect the associated orders individually. It is used in [125].
- **FULL+SHORT:** This heuristic selects the fullest. If there are more than one, the ties are broken by selecting the batch with the shortest service time. This heuristic is used in this Doctoral Thesis in [92, 93].

The sequencing task is commonly used when the objective function is related to the due date of the orders, such as minimizing the tardiness or minimizing the earliness. In the literature, we can find several methodologies to tackle this task. We highlight heuristics such as the Earliest Due Date algorithm (EDD) used in [12, 312] or Earliest Start Date algorithm (ESD) used in [126, 252]; metaheuristics, such as the Genetic Algorithm used in [12]; in other cases, this task is addressed together with batching task using a metaheuristic algorithm. We can find this approach in [127, 188, 255].

2.1.5 Assigning task

This task consists of assigning each selected batch to a picker. We find only this task in variants of the problem with multiple pickers. This task mainly affects the balance of the workload among the pickers, but also influences the completion time, because a better distribution of the balance of the workload improves this objective function [93]. To solve this task, we use simple heuristics, such as the first available picker or the picker with a lower workload [90, 127, 304]. In other cases, this task is addressed together with the batching task using a metaheuristic algorithm. We can find this approach in [6, 190, 279, 252].

2.1.6 Waiting task

This task consists of deciding how long the picker should wait for the arrival of new orders in the system, which could improve the current distribution of orders in batches. Although waiting may seem unnatural, it has two main benefits. While waiting, new orders arrive at the system, and it is possible to improve batch quality. Additionally, metaheuristic algorithms run longer, which leads to better results. This task can only be found in the online variants. In the literature, this task is also named the task of determining the time window, and initially it was proposed as a batching method which grouped in the same batch the orders arriving in a particular time window. In Chapter 6, we review this task in detail and how it affects several objective functions. This waiting task is also presented in [95] in an OOBP context.

For a better understanding of this task, we can consult Figure 2 in the article [91] (attached to this Doctoral Thesis). This figure represents a timeline where each important timestamp is recorded (new orders received, new batches generated, and picker availability).

To solve this task, different heuristic methods are used in the literature. These heuristics are classified into two categories: fixed-time window methods, which is based on waiting a fixed amount of time ([91, 95, 304]); and Variable Time Window, which is based on assigning a variable waiting time depending on the context. This waiting time for each batch is calculated using different methods, such as arrival times, service times, and the number of orders or batches currently in the system. These methods have been used in [91, 95, 125].

2.1.7 Sorting task

Using a batching policy implies that several orders might be collected at the same time. Therefore, items belonging to different orders are retrieved in the same picking tour and, at some point,

they have to be separated in different orders. The sorting task consists of classifying the collected items into different orders. There are two main policies to solve this task in the literature: the sort-while-pick policy and the pick-and-sort policy. The Sort-while-pick policy consists of performing the sorting task during the picking process. In this case, the picker carries a sorting cart with different bins for each order. In Figure 2.9, we can see different types of commercial sorting carts from several companies. Among others, this sorting policy is used in [92, 130, 5]. On the other hand, the pick-and-sort policy consists of performing the sorting task after the picking task has been completed. In this case, all collected items belonging to the same batch are placed in a specific zone of the warehouse to perform the sorting. Among others, we can find this sorting policy in [282, 177, 151]. Using this sorting policy does not affect the collection process, but it affects by the value of some objective functions. These sorting policies are compared in [281], in which we can see the advantages and disadvantages of each policy. The sorting policy that we use in this Doctoral Thesis is the sort-while-pick.

(a)
IRSG
SKU:$S3.S2448LZ.001$[1]

(b)
LLM Handling Equipment Ltd.
SKU:$CWSTB.D.CD$[2]

(c)
Equip4work Ltd.
SKU:$GP760$[3]

Figure 2.9: Different types of commercial sorting cards from several companies.

2.2 Joint variants of the Online Order Batching Problem

In recent years, we have observed that there are many related variants of the problem in the literature, denoted as "Joint Online Order Batching". This term is used when the problem being addressed optimizes several tasks at the same time, further than just the batching task [94]. These tasks can be solved together or separately, but solving these tasks together might result in a better result. In the case of batching and routing tasks, it has been studied that if they are treated together, they can lead to savings of up to 35% of the total cost [52]. We can find different joint variants in the literature depending on the tasks handled: batching and routing [275], batching and sequencing [38], batching and assigning [279], or even batching, sequencing, and assigning [252].

[1] https://irsg.com
[2] https://www.llmhandling.co.uk
[3] https://www.equip4work.co.uk

2.3 Objective functions

When we address a problem that belongs to the OBP family, we can search for the optimization of several objectives. In this section, we review the main objective functions proposed in the literature. To start, we review the different timestamps used in the OBP to understand some of the objective functions addressed. In Figure 2.10, we have compiled different timestamps that occur during the picking process. Also, we have highlighted the relevant time periods that occur between some pairs of timestamps, which are needed to define the objective functions of the problem. Among them, we can find:

- The **waiting time** value of an order is the time that has elapsed since the order arrived in the system until the picker started collecting the batch that contains the order.
- The **service time** of a batch is the time required for a batch to be collected. The time count starts when the batch is assigned to a picker and ends when the batch is delivered to the warehouse depot. This time consists of setup time, routing time, extraction time, and blocking time.
- The **turnover time** for an order is the total time that the order remains in the system before it is handled. In this research, the time counts start when the order arrives at the system and end when the batch that contains the order is delivered to the warehouse depot. This time includes the waiting time of the order and the service time of the batch that contains the order. It is important to note that, in some other contexts, the turnover time might also include the time needed to package the items and the time that the package is awaiting for transportation before leaving the facilities.
- The **tardiness** for an order is the time that has elapsed since the due date of the order is reached until the order batch is delivered to the warehouse depot. This period of time does not appear in all cases, since not all orders have a dude date associated.
- The **earliness** occurs when an order is delivered to the depot before its due date. It consists of the time that has elapsed since the order was delivered up to its due date. Again, this period does not appear in any case, since not all orders include a due date associated.
- The **time window** for a picker is the time since the picker is available to collect a new batch until the picker actually starts collecting it. A detailed description of this period can be found in Figure 2 of the article [91] (attatched in this Doctoral Thesis in Chapter 6.
- The **completion time** of the OBP process is the time since the system was started and the pickers are available until the last batch is delivered to the warehouse depot.

Once the previously introduced periods of time are known, we review the main objective functions studied in the literature.

- **Minimize the Picking Time:** Minimize the sum of service times for all processed orders. We can find this objective function in [4, 69, 91, 187]. In the literature, the picking time is also known as the retrieval time [141, 241], the travel time [134], or the lead time [85].
- **Minimize the Maximum Retrieving Time:** Minimize the maximum service time for all processed orders. This objective function is related to the balance of workload among batches. We can find this objective function in [85, 190].

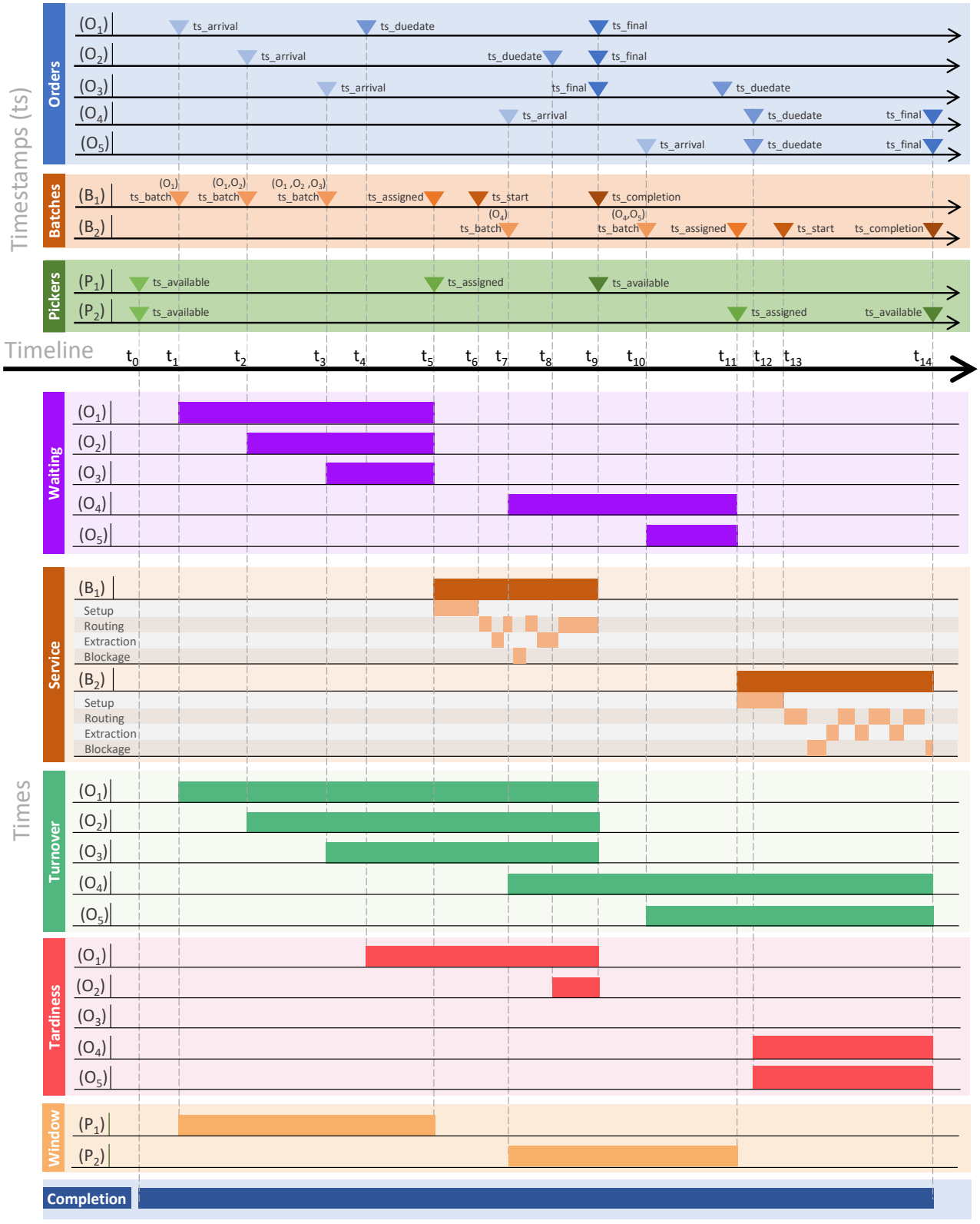


Figure 2.10: Timeline with the different timestamps that we can find in the OBP.

- **Minimize the Travel Distance:** Minimize the sum of the distance required to collect all batches in the warehouse. Some authors consider that the distance traveled is closely related to the picking time [129, 150]. We can find this objective function in [19, 53, 130, 146]
- **Minimize the Completion Time:** Minimize the time since the system was started and

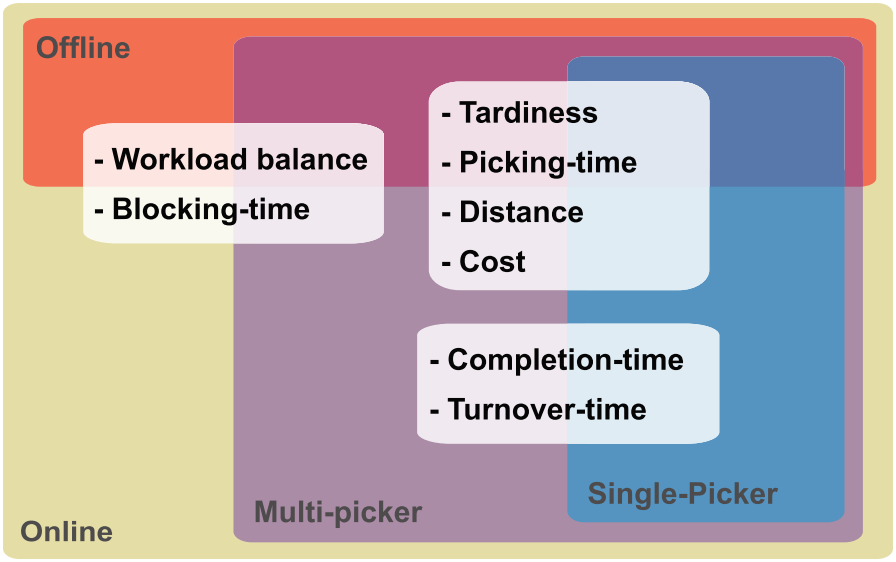


Figure 2.11: Relationship between objective functions and OBP variant classification.

the pickers are available until the last batch is delivered to the warehouse depot. In the literature, it is also defined as minimizing the maximum completion time of the last batch. Sometimes, the completion time is also known as the makespan [6, 151, 243]. We can find this objective function in [87, 125, 175, 304].

- **Minimize the Average/Maximum Turnover Time:** Minimize the average or maximum turnover time that an order remains in the system. This time starts when the order arrives to the system and ends when the order is delivered to the depot. In the literature, the turnover time is also known as the throughput time [172, 248, 281]. We can find this objective function in [40, 92, 137, 266].

- **Minimize the Sum of Tardiness:** Minimize the sum of delays for each order. This objective function can only be found in problems where orders have a due date associated. We can find this objective function in [38, 126, 252, 188].

- **Minimize the Sum of Earliness:** Minimize the sum of all the time that each order has been delivered in advance with respect to its due date. This objective function can only be found in problems where orders have a due date associated. We can find this objective function in [29, 72, 271].

- **Minimize the sum of the Costs:** The costs related to OBP are variable. We can apply a cost to different activities involved in the problem, such as the number of pickers, the travel time or distance, the delay or advance in the deliveries, the number of batches, etc. It is even possible to consider a combination of the previous ones. This objective function minimizes the sum of one or more of these costs. We can find this objective function in [225, 254, 268, 305].

- **Minimize the difference in the Workload Balance:** This objective function balances the workload among all pickers. We can only find this objective function in variants with multiple pickers. The workload can be understood, such as the number of items, the distance/time traveled, or the number of processed batches, among others. We can find this objective function in [41, 93, 147, 283].

- **Minimize the Congestion:** also known as Minimize the sum of the blocking time. Blocking occurs when several pickers try to access the same resource at the same time. These resources might be a narrow aisle, the position of an item, or the depot. Blocking time is the time that a picker needs to wait to access the resource before another picker leaves it. We can find this objective function in [1, 34, 35, 116].

- Other objective functions: There are more objective functions that can be identified in the literature related to OBP variants; however, they are not as frequent as those previously reviewed. Among them, we can highlight **Minimize the Number of batches** used in [147], **Minimize the Number of pickers** used in [1], or **Minimize the Picking error rate** used in [307].

In addition, in the literature, we find problems in which several objectives are processed at the same time. We can do this through weighted sums of several objectives for a single objective function [29, 63, 306]. But also some authors address the use of two objectives as a multi-objective optimization problem. There are different technical approaches to dealing with this kind of problem. We can find multi-objective approaches related to OBP in [1, 41, 147, 307].

In this Doctoral Thesis, we studied four different single-objective functions: Minimize the picking time, Minimize the completion time, Minimize the maximum turnover time, and Minimize the difference between the picking time of each picker (i.e., the workload balance of the pickers).

To end this section, we have classified how the previous objective functions are related to a variant within the OBP family. In Figure 2.11, we can see the relationship between objective functions and the classification of OBP variants. For example, the completion time and the turnover time make sense only in the online context. On the other hand, the workload balance or the blocking time make sense only in contexts with multiple pickers. Other objective functions, such as tardiness, pick-up time, or cost, are used in any context.

2.4 Mathematical model

Once we have reviewed the different tasks involved in the OBP and their main objective functions, we can study and understand a mathematical model to represent the problem. In this section, we present a new general mathematical model that can be used for each variant of the problem addressed in this Doctoral Thesis. First, in Table 2.1, we present the parameters and variables of the problem. Then, we present the model of the different objective functions tacked. In the following, we present the general constraints of the problem. Finally, we present the specific constraints for each different time-window method.

First, in Equation (2.1) we define the random parameter ar_i. The arrival time of order o_i, which is equal to the sum i times of the random variable Z following an exponential distribution of the parameter λ:

$$ar_i = \sum_{1}^{i} Z, \quad \forall\, i \in \{1, \ldots, n\}, \text{ and } Z \sim EXP(\lambda) \tag{2.1}$$

To define the four objective functions addressed in this Doctoral Thesis, we first need to define some of the times that we reviewed in the previous section. First, in Equation (2.2) the routing time of a batch b_j. The routing time is determined by the routing algorithm and the speed of the picker to travel through the warehouse per unit of time. We choose to model the S-Shape routing algorithm because it is the most widely used in this Doctoral Thesis, but this model can be implemented with any other routing algorithm. The routing algorithm is

Parameters		
n	$\rightarrow$	Number of customer orders received in the system
m	$\rightarrow$	Upper bound of the number of batches (a straightforward value is $m = n$).
l	$\rightarrow$	Number of order pickers.
$v_{routing}$	$\rightarrow$	Routing velocity: number of length units that the picker can traverse in the warehouse per unit of time.
$v_{extraction}$	$\rightarrow$	Number of items that the picker can search and pick per time unit.
t_{setup}	$\rightarrow$	Time that the picker needs to initiate a new route with a new order list and end the route let the collected orders in the depot.
w_i	$\rightarrow$	Number of items of order o_i for $1 \leq i \leq n$.
W	$\rightarrow$	Maximum number of articles that can be included in a batch (device capacity).

Random parameter		
ar_i	$\rightarrow$	Arrival time of order i for $1 \leq i \leq n$.

Variables		
st_j	$\rightarrow$	Start time of batch j for $1 \leq j \leq m$.
x_{ji}	$\rightarrow$	$\begin{cases} 1, & \text{if order } o_i \text{ is assigned to batch } b_j, \\ & for\ 1 \leq i \leq n,\ and\ 1 \leq j \leq m. \\ 0, & \text{otherwise.} \end{cases}$
y_{jk}	$\rightarrow$	$\begin{cases} 1, & \text{if picker } p_k \text{ is assigned to batch } b_j, \\ & for\ 1 \leq k \leq l,\ and\ 1 \leq j \leq m. \\ 0, & \text{otherwise.} \end{cases}$

Table 2.1: Parameters and variables for the General OOBP.

developed as the function $\texttt{dis}(b_j)$ that receives orders in batches b_j and returns the distance traveled to collect these orders. The $\texttt{dis}(b_j)$ function with the S-Shape algorithm is presented at the end of this section.

$$T_{routing}(b_j) = \frac{\texttt{dis}(b_j)}{v_{routing}}, \quad \forall j \in \{1, \ldots, m\}. \tag{2.2}$$

Then, in Equation (2.3) the extraction time of a batch b_j. We consider that w_i is the number of items in the order o_i assigned to b_j, and $v_{extraction}$ is the speed of the picker to search and pick up an item from the warehouse per unit of time.

$$T_{extraction}(b_j) = \sum_{i=1}^{n} \frac{w_i x_{ji}}{v_{extraction}}, \quad \forall j \in \{1, \ldots, m\}. \tag{2.3}$$

In Equation (2.4) the service time of a batch b_j is determined by the sum of the routing time, the extraction time, and the setup time.

$$T_{service}(b_j) = T_{routing}(b_j) + T_{extraction}(b_j) + t_{setup}, \quad \forall j \in \{1, \ldots, m\}. \tag{2.4}$$

Finally, in Equation (2.5) the turnover time of an order o_i is the time in which the order is in the system, starting from the arrival of the order to the system until the order is collected and delivered to the depot.

$$T_{turnover}(o_i) = \sum_{j=1}^{m} \Big((st_j + T_{service}(b_j)) * x_{ji} \Big) - ar_i, \quad \forall\, i \in \{1, \ldots, n\}. \tag{2.5}$$

Once we know the times involved in these objective functions, we can define them.

» **Minimize the sum of picking time of all pickers**:

$$\min \sum_{j=1}^{m} T_{service}(b_j). \tag{2.6}$$

» **Minimize the maximum completion time of the received orders**:

$$\min \max_{j \in \{1,\ldots,m\}} \Big(st_j + T_{service}(b_j) \Big). \tag{2.7}$$

It is worth mentioning that this objective is determined by the moment in which the picker delivers the last batch.

» **Minimize maximum difference between sum of picking time of one order pickers and the average picking time**:

$$\min \Big(\max_{k \in \{1,\ldots,l\}} \sum_{j=1}^{m} y_{jk} * T_{service}(b_j) \Big) - \sum_{j=1}^{m} \frac{T_{service}(b_j)}{m}. \tag{2.8}$$

» **Minimize the maximum turnover time of the received orders**:

$$\min \max_{i \in \{1,\dots,n\}} T_{turnover}(o_i). \tag{2.9}$$

Note that the value of this objective function is determined by the turnover time of the order that remains longer in the system.

Once we know the objective functions addressed in the model, we need to define the constraints of the problem that we present next. The set of feasible solutions is given, in all cases, by the following constraints:

- The constraint in (2.10) guarantees that each order is assigned to a single batch:

$$\sum_{j=1}^{m} x_{ji} = 1, \quad \forall\, i \in \{1,\dots,n\}. \tag{2.10}$$

- The constraint in (2.11) guarantees that each batch is assigned to a single picker:

$$\sum_{k=1}^{l} y_{jk} = 1, \quad \forall\, j \in \{1,\dots,m\}. \tag{2.11}$$

- The constraint in (2.12) guarantees that the maximum capacity of each batch is not exceeded:

$$\sum_{i=1}^{n} w_i * x_{ji} \leq W, \quad \forall\, j \in \{1,\dots,m\}. \tag{2.12}$$

- The constraint in (2.13) guarantees that batch b_j begins to collect once any picker is available:

$$st_j \geq \min_{k \in \{1,\dots,l\}} \max_{s \in \{1,\dots,j-1\}} y_{sk} * \Big(st_s + T_{service}(b_s)\Big), \quad \forall\, j \in \{2,\dots,m\}. \tag{2.13}$$

- The constraint in (2.14) guarantees that the batch b_j starts to collect, once the batch b_{j-1} has started to collect:

$$st_j \geq st_{j-1}, \quad \forall\, j \in \{2,\dots,m\}. \tag{2.14}$$

- The constraint in (2.15) guarantees that the route to collect a batch b_j cannot start before the timestamps (moments in time) when the orders o_i assigned to that batch have arrived at the system:

$$st_j \geq ar_i * x_{ji}, \quad \forall\, i \in \{1,\dots,n\}, \text{ and } \forall\, j \in \{1,\dots,m\}. \tag{2.15}$$

- The constraint in (2.16) guarantees the state of non-negativeness of st_j:

$$st_j \geq 0, \quad \forall j \in \{1,\dots,m\}. \tag{2.16}$$

- The constraint in (2.17) guarantees that the variables x_{ji} are binary:

$$x_{ji} \in \{0,1\}, \quad \forall\, j \in \{1,\dots,m\} \text{ and } \forall\, i \in \{1,\dots,n\}. \tag{2.17}$$

- Finally, the constraint in (2.18) guarantees that the variables y_{jk} are binary:

$$y_{jk} \in \{0,1\}, \quad \forall\, j \in \{1,\dots,m\} \text{ and } \forall\, k \in \{1,\dots,l\}. \tag{2.18}$$

2.4.1 Time-window constraints

As we have seen in the previous section, the time window algorithm defines the moment when a batch b_j begins to be collected by a picker. This moment is the start time of the batch b_j. If we want to optimize the start time of each batch, we do not need to define any more constraints. But if we want to delay the starting time to collect a batch, a time-window algorithm appears. Then, we need to define a specific constraint for this time-window algorithm that replaces the constraint (2.13). Next, we present the constrains for the main time-window algorithm used in this Doctoral Thesis.

First, we need to define TPA_b_j in (2.19), as the fist timestamp of any available picker that can pick up the batch b_j.

$$TPA\_b_j = \min_{k\in\{1,\dots,l\}} \max_{s\in\{1,\dots,j-1\}} y_{sk} * \Big(st_s + T_{service}(b_s)\Big), \quad \forall\, j \in \{2,\dots,m\}. \tag{2.19}$$

Once we have defined TPA_b_j it will be used in the following constraints to define the different time-window algorithms.

» **Constraint for No-Waiting algorithm:** We define the start time st_j of batch b_j as follows.

$$st_j = \min\{\max\{ar_i * x_{ji}\}, TPA\_b_j\}, \quad \forall\, i \in \{1,\dots,n\}, \text{ and } \forall\, j \in \{2,\dots,m\}. \tag{2.20}$$

» **Constraint for Fixed Time Window algorithm:** In this algorithm, we need to define the parameter t^{FTW}. This parameter is the time that we need to wait before starting the picking. We also need to define the variable t_j^{FPA} in Equation (2.21). This variable is the timestamp that indicates that there is a picker available to collect the batch b_j. Then, we define the starting time st_j of the batch b_j in Equation (2.22).

$$t_j^{FPA} = \min_{k\in\{1,\dots,l\}} \max_{s\in\{1,\dots,j-1\}} y_{sk} * \Big(st_s + T_{service}(b_s)\Big), \quad \forall\, j \in \{2,\dots,m\}. \tag{2.21}$$

$$st_j = \min\{\min\{ar_i * x_{ji}\} + t^{FTW}, TPA\_b_j\}, \quad \forall\, (ar_i * x_{ji}) \geq t_j^{FPA}, \\ \text{and } \forall\, j \in \{1,\dots,m\}, \text{ and } \forall\, i \in \{1,\dots,n\}. \tag{2.22}$$

» **Constraint for Variable Time Window algorithm depending on the number of batches:** In this algorithm, we need to define the parameter $MinNumBatches$. This parameter is the minimum number of batches in the queue to start the batch collection in the system.

$$st_j = \min\{\max\{ar_i * x_{ji}\}, TPA\_b_j\}, \Rightarrow \sum_{q=j}^{m} \max x_q \geq MinNumBatches, \\ \forall\, j \in \{1,\dots,m\}, \text{ and } \forall\, i \in \{1,\dots,n\}. \tag{2.23}$$

» **Constraint for Variable Time Window algorithm depending on the number of orders:** In this algorithm we need to define the parameter $MinNumOrders$. This parameter is the minimum number of orders in the queue to start the batch collection in the system.

$$st_j = \min\{\max\{ar_i * x_{ji}\}, TPA\_b_j\}, \Rightarrow \sum_{q=j}^{m}\sum_{w=1}^{n} x_{qw} \geq MinNumOrders, \quad \forall\, j \in \{1,\ldots,m\}, \text{ and } \forall\, i \in \{1,\ldots,n\}. \tag{2.24}$$

» **Constraint for Variable Time Window algorithm based in the capacity W of the current batch:** In this algorithm we need to define the parameter *threshold*. This parameter is the threshold that limits the estimated probability that the next order will fit in the available capacity. This time-window algorithm (VTF_M1) is published in [95], where we can find more details about the implementation.

$$st_j = \min\{\max\{ar_i * x_{ji}\}, TPA\_b_j\}, \Rightarrow\ P(W - (\sum_{z=1}^{n}(w_z * x_{jz})) \leq w_{i+1}) \geq threshold, \quad \forall\, j \in \{1,\ldots,m\}, \text{ and } \forall\, i \in \{1,\ldots,n\}. \tag{2.25}$$

There are more time-window algorithms in the literature [95], but we review here only the most representative sample of the time window algorithms used in the context of OBP.

2.4.2 S-Shape routing constraints

In this section, we model the S-Shape routing algorithm. This model represents the function $dis(b_j)$ and returns the distance of the picking tour for batch b_j. We choose modeling the S-Shape routing algorithm as an example model, but we can use another routing algorithm to define the function $dis(b_j)$. We use the S-Shape routing algorithm because it is the most common routing algorithm in the literature and is widely used in this Doctoral Thesis. First, in Table 2.2, we present the parameters and variables to define the model of the S-Shape routing algorithm. Then, we define the equation that model the algorithm.

$$z_{jq} = p_{iq} \cdot x_{ji}, \quad \forall\, j \in \{1,\ldots,m\},\ \forall\, i \in \{1,\ldots,n\},\ \forall\, q \in \{1,\ldots,Q\}. \tag{2.26}$$

$$A_j = \sum_{q=1}^{Q} z_{jq}, \quad \forall\, j \in \{1,\ldots,m\}. \tag{2.27}$$

$$odd_j = A_j \mod 2, \quad \forall\, j \in \{1,\ldots,m\}. \tag{2.28}$$

$$A_j^L = \min_{q\in\{1,\ldots,Q\}} q \cdot z_{jq} : (q \cdot z_{jq} > 0), \quad \forall\, j \in \{1,\ldots,m\}. \tag{2.29}$$

$$A_j^R = \max_{q\in\{1,\ldots,Q\}} q \cdot z_{jq}, \quad \forall\, j \in \{1,\ldots,m\}. \tag{2.30}$$

$$D_j^F = \max_{i\in\{1,\ldots,n\}} pax_{i,A_j^R} \cdot x_{ji}, \quad \forall\, j \in \{1,\ldots,m\}. \tag{2.31}$$

$$D_h = ABS\big(A_j^L - A_j^D\big)W + \big(A_j^L - A_j^R\big)W + ABS\big(A_j^R - A_j^D\big)W. \tag{2.32}$$

$$dis_j \begin{cases} D_h + A_j L, & odd_j = 0 \\ D_h + \big(A_j - 1\big)L + 2D_j^F, & odd_j = 1 \end{cases} \tag{2.33}$$

Variables		
dis_j	$\rightarrow$	Distance function of the picking tour for batch b_j.
A_j	$\rightarrow$	Number of aisles that contains at least one pick location in batch b_j.
A_j^L	$\rightarrow$	Leftmost aisle number from the depot that contains at least one pick location in batch b_j.
A_j^R	$\rightarrow$	Rightmost aisle number from the depot that contains at least one pick location in batch b_j.
D_j^F	$\rightarrow$	Distance from farthest item which needs to pick to the front aisle in batch b_j.
odd_j	$\rightarrow$	$\begin{cases} 1, & \text{if } A_j \text{ is odd,} \\ 0, & \text{otherwise.} \end{cases}$
z_{jq}	$\rightarrow$	$\begin{cases} 1, & \text{if batch } b_j \text{ has a item in the aisle } q, \\ 0, & \text{otherwise.} \end{cases}$

Parameters		
A^D	$\rightarrow$	Aisle number in front of the depot.
W	$\rightarrow$	Center-to-center distance between two aisles.
L	$\rightarrow$	Length of aisle.
Q	$\rightarrow$	Number of aisles in the warehouse.
pax_{iq}	$\rightarrow$	Length to the last article for aisle a_q in the order o_i.
p_{iq}	$\rightarrow$	$\begin{cases} 1, & \text{if order } o_i \text{ has a item in the aisle } q, \\ 0, & \text{otherwise.} \end{cases}$

Table 2.2: Parameters and variables for the S-Shape routing algorithm.

2.5 Instances

In this section, we describe and conceptualize the instances of order batching problems by identifying the main characteristics that appear in most articles related to OBP and then we enumerate some additional information that might be available when handling some specific variants of OBP. We also present the data sets used in this Doctoral Thesis and in the associated literature. We present a table with the main characteristics of the papers involved in the OBP. In addition, we review the common layouts of the warehouses studied in the context of order bathing problems.

Generally speaking, an instance for any of the OBP could be split into two different kinds of data that can be provided together into the same file or separately: (i) data related to the warehouse; and (ii) data related to the orders that need to be processed. The information classified under (i) contains a description of the warehouse with details such as: Dimensions, number of aisles, width and length of the aisles, number of heights, number of picking positions per aisle, depot location, picking cart capacity, speed of the picking carts/pickers, position of the orders, etc. On the other hand, in (ii) there is information related to the orders of the customers, such as: number of orders, specific items per order, due date of the order, etc. Note that depending on the variant of the OBP being tackled, the information needed may be slightly different.

More formally, an instance of OBP can be conceptually modeled using the Unified Modeling Language (UML) standard [244]. In particular, in Figure 2.12 we represent the model of an instance of Order Batching in a UML class diagram. The legend of Figure 2.12 is depicted in a light yellow UML note. The diagram includes classes (abstractions of a family of real objects), attributes (characteristics belonging to each class), relationships among classes (either composition, aggregation, or dependency/use), and the cardinal of the relationships (indicating the minimum and maximum number of objects of each class that participate in each relationship; notice that a value 1 indicates that the minimum and maximum equal 1).

The OBP instance is represented as the class `OrderBatchingInstance`, which is modeled as the aggregation of the classes `Warehouse`, `Picker`, and `Order`. Note that each particular instance has one warehouse, one or more pickers, and one or more orders. Similarly, the `Warehouse` class is made up of class `Floor` (one or more) which, at the same time, is obtained as the composition of the classes `Depot` (zero or more depots per floor) and `Aisle` (one or more aisles per floor). Each `Aisle` is composed of class `PickingPosition` (zero or more, usually depending on the type of aisle). Finally, products are represented by the class `Product`. Each order is obtained as the aggregation of one or more products, and each picking position might contain zero or more products. Similarly, a product might be present in more than one order and in more than one picking position.

Once we have a grasp of the structure of the instances of OBP, we review the different data sets used in the related literature. However, we first present the data sets used in this Doctoral Thesis. Specifically, we selected two widely used data sets of instances previously reported in the literature of different variants of the Order Batching family of problems. The first data set was introduced in [4]. This data set consists of four different warehouses previously published in the literature. Three warehouses (W1, W2, and W3) are first published in [53] and the fourth warehouse (W4) is first published in [132]. The second data set was presented in [125]. This data set consists of one warehouse (W5) based on the data sets presented in [53, 85, 130]. Subsequently, both data sets are widely used in the related literature [5, 160, 188, 189, 252, 250, 304].

To complement this information, in 2.5.2 we perform an extensive analysis of the sets of instances used in the state of the art to evaluate the algorithms proposed for OBP. In Table 2.4 we present a compilation of the main characteristics of different warehouse and order configurations used in the literature for OBP. In particular, we have compiled: The design of the warehouse,

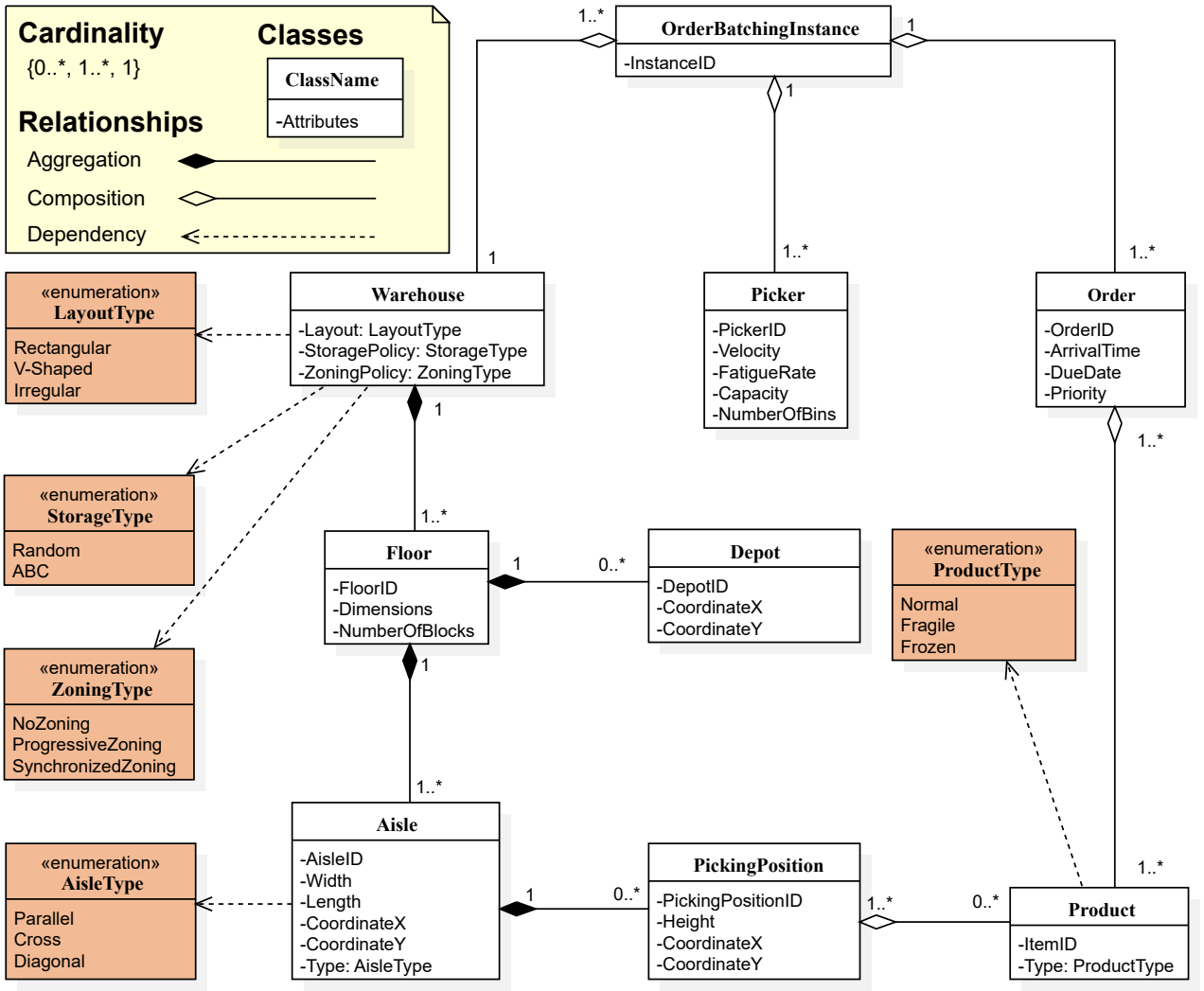


Figure 2.12: UML Class Diagram of an Order Batching instance.

including the shape, the number of floors and blocks; the number of parallel and cross aisles of the warehouse; the number of storage locations and the availability of one or more products in that location; the storage of each product in a single or multiple locations; the number of depots and its position in the warehouse; the capacity of the picking device to collect the items; the number of products in the warehouse; the orders per instance and the number of items per order. We also report some additional characteristics which are related to particular variants of the OBP such as the existence of narrow aisles or AS/RS machines; the number of pickers available; or the time horizon observed in the online problems. For each configuration, we report the reference of the source paper where the instance type is considered.

2.5.1 Warehouse layout

The layout of the picking zone in logistic warehouses is diverse. The main characteristics in the picking zone are the layout of the floor, the number of aisles and its type (cross, parallel, or diagonal), the width of the aisles, and the position of the depots, among others. We understand as a depot the place where pickers begin and finalize the collection route and where items collected on a picking route are delivered [297]. In the literature, the most common floor plan is rectangular, but it is also possible to find irregular floors [232, 311]. In addition, we can find three types of aisles (parallel aisles, cross aisles, and diagonal aisles). Warehouses with diagonal

aisles are known as warehouses with flying-V, fish-bone o chevron, according to the layout of the diagonal aisles [15, 66, 182, 208, 227]. Warehouses with rectangular floor plans and parallel and cross aisles are typically divided into blocks according to the number of cross aisles. Each block is a group of parallel aisles delimited by two cross aisles. The width of the aisles is also important because in warehouses with narrow aisles, the pickers cannot turn around or cross with other pickers within the corridors [33, 141]. In Figure 2.13 we can find several examples of different layout configurations of the picking zone in a warehouse. This figure was inspired by several publications in the literature [183, 209, 310] and collects some of the most common layouts.

In this Doctoral Thesis, we focus on warehouses manually operated, however, the layout of the warehouse might also be linked to another important characteristic of a warehouse management, which is the degree of automation in the warehouse. We can highlight some automatic systems with different levels of automatization, such as the Automated Storage and Retrieval System (AS/RS) [69, 70, 71, 72, 212], vertical lift modules [174], carousels [135, 175], autonomous mobile robots (AMRs) [309], or dense mobile racks [299].

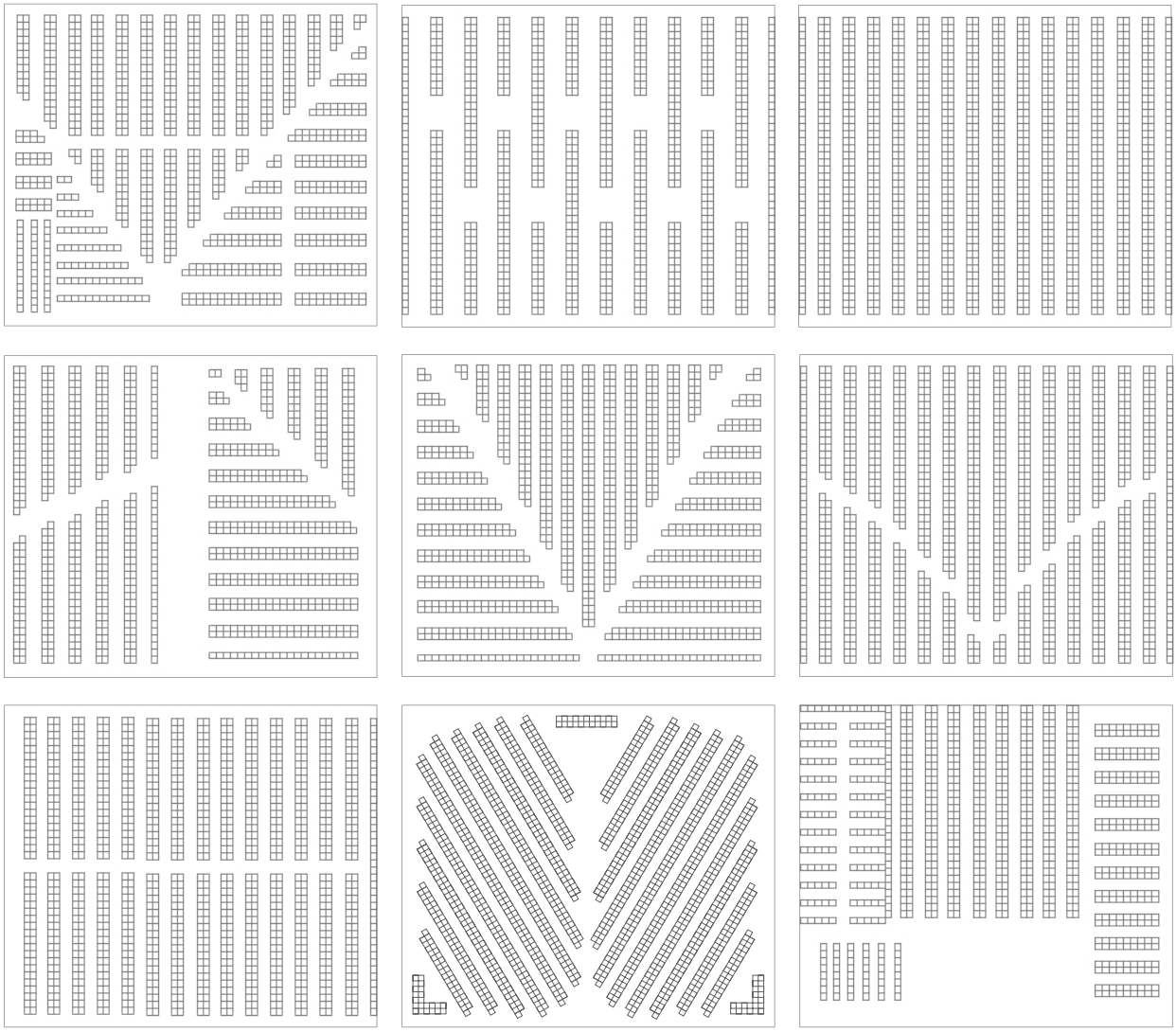


Figure 2.13: Example of different layouts of the picking zone in a warehouses.

In Figure 2.14 we show an example of the warehouse layout studied in this Doctoral Thesis. As we can observe, it is a warehouse with a rectangular shape and only one block (i.e., with two

cross aisles placed at the front and back of the block). In this case, there are five parallel aisles, and the aisles are considered wide enough so that pickers cross with other pickers and turn around. The warehouse is not automated, and the pickers collect the items manually. There is only one depot. In this case, the depot is located in the leftmost part of the front cross aisle. However, the number of parallel aisles or the placement of the depot depends on the particular instance. In Table 2.3, we can find a summary of the main characteristics of the warehouses studied in this Doctoral Thesis and the associated working parameters considered. These data sets were introduced in [4] and [125].

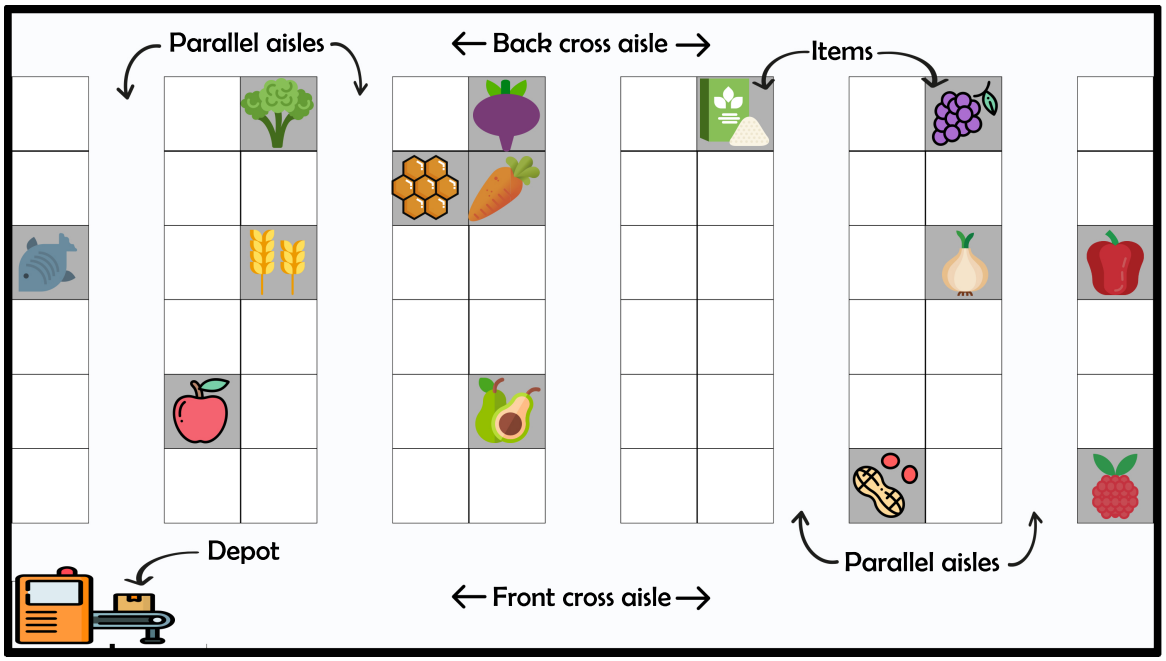


Figure 2.14: Example of the layout of the logistic warehouse used in this Doctoral Thesis.

	W1 [125]	**W2** [125]	**W3** [125]	**W4** [125]	**W5** [4]
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	2x30	2x20	2x25	2x16	2x45
Number of items	240	400	1250	384	900
Parallel aisle length	50m	10m	50m	80m	45m
Parallel aisle width	4.3m	2.4m	5m	15m	5m
Number of instances	20	20	20	20	64
Travel speed (m/min.)	48	48	48	48	48
Extraction speed (items/min)	6	6	6	6	6
Batch setup time	3 min	3 min	3 min	3 min	3 min

Table 2.3: Warehouse characteristics and the work parameters used in this Doctoral Thesis.

2.5.2 Characteristics of the instance sets in the literature

In this section, we have compiled the main characteristics of the instances used in articles belonging to the OBP literature. In particular, we have selected the characteristics compiled in Table 2.4. Then, in Table 2.5 we have sorted the articles chronologically and compiled the aforementioned characteristics. Notice that when the information was not available in the paper, we denoted it with a '–' in the table. For each set of instances, we also identify if the instances were obtained or inspired from a previous paper (column 'Source').

Columns	Values
(#OR) Number of orders	Max. . . Min
(L/F) Layout/Floors	R:rectangular ; I:irregular / S:single ; VL: Vertical Lift ; DMR: Dense Mobile Rack
(DP) Depot position	Co: Corner ; Ce: Center ; CV: Conveyor ; Li: Line depot
(#BL) Number of blocks	MB: Multi-blocks ; Max. . . Min
(#PA) Number of parallel aisles	Max. . . Min
(#PR) Number of products	Max. . . Min
(ASS) Assignment	R: R ; ABC: A ; SIM: S
(PCC) Picking cart capacity	Max. . . Min
(#PI) Number of pickers	V: Varied ; AGV: Automated guided vehicles; Max. . . Min
(TI) Type of instance	Syn: Synthetic ; Rea: Real ; Reu: Reused

Table 2.4: Characteristics reviewed in the instances related to OBP.

#ID	Papers	#OR	L/F	DP	#BL	#PA	#PR	ASS	PCC	#PI	TI	Source
#1	[9]	30	R/S	CV	1	12	2000	–	6	12	Rea	–
#2	[69]	10	AS/RS	–	–	–	22	R	30. . . 39	1	Syn	–
#3	[71]	23. . . 52	AS/RS	–	–	–	80	R	350	1	Syn	–
#4	[88]	100. . . 1200	R/S	Co	1	10	800	A	10	1	Syn	–
#5	[72]	20	AS/RS	–	–	–	80	R	400. . . 1200	1	Syn	–
#6	[212]	20. . . 50	AS/RS	–	–	–	40	R/A	70. . . 200	1	Syn	–
#7	[240]	100. . . 1000	R/S	Co	1	15	750	A	50	1	Syn	–
#8	[70]	N(50,15)	AS/RS	–	–	–	120	R	400. . . 1200	1	Syn	–
#9	[266]	–	R/S	–	–	40	–	A	4. . . 20	1	Syn	–
#10	[53]	30	R/S	Co	1	4. . . 25	240. . . 1250	R	12. . . 150	1	Syn	–
#11	[52]	–	R/2	Co	18	11	300000	A	12. . . 16	–	Rea	–
#12	[241]	4000	R/S	No	1	9	–	R/A	15. . . 60	5. . . 90	Syn	–
#13	[40]	–	R/S	Co	1	40	–	A	7. . . 20	1	Syn	–
#14	[222]	3000. . . 12000	R/S	Co	1	10	1000	A	30	8. . . 32	Syn	–
#15	[85]	15. . . 32	R/S	Ce	1	4. . . 20	250. . . 400	R/A	3. . . 8	3. . . 8	Syn/Reu	#10
#16	[146]	40. . . 300	R/S	Co	1	5	80. . . 400	–	100. . . 500	1	Syn	–
#17	[84]	15. . . 32	R/S	Ce	1	4. . . 20	250. . . 400	R/A	3. . . 8	1	Reu	#15
#18	[148]	10. . . 30	R/S	Co	1	10	400	R	24	1	Syn	–
#19	[296]	10..100 o/h	AS/RS	–	–	–	2000	R	5. . . 30	1	Syn	–
#20	[132]	250	R/S	Co	1	12	384	R/A	80	1	Syn	–
#21	[65]	5. . . 40	R/S	Co	1	6. . . 8	–	R/A	2. . . 6	1	Syn	–
#22	[172]	24 h/o	R/S	Ce	2	6. . . 16	–	R	3. . . 40	1	Syn	–
#23	[271]	25. . . 250	R/S (3D)	Co	1	–	–	–	7000. . . 50000	1	syn	–
#24	[19]	20..25	R/S	Ce	1	11. . . 22	–	R	25	1	Syn	–
#25	[133]	250	R/S	Co	1	12	384	R/A	80	1	Reu	#20
#26	[4]	50. . . 250	R/S	Co/Ce	1	4. . . 25	240. . . 1250	R/A	12. . . 150	1	Reu	#10 #20
#27	[281]	–	R/S	Ce	2	10. . . 20	–	R	2. . . 20	4. . . 8	Syn	–

Continued on the next page

#ID	Papers	#OR	L/F	DP	#BL	#PA	#PR	ASS	PCC	#PI	TI	Source
#28	[302]	105...142 o/h	R/S	CV	1	36	240	R	1...7	27	Rea/Syn	–
#29	[130]	20...60	R/S	Co	1	10	900	A	30...75	1	Reu	#10 #11
#30	[243]	234...1136	I/S	–	–	–	1000	–	60	8	Rea	–
#31	[145]	300	R/S	Ce	2	10	400	A	50	1	Syn	–
#32	[248]	–	R/S	Co	1	10...30	400...1000	R	8...25	1	Syn	–
#32	[125]	30...120	R/S	Co	1	10	900	A	45...75	1	Reu	#17 #29
#33	[129]	40...100	R/S	Co	1	10	900	R/A	30...75	1	Reu/Syn	#10 #29
#34	[141]	16...1440	R/S	Co	1	4...30	–	R/A	10	4...24	Reu	#13
#35	[165]	50...250	R/S	Co/Ce	1...2	5	100	R	100	1	Syn	–
#36	[140]	360...2160	R/S	Co	1	10...40	400...1600	R/A	10	1	Reu/Syn	#14
#37	[23]	–	–	–	–	–	–	–	–	1	Syn	–
#38	[73]	15...45	R/S	Ce	2	5	2280	A	–	1	Rea	–
#39	[12]	40...150	R/S	Co	1	–	3000	–	200...550	1	Syn	–
#40	[128]	20...80	R/S	Co	1	10	900	A	45...75	1	Reu	#10 #29
#41	[298]	257...360 o/h	R/S	Co	1	10...30	–	R	20	1	Reu	#22 #27
#42	[184]	1536	R/S	Ce	3	57	2912	–	3...4	1	Rea	–
#43	[219]	30...60	R/S	Co	1	10	900	A	45/75	1	Reu	#29
#44	[312]	60	R/S	Co	1	5	60	–	100	1	Syn	–
#45	[187]	50...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	1	Reu	#26
#46	[127]	100...200	R/S	Co	1	10	900	A	30...75	2...8	Reu	#29
#47	[218]	30...90	R/S	Co	1	10	900	–	45...100	1	Reu	#29
#48	[214]	100...500	R/S	CV	1	1...3	–	–	15...50	5...10	Syn	–
#49	[204]	10...100	R/S	Co/Ce	1	10...22	200...900	R/A	24...75	1	Reu	#18 #24 #33
#50	[38]	6...8	R/S	Co	1	3	12	–	4...5	1	Rea	–
#51	[202]	20...100	R/S	Co	1	10	200	R	24...48	1	Reu	#18
#52	[39]	5...200	R/S	Co	1	3	12...24	–	4...9	1	Rea	–
#53	[303]	600	R/S	Co	1	10	900	R	30...60	8...18	Reu	#17 #29
#54	[160]	20...60	R/S	Co	1	10	900	A	30...75	1	Reu	#10 #29
#56	[179]	47...100	R/S	Co	1	3...4	16...48	–	4	1	Rea	–
#57	[274]	5...30	R/S	Co	2	3...8	108...1584	A	40	1...6	Rea	–
#58	[178]	200...10000	R/S	–	2	11	2000	–	50...100	1	Rea/Syn	–
#59	[277]	–	R/S	Co	2	16	1280	R/A	26	1...4	Rea	–
#60	[139]	200...500	R/S	Co	1	6	–	A	2...10	1	Syn	–
#61	[304]	120...480	R/S	Co	1	10	900	R	30...75	1...8	Reu	#28 #32
#62	[275]	5...5000	R/S	Co	2...4	3...12	108...1584	A	40	1...6	Rea/Reu	#57
#63	[190]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	–	Reu	#26 #33
#64	[189]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	1	Reu	#26 #33
#65	[188]	20...80	R/S	Co	1	10	900	R/A	45...75	1	Reu	#40
#66	[252]	100...200	R/S	Co	1...3	10...30	500...1500	A	45...75	2...5	Reu	#46
#67	[250]	20...80	R/S	Co	2	10...30	1000...3000	A	30...75	1	Reu	#29 #33
#68	[175]	31...72	VC(AS/RS)	–	–	–	–	–	6...8	1	Rea	–
#69	[185]	2048...2056	R/S	Ce	2	57	–	–	3	3...86	Rea	–
#70	[87]	288...1440	R/S	Ce	1	10	300	R	20	1	Rea	–
#71	[28]	20...100	R/S	Co	1	10	900	–	25...100	1	Reu	#43 #54
#72	[36]	100...200	R/S	Co	2	15	900	R	20...400	1...5	Reu	#50 #53
#73	[6]	5...1093	R/S	Co	1	23	1150	–	2...12	2...8	Syn	–
#74	[22]	45...18209	R/S	CV	4	7	–	–	–	–	Rea	–
#75	[151]	–	R/S	Co	1	–	–	–	–	1	Syn	–
#77	[308]	200...600	R/S	Co	1	10	900	A	6...15	1	Reu/Syn	#33
XXX	[176]	17...192	VLM(AS/RS)	–	–	1...4	70...520	–	6...8	1	Rea	#68
#78	[305]	2400...9600	R/S	Co	1	1...100	100...1000	R	20	5	Reu	#17 #32
#79	[147]	500...2000	R/S	–	5	–	10000	–	10	25...75	Rea	–
#80	[89]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	1	Reu	#26 #29
#82	[191]	25...200	R/S (3D)	Co	1	3	30...300	–	7000...20000	1	Reu	#23
#83	[7]	5...200	Any	–	–	–	–	–	3...8	1	Syn	–
#84	[279]	100...300	R/S	Co	2	6...18	360...3240	R/A	4...12	2...8	Reu	#59

Continued on the next page

#ID	Papers	#OR	L/F	DP	#BL	#PA	#PR	ASS	PCC	#PI	TI	Source
#85	[21]	5...5000	R/S	Co/Ce	2	4	1584	–	40	1	Rea	#62 #74
#86	[225]	10...50	R/S	Co	19	–	20...300	A	50...200	1	Syn	–
#87	[254]	195...2673	R/S	–	1	35...50	2491	R	30...90	2...8	Reu	#15
#88	[90]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	2	Reu	#26 #29
#89	[26]	40	R/S	Co	2	10	900	–	50	1	Reu	#46 #64 #65
#90	[63]	30...520	R/S	Li	1	–	–	–	–	5...40	Rea	–
#91	[137]	120...480	R/S	Co	2...6	10	900	R	30...75	2...6	Syn	–
#92	[29]	–	R/S	Co	2	3	50...3800	–	–	–	Syn	–
#93	[177]	2000	R/S	–	–	–	800	–	–	1...18	Rea	–
#94	[163]	50...200	R/S	Co	3	10	6000	A	100	1	Syn	–
#95	[301]	4...300	R/S		2	2...10	16...400	R	2...75	1	Reu	#77
#96	[5]	60...240	R/S	Co	1	10	900	A	45...75	2	Reu	#32
#97	[273]	5...75	R/S	Co	1...2	8...16	1560	A	–	1...12	Reu	#56
#98	[8]	10...100	R/S	Co	1	23	1150	–	50...100	2...5	Rea	–
#99	[91]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	1	Reu	#26 #29
#100	[32]	10...100	R/2	Ce	1	5...19	100...7600	A	20	6...25	Rea	–
#101	[92]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	1	Reu	#26 #29
#102	[164]	5...200	R/S	Co	2	5	200	A	5...60	1...6	Reu/Rea	#81
#103	[11]	2...40	R/S	Co	–	–	2...600	–	4...6	2...10	Syn	–
#104	[25]	330...500 o/h	Mix AS/RS	CV	2	–	–	–	50	3...5	Syn	–
#105	[259]	2000...8500	R/S	Co	1	40	–	–	6	2...8	Syn	–
#106	[297]	10...300	R/S	Ce(2)	2	6	24...3240	A	18	2+AGV(2)	Reu	#84
#107	[10]	–	R/S	Ce	2	4	54	–	45	2...4AGV	Syn	–
#108	[306]	–	R/S	Co	1	100	1000	R	5...10	2	Rea	#78
#109	[75]	60...800	R/S	CV	1	–	–	–	15	30	Syn	–
#110	[299]	20...50	DMR	Ce	1	30...70	–	–	–	1	Syn	–
#111	[30]	10...50	R/S	Co	1	–	400...2000	R	30...50	3	Syn	–
#112	[93]	40...250	R/S	Co/Ce	1	4...25	240...1250	R/A	12...150	1...5	Reu	#26 #29
XXX	[309]	5...100	R/S	Co	1	10...20	200...800	R	6...60	4...12AGV	Syn	–
#113	[207]	5...5000	(R-I)/S	Co/Ce	2...4	8...12	254...42	R	8x40	1	Rea/Reu	#62
#114	[136]	–	R/S	CV	–	2...4	2212	–	–	2...4	Rea	–
#115	[152]	50...1200	R/S	Co	1	10	900	A	50	1	Syn/Reu	#65
#116	[192]	25...250	R/S (3D)	Co	1	–	–	–	7000...50000	1	Reu	#23
#117	[247]	–	R/S	Ce/1...63...11		5...60	5600	–	–	1	Syn/Rea	–
#118	[233]	10...250	R/S	Co	1	4...40	64...640	R/A	3...10	3	Syn	–
#119	[287]	40...600	R/S	Co	1	20	1000	R/A	6...24	1...5	Reu	#33 #77
#120	[300]	1000	R/S	Ce	MB	–	–	S	–	AGV	–	–

Table 2.5: Characteristics of instances used in articles related to OBP.

Chapter 3

State of the art

In this chapter we review the state of the art related to Order Batching problems. To that aim, we enumerate and sort in chronological order all the articles found in the literature published in a journal indexed in either the Journal of Citation Reports (JCR) or the Scimago Journal Rank (SJR). We also compile the authors which have published at least two papers in the field. Finally, we present the state-of-the-art of the OBP through our own article titled "*Order Batching Problems: taxonomy and literature review*". To that aim we contextualize the paper and summarize its content.

To prepare this state-of-the-art study, we analyzed the different survey publications that tackle the literature of OBP. During our study, we identified six previous surveys [18, 31, 54, 114, 115, 226] published in a journal indexed in the JCR and other eleven surveys [3, 27, 50, 74, 131, 149, 155, 161, 181, 186, 201] published in other journals or books, that are somehow related to the topic under consideration. In those surveys we detected different approaches when dealing with the study of the state of the art and, among them, we highlight four papers. The latest publication on the topic, made in [226], performed a detailed bibliographic analysis of the OBP and also analyzed the different strategies/methodologies previously used to solve the problem. Furthermore, they studied the different subtasks involved in the OBP. We can also highlight another recent publication [31] which deals with the different operations that are addressed when solving the order batching problem. The authros present an overview classification of the problem and also discusses different solution techniques and future research topics. Another work to consider is the one published in [27] which focuses on the trends in order batching, sequencing, and routing problems. This article also tackles several subtasks of the OBP. Finally, we would like to highlight the work published in [131]. Despite of the fact that it is a little bit older than the previous ones, this was the first work which focused exclusively on the Order Batching Problem. It is important to notice that among the aforementioned surveys not all of them deal exclusively with order batching, but they present the set of related activities as a part of logistic warehouses [18, 54, 114, 115]. In these cases, the batching process is briefly discussed in a particular section of the article.

Due to this previous analysis, we were able to identify the need to conduct a new survey, titled "*Order Batching Problems: taxonomy and literature review*" [94], that would improve the state of the art of the field under study. In this publication, we reviewed the state-of-the-art of order batching. Particularly, we identified 193 publications related to OBP. Among them, we found journal papers, but also other works, such as conference papers, books, book chapters, and Doctoral Theses. However, we focused our study in the 122 papers published in a journal indexed in the JCR or SJR. As we state in the paper, the order batching is a family of problems in constant growth. In this paper, we compile and review the publications indexed in JCR or SJR classified by year and grouped by category. We can observe that the first publication related to OBP was performed in 1979. Since then, four decades later, the number of papers

related to OBP published in top level journals are more than a hundred. In the first and second decades (80's-90's), we observed that very few papers related to OBP could be found in the literature. The interest on the problem started growing the late 1990s and early 2000s, when more studies on the OBP appeared. But the real hatching began in 2011 where a large number of annual publications related to OBP started to be produced. Nowadays, we can find more than 10 annual publications in the literature about OBP.

In the review presented in [94] we collected the most well-known variants of the optimization problems named as order batching, for rectangular-shaped warehouses, classified as picker-to-part (i.e., when the pickers move through the warehouse to collect the orders). In this work, we identify the different factors and tasks that influence the picking process and we classify them into decision levels (strategic, tactical, and operational). In addition, we characterize and identify the optimization problems that belong to the OBP family. We also propose a new taxonomy to classify the identified problems. The proposed taxonomy can host future problems within this family. In this work, we study each variant of order batching and highlight the common characteristics of the different variants of the problem. We identify the strategies and algorithms proposed for the set of articles analyzed. Finally, we detail and summarize in several tables the works on OBP that is indexed in the JCR and SJR rankings.

To complement the content of the state of the art presented in [94], we include in this chapter a new bar chart to provide a general vision of the literature reviewed. The bar chart, shown in Figure 3.1, presents a chronological classification of the papers published in either a JCR or SJR journal. Those publications are classified depending on the variant of the OBP studied. Particularly, we can observe that there are 18 variants of the problem that have been previously studied out of 36 variants identified in the taxonomy introduced in [94].

Among the variants studied, we observe that 48.36 % of the articles deal with offline single picker variants, 26.23% of the articles deal with offline multiple picker variants, 13.11% of the articles deal with online single picker variants, and 12.30% of the articles deal with online multiple picker variants. Also, we observe that the offline and the single-picker variants of the problem have been studied further than the online and the multiple-picker variants. The most studied variant is the classic OBP with 31 articles. The next ones are the variants that optimize only the batching task, such as OBRP with 16 articles, OOBPMP with 11 articles, OOBP with 10 articles, and OBPMP with 9 articles. It is worth mentioning that further than the batching, the routing task must be always addressed in any OBP paper. However, other tasks are not always considered. In this sense, we observe that only 29 papers study the sequencing task, 15 papers study the assignment task, and 3 papers study the waiting task.

To extend the previous information, in Table 3.1 we show a classification of authors with more than two publications related to OBP in a journal indexed in JCR or SJR. Authors are sorted depending on the number of publications. We also present the reference to the articles published by each author. In this table, we can also observe that the author of this Doctoral Thesis is currently the sixth author with more publications related to OBP.

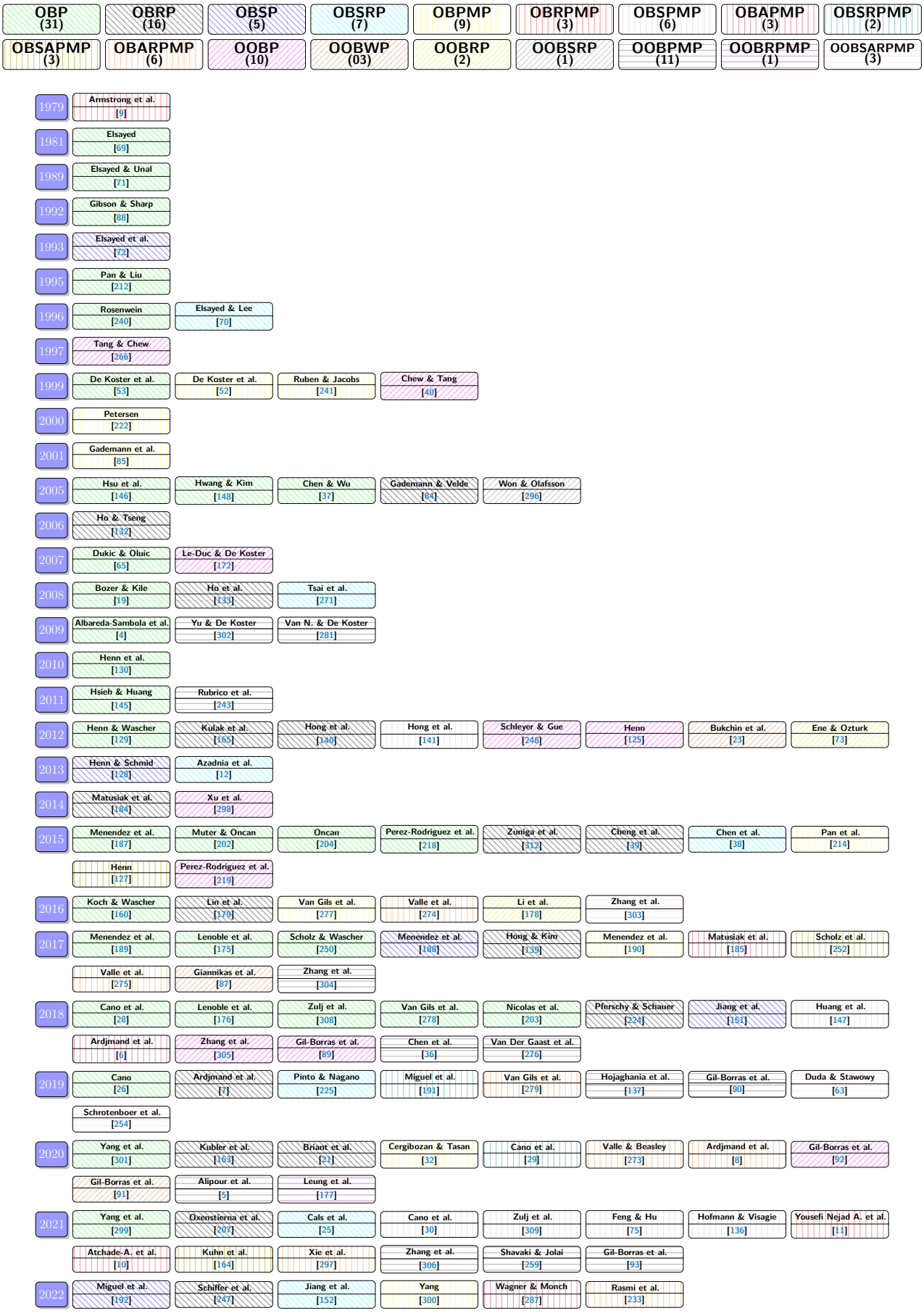


Figure 3.1: Publications index in JCR and SJR classified by year and grouped by variant of the problem.

Duarte, A.	9	[188] [189] [190] [89] [91] [90] [93] [187] [92]
Pardo, E. G.	9	[188] [189] [190] [89] [91] [90] [93] [187] [92]
Alonso-Ayuso, A.	8	[4] [189] [89] [91] [90] [93] [187] [92]
De Koster, R. B. M.	7	[53] [281] [184] [302] [52] [172] [185]
Wäscher, G.	6	[160] [129] [255] [130] [252] [250]
Gil-Borrás, S.	5	[89] [91] [90] [93] [92]
Zhang, J.	4	[304] [305] [303] [306]
Elsayed, E. A.	4	[71] [72] [69] [70]
Cano, J. A.	4	[28] [26] [30] [29]
Menéndez, B.	4	[188] [189] [190] [187]
Valle, C. A.	4	[273] [275] [274] [272]
Beasley, J. E.	4	[273] [275] [274] [272]
Roodbergen, K. J.	3	[276] [52] [254]
Scholz, A.	3	[255] [252] [250]
Henn, S.	3	[129] [128] [130]
Ardjmand, E.	3	[7] [6] [8]
Lenoble, N.	3	[175] [174] [176]
Frein, Y.	3	[175] [174] [176]
Hammami, R.	3	[175] [174] [176]
Hong, S.	3	[141] [140] [139]
Wang, X.	3	[304] [305] [303]
Hu, X.	3	[152] [151] [75]
Schubert, D.	3	[164] [255] [252]
Molina, E.	3	[4] [189] [187]
Correa-Espinal, A. A.	3	[28] [30] [29]
Cheng, C.-Y.	3	[38] [39] [179]
Chew, E. P.	2	[40] [266]
Tang, L. C.	2	[40] [266]
Pérez-Rodríguez, R.	2	[219] [218]
Hernández-Aguirre, A.	2	[219] [218]
Öncan, T.	2	[202] [204]
Bajgiran, O. S.	2	[7] [6]
Ho, Y. C.	2	[133] [132]
Van Gils, T.	2	[279] [277]
Caris, A.	2	[279] [277]
Ramaekers, K.	2	[279] [277]
Braekers, K.	2	[279] [277]
Koch, S.	2	[160] [130]
Huang, K.	2	[305] [303]
Miguel, F. M.	2	[192] [191]
Frutos, M.	2	[192] [191]
Tohmé, F.	2	[192] [191]
Jiang, X.	2	[152] [151]
Sun, L.	2	[152] [151]
Zhang, Y.	2	[152] [151]
Johnson, A. L.	2	[141] [140]
Peters, B. A.	2	[141] [140]
Matusiak, M.	2	[184] [185]
Saarinen, J.	2	[184] [185]
Gademann, N.	2	[84] [85]
Gómez-Montoya, R. A.	2	[28] [29]
Lee, M. K.	2	[72] [70]
Žulj, I.	2	[309] [308]
Schneider, M.	2	[309] [308]
Chen, T.-L.	2	[38] [39]
Chen, Y.-Y.	2	[38] [39]
Da Cunha, A. S.	2	[275] [274]
Pan, J. C.-H.	2	[212] [214]

Table 3.1: Classification of authors by number of publication with more than 2 articles of OBP in JCR and SJR index.

To end this review of the state of the art, we collect some notes related to the resolution methods used for the tasks addressed in the OBP. First, focusing on the routing task, we observe that according to [94], most of the papers studied use more than one routing strategy. In particular, any kind of heuristic/metaheuristic strategy is used in all the articles studied. Among these strategies, the S-shape method is used in 50.81% of the articles. On the other hand, 34.42% of the articles also propose the use of an exact approach, based mainly on a mathematical model or dynamic programming.

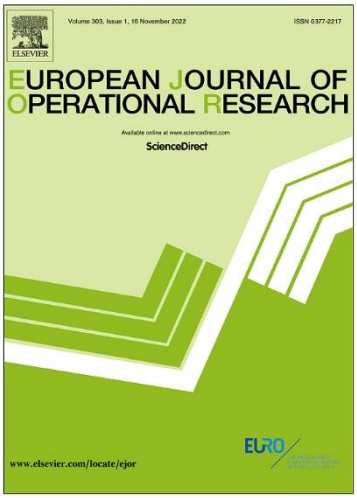
To tackle the batching task, metaheuristics are the most common strategies, since 60.65% of the articles studied included at least one metaheuristic method. Simpler heuristics are studied in 36.06% of the articles; among them, we can highlight the Seed, Savings, and FCFS methods as the most used ones. Furthermore, we identify that an exact method (usually a mathematical model solved with a commercial solver) has been proposed in 27.04% of the articles studied.

Other tasks, such as sequencing or assigning, are usually handled together with the batching task. Finally, we can say that the waiting task has been little explored in the literature, despite of the fact that it has been shown to have a profound impact on the overall performance of the picking algorithms.

3.1 Order Batching Problems: Taxonomy and literature review

Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte A. (2022). Order Batching Problems: taxonomy and literature review. European Journal of Operational Research, Article in 2nd review.

Published in "**European Journal of Operational Research**"



ISSN: 0377-2217 / 1872-6860

Article under review (2nd Round)

Impact factor (JCR 2020): 5.334

Journal Citation Indicator (JCI): 1.42

Rank by Journal Impact Factor

- Operations Research & Management Science. Ranking 15/84 (Q1).

Rank by Journal Citation Indicator (JCI)

- Operations Research & Management Science. Ranking 13/99 (Q1).

Rank by Journal Impact Factor

Journals within a category are sorted in descending order by Journal Impact Factor (JIF) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

EDITION
Science Citation Index Expanded (SCIE)
CATEGORY
OPERATIONS RESEARCH & MANAGEMENT SCIENCE
15/84

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	15/84	Q1	82.74
2019	14/83	Q1	83.73
2018	13/84	Q1	85.12
2017	12/84	Q1	86.31
2016	7/83	Q1	92.17

EDITION
Science Citation Index Expanded (SCIE)
CATEGORY
MANAGEMENT
n/a

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	n/a	n/a	n/a
2019	n/a	n/a	n/a
2018	n/a	n/a	n/a
2017	n/a	n/a	n/a
2016	n/a	n/a	n/a

Rank by Journal Citation Indicator (JCI)

Journals within a category are sorted in descending order by Journal Citation Indicator (JCI) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

CATEGORY
OPERATIONS RESEARCH & MANAGEMENT SCIENCE
13/99

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	13/99	Q1	87.37
2019	10/99	Q1	90.40
2018	10/98	Q1	90.31
2017	10/96	Q1	90.10

European Journal of Operational Research

Order Batching Problems: taxonomy and literature reviiew

--Manuscript Draft--

Manuscript Number:	EJOR-D-22-01501R1
Article Type:	Invited Review
Section/Category:	(T) Supply chain management
Keywords:	Supply chain management; Order Batching; Order Picking; Warehousing
Corresponding Author:	Eduardo G. Pardo, Ph.D. SPAIN
First Author:	Eduardo G Pardo
Order of Authors:	Eduardo G Pardo
	Sergio Gil-Borras
	Antonio Alonso-Ayuso
	Abraham Duarte
Manuscript Region of Origin:	SPAIN
Abstract:	Order Batching is a family of optimization problems related to the process of picking items in a warehouse as part of supply chain management. In particular, problems classified under this category are those whose picking policy consists of grouping the orders received in a warehouse into batches, prior to starting the picking process. Once the batches have been formed, all items within the orders of the same batch are picked together on the same picking route. In this survey, we focus our attention on manual picking systems and rectangular-shaped warehouses with only parallel and cross aisles, which is the most common warehouse configuration in the literature. The objective function of each problem that belongs to this family differs slightly. This survey reviews the optimization problems known in this category. First, we identify the strategic, tactical, and operational decision levels that influence the picking task. Then, we characterize the optimization problems belonging to this family. The identified problems are classified into a taxonomy proposed in this paper which is able to host future problems within this family. Later, we deep in the most outstanding papers by category and review the strategies and algorithms proposed for the most relevant activities: batching, routing, sequencing, waiting, sorting, and assigning. To conclude this paper, we outline the open issues and future paths of the topic under study.

Order Batching Problems: taxonomy and literature review[☆]

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

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

Abstract

Order Batching is a family of optimization problems related to the process of picking items in a warehouse as part of supply chain management. In particular, problems classified under this category are those whose picking policy consists of grouping the orders received in a warehouse into batches, prior to starting the picking process. Once the batches have been formed, all items within the orders of the same batch are picked together on the same picking route. In this survey, we focus our attention on manual picking systems and rectangular-shaped warehouses with only parallel and cross aisles, which is the most common warehouse configuration in the literature. The objective function of each problem that belongs to this family differs slightly. This survey reviews the optimization problems known in this category. First, we identify the strategic, tactical, and operational decision levels that influence the picking task. Then, we characterize the optimization problems belonging to this family. The identified problems are classified into a taxonomy proposed in this paper that is able to host future problems within this family. Later, we review the most outstanding papers by category and review the strategies and algorithms proposed for the most relevant activities: batching, routing, sequencing, waiting, sorting, and assigning. To conclude this paper, we outline the open issues and future paths of the topic under study.

Keywords: Supply Chain Management, Order Batching, Order Picking, Warehousing

1. Introduction

The supply chain is the sequence of events that cover the entire life cycle of a product or service, from conception to consumption (Blanchard, 2010). Among others, it involves entities (manufacturers, suppliers, wholesalers, retailers, etc.), resources (information, materials, human, financial, etc.), and activities (acquisition, transformation, storage, distribution, etc.). Management of the supply chain includes a variety of decisions and transactions between the different entities involved in the process, which are typically classified into strategic, tactical, and operational (Misni & Lee, 2017).

A key part of the supply chain usually occurs within a warehouse, where different materials or products are received, processed, and stored for later use (i.e., they are usually

[☆]This research has been partially funded by: Ministerio de Ciencia, Innovación y Universidades, Spain (Refs.: RTI2018-094269-B-I00, PGC2018-095322-B-C22, PID2021-125709OA-C22); Comunidad de Madrid and European Regional Development Fund (Ref. P2018/TCS-4566); and by the Programa Propio de I+D+i de la Universidad Politécnica de Madrid (Ref.: Programa 466A).

[*]Corresponding author

Email addresses: eduardo.pardo@urjc.es (Eduardo G. Pardo), sergio.gil@upm.es (Sergio Gil-Borrás), antonio.alonso@urjc.es (Antonio Alonso-Ayuso), abraham.duarte@urjc.es (Abraham Duarte)

picked up and shipped). Within a warehouse, strategic decisions are usually long-term decisions related to aspects such as: determining the place to set the warehouse, choosing the right network of suppliers and transporters, selecting and customizing the software systems, determining the layout or mechanization of the warehouse, etc. The tactical point of view usually includes mid-term decisions such as: defining guidelines to meet quality and safety, determining the location where the products are going to be stored, setting the number and position of the depots (also known as Input/Output points), or performing inventory logistics, among others. Finally, the operational decisions are commonly short-term decisions such as: forecasting the daily and weekly demand, scheduling of production operations, managing incoming and outgoing products/materials, managing the orders received in the warehouse, etc. See further details in: Il-Choe & Sharp (1991); De Koster et al. (2007); Misni & Lee (2017).

The profit obtained in the warehouse is strongly dependent on how the management systems determine the operational actions. In this literature review, we focus our attention on the operational activities related to the picking of orders in a warehouse. However, these activities are also influenced by some tactical and strategic decisions. In Figure 1, we have compiled and classified the most relevant decisions, highlighted in the literature, that occur in warehouse management in relation to picking activities. Particularly, the decisions are classified into strategic, tactical, and operational. Additionally, we have split those categories into several subcategories, and then we have compiled the related decisions. These subcategories have been partially inspired by those introduced in previous research (De Koster et al., 2007; Gu et al., 2007; Misni & Lee, 2017). Notice that there are many other decisions related to the supply chain within a warehouse, but we have focused only on those that have a deep influence on the picking process and have been previously reported in the literature.

On a daily basis, warehouses receive orders from customers, consisting of a list of products that need to be retrieved from its current location, moved to a common area, packaged together, and shipped to the customer. The operational management of the warehouse must determine the sequence in which those orders/products are picked, the picker assigned to collect them, the route that the picker must follow, or the moment in time when starting the picking, among others. The picking process in a warehouse was one of the first optimization targets to be considered (Gudehus, 1973). The picking is highly influenced by the storage policy (Ruben & Jacobs, 1999), the layout of the warehouse, and the routing strategy, among others (Petersen, 1997). Some authors stated that this process can consume up to 60% of the total time of all labor activities in a warehouse (Drury, 1988; Coyle et al., 1996), which can suppose more than half of the total operational costs (Tompkins et al., 2010). In this context, an important operational decision consists of determining whether orders are collected in isolation (strict order picking) or grouped prior to be picked (order batching). The strict order picking is usually considered a naive strategy, consisting of assigning each order to a picker, who collects all the items within that order. Once all the items have been retrieved, they are placed in a depot, and then the system assigns a new order to the picker, and so on. On the contrary, order batching is considered a more advanced strategy. It consists of grouping several orders into a batch that does not exceed a predefined capacity. Then, once the batch is conformed, it is assigned to a picker who retrieves all items in the orders of the batch on a single route (Elsayed, 1981). The order batching policy has been shown to be very effective compared to the strict order picking policy. Additionally, it is possible to reduce travel time by up to 35% if the routes are designed considering batching at the same time (De Koster et al., 1999a). However, it is important to remark that order batching policy is specially handful in Business-to-Customer contexts, where several small items/packages can be picked together instead of picking a

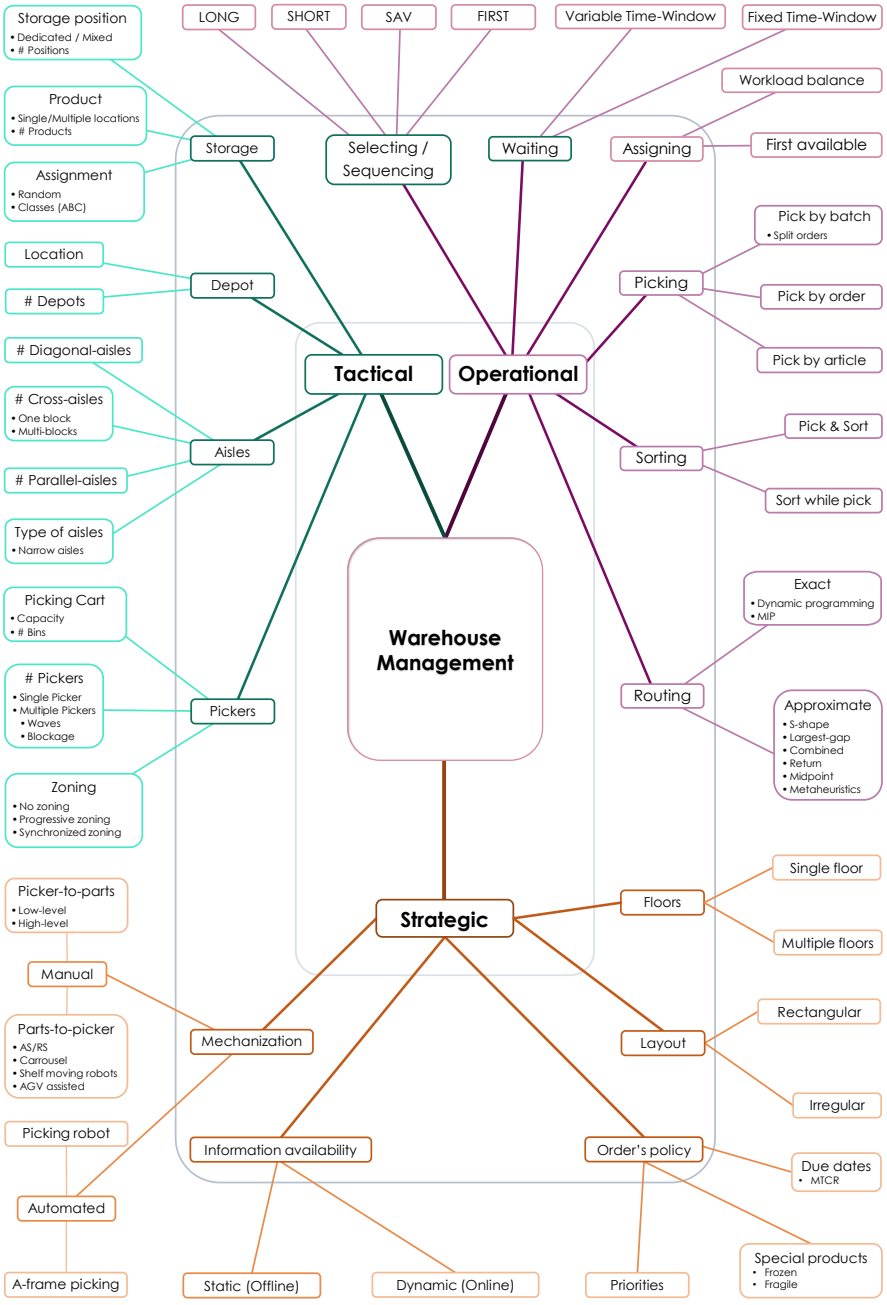


Figure 1: Warehouse management decisions related with the picking of orders.

whole pallet, which is typical from Business-to-Business (B2B) contexts. Therefore, in B2B scenarios, sometimes strict order picking is necessary. Other picking strategies also include zone picking and wave picking. In zone picking, a picker is assigned to a particular area in the warehouse, where the picker is responsible of picking only the items within that zone, while in wave picking, multiple picking routes are synchronized. Petersen (2000) performed a comparison of the main picking strategies: strict order picking, order batching, zone picking, and picking with waves.

In this survey, we focus our attention on a family of optimization problems that appears in a warehouse when the order-picking strategy is determined by the order batching policy. In particular, according to Figure 1, we review all operational decisions that occur in manual order picking systems (strategic/mechanization), in single-floor warehouses (strategic/floors), rectangular-shaped design (strategic/layout) that present only parallel and crossing aisles (tactical/aisles), which is probably the most common warehouse configuration in the literature. This family of problems has received growing interest from practitioners in operational research over the last few decades.

The rest of the paper is organized as follows: in Section 2 we define the family of problems denoted as "order batching problems", its main characteristics and common activities related to the problems within this family. In Section 3 we introduce a new taxonomy for the problems belonging to this family, based on four criteria: online/offline, single/multiple pickers, the optimized objective function, and the tasks handled. Then, in Section 4, we review in detail the literature of all journal articles found related to order batching. Closely related to this section, in Appendix A, we classify the papers related to order batching reviewed in this survey, following the proposed taxonomy. Finally, in Section 5, we present our conclusions and future perspectives on the field reviewed.

2. Order batching problems

The order batching family of problems groups all optimization problems whose main goal is to determine the best way to perform an efficient picking operation through the use of the batching strategy. The order batching strategy belongs to the operational decisions in a warehouse and consists of grouping several orders into the same batch before starting the picking. This indicates that items belonging to orders in the same batch must be retrieved together (i.e., on the same picking route). Each batch is restricted to contain a maximum capacity that might be measured in: weight, volume, number of items, or number of orders.

The batching strategy has led practitioners in the field of operations research to a wide range of related optimization problems. However, this strategy itself is not enough to define an applied optimization problem. Therefore, every problem that belongs to this family, in addition to following the pick-by-batch strategy, must face additional and separate optimization problems within it, which are closely related to any of the operational decisions presented in Figure 1. Nevertheless, the batching strategy deeply influences the rest of operational decisions: new constraints must be taken into consideration; each picking route is more lengthy and complicated, since it needs to consider more products; orders collected together have to be sorted while picking or in a later process, since more than one order is collected together; waiting strategies for the arrival of new orders might be considered; the assignment of batches to pickers directly influences how workload is balanced; determining the batch to select next also has additional challenges, since orders within the same batch might have different due dates or priorities; among others.

Additionally, it is important to note that many tactical and strategic decisions related to order batching problems are usually stated in the instances within the data sets, either

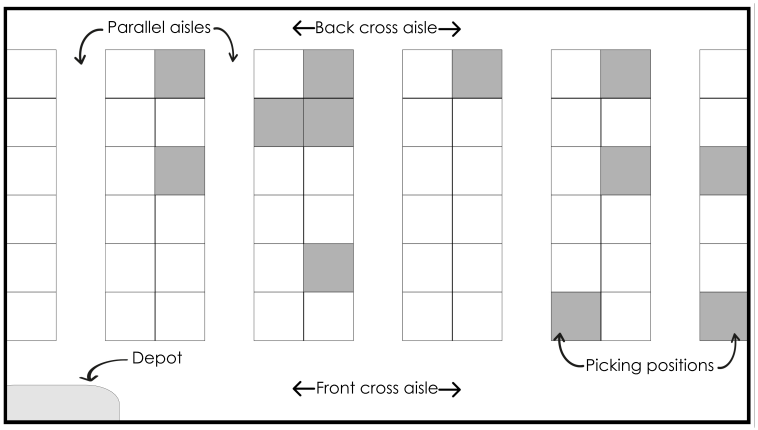


Figure 2: Example of the layout of the logistic warehouse studied in this paper.

in the information about the characteristics of the warehouse or in the information about the orders.

It is important to notice that in this paper we are focusing on picker-to-part problems that occur in rectangular-shaped warehouses, where the picking is usually performed manually. These warehouses are usually formed by a set of parallel aisles which contain several picking positions at each side of the aisle, and one or more (typically two) cross aisles, which allow the pickers to change from one parallel aisle to another. In Figure 2, we depict an example of a typical layout of the warehouse studied. Particularly, the warehouse represented in the figure has two cross aisles (one at the front and another one at the back) and five parallel aisles with twelve picking positions each (six at each side of the aisle). Picking routes start and end at a particular point of the warehouse denoted as the depot. In this example there is only one depot, placed in the leftmost corner of the front cross aisle. However, the depot can also be placed in the center or in the right corner of a cross aisle. Furthermore, sometimes more than one depot might exist. In the example of the figure, we have also highlighted in gray some picking positions, as an example of the positions that a picker must visit on a single picking route.

2.1. Operational decisions involved in the picking

In this section, we describe each of the operational decisions that might be addressed when handling any of the optimization problems belonging to the order batching family. Notice that some papers avoid studying some of the following tasks by handling a simplified variant of the problem or fixing a particular strategy for any of them. Specifically, the most common processes / tasks involved in the picking, together with the order batching task, are: waiting, selecting, sequencing, assigning, routing, and sorting.

The waiting task consists of determining the time that an available picker must wait to start a new route. This time is generally known in the literature as the "time window". The rationale behind this idea is that the longer the picker waits, the more orders are available in the system, which helps to construct more compact batches.

The selecting / sequencing task consists of determining the order in which the available batches are collected. When only one batch is selected as the next one to be collected, the task is generally known as "selecting". Otherwise, if all available batches are prioritized / sorted, the task is usually known as "sequencing".

The assigning task consists of determining which picker, among the available ones, is assigned to the next batch to be collected. This task only makes sense in scenarios with multiple pickers and is closely related to the balance of workload of the pickers.

The routing task consists of determining the route (i.e., the sequence of steps) that a picker must follow through the warehouse to collect all items in the orders contained in the assigned batch. The starting and finishing point for the routes is named the depot and is the place where items are dropped once the picker has collected them. As mentioned above, the routing strategy has a profound impact on the results obtained for order batching problems. Determining the route is necessary to calculate the distance traveled by the pickers or the time required to collect the items.

The sorting task consists of defining the strategy followed to sort the items picked in orders. Notice that when following a batching strategy, items from different orders are collected at the same picking route by the same picker, and they have to be classified later into orders (pick and sort), unless the picking cart has separate bins for each order (sort while pick).

It is worth mentioning that in the vast literature related to order batching problems, it is also possible to find variants of the problem that consider other activities, not explored in detail in this review, that can be the subject of optimization.

2.2. Objectives pursued

The target of any of the optimization problems belonging to the order batching family can be formalized through the use of different objective functions, not necessarily in conflict. Depending on the number of objectives pursued, optimization problems are traditionally classified in: single-objective optimization (only one objective function is optimized) or multi-objective optimization (two or more, in conflict, objective functions are optimized). However, in the context of order batching problems, further than the number of objectives that are optimized, we can classify the problems depending on the number of tasks that they are needed to solve in order to find a solution to the problem. If only the batching task is addressed, we denote the problem as "simple". In contrast, when more than one task is optimized, we denote the problem as "joint" (Chen et al., 2015; Shavaki & Jolai, 2021; Feng & Hu, 2021). This means that the variables that represent a solution contain the information necessary to complete the tasks being pursued. In these contexts, the algorithms proposed assign values to all variables that represent the solution simultaneously. The most common task optimized together with batching is the routing task (Hong & Kim, 2017; Valle et al., 2017). However, we can also find simultaneous optimizations of batching with selecting / sequencing (Menéndez et al., 2017a; Jiang et al., 2018) or assigning (Matusiak et al., 2017; Ardjmand et al., 2020), among others.

Objective functions are usually related to the maximization or minimization of a particular dimension. The most common ones are: time, distance, workload balance, or cost. However, some other minority objectives can also be found in the literature.

The time dimension is probably the most studied one and it includes aspects such as: the time required to collect an order, a batch or a group of orders (Rubrico et al., 2011; Henn, 2012; Zhang et al., 2017; Chen et al., 2018; Gil-Borrás et al., 2020b); the time that orders or pickers wait before starting their picking (Zhang et al., 2017; Henn, 2012; Gil-Borrás et al., 2020a); the total time that an order remains in the system (Gil-Borrás et al., 2020b; Tang & Chew, 1997; Chew & Tang, 1999), also known in the literature as throughput time (Le-Duc & De Koster, 2007; Van Nieuwenhuyse et al., 2007; Yu & De Koster, 2009); the delay with respect to a due date, known in the literature as tardiness (Chen et al., 2015; Henn & Schmid, 2013; Zhao et al., 2019; Henn, 2015; Menéndez et al., 2017a; Scholz et al., 2017); the handing time before the expected due date, known in the literature as earliness

Elsayed et al. (1993); or the blocking time of pickers (Chen et al., 2013, 2016; Hahn & Scholz, 2017). Closely related, the distance dimension measures the distance traversed by pickers when collecting orders (Öncan, 2015; Pérez-Rodríguez & Hernández-Aguirre, 2015). However, this dimension can be easily transformed into time (Jarvis & Mc. Dowell, 1991; Henn & Wäscher, 2012). In the case where multiple pickers are considered, a common dimension is related to the workload of the pickers measured in either: number of orders processed, distance traversed, number of items retrieved, total time retrieving items, etc. (Chen et al., 2016; Zhang et al., 2017; Huang et al., 2018; Mohring et al., 2020; Gil-Borrás et al., 2021). Other minority dimensions identified in the literature include the amount of work in progress (Hong, 2019), or the orders picked per unit of time (Hong, 2019). Finally, some authors have studied the economic aspects associated with picking operations, and in this case, the dimension to minimize is the cost (Miguel et al., 2019; Pinto & Nagano, 2019; Tian et al., 2019; Zhang et al., 2018; Bukchin et al., 2012).

3. Taxonomy and classification of order batching problems

Once the order batching family of problems has been described, many different specific problems can be found in the literature, depending on the objective pursued, the tasks optimized, the warehouse characteristics, the types of products handled, the availability of information, the number of pickers, etc. In this paper, we propose a taxonomy to classify all variants of order batching problems present in the literature based on the constraints considered, the objective function tackled, and the tasks (decision variables) optimized. For each criterion, several subcategories have been outlined. Each subcategory is represented by the text in parentheses, and the particular classification in the taxonomy of a specific problem is composed as the assignment of several subcategories simultaneously.

1. **Constraints**: among the different constraints that can be identified in the order batching literature, we have selected the two most significant. However, many others can be found, depending on the variant studied.
    - **Availability of information**: it indicates when the information related to the orders is available for the optimization process.
        - **Offline** (OFF): a problem is considered offline when all information about the orders to process is already available when the batching process starts.
        - **Online** (ON): a problem is considered online when the information about all orders to process is not fully available when the batching process starts (i.e., orders arrive to the system dynamically).
    - **Number of pickers**: it indicates the number of people who are simultaneously working on the picking task.
        - **Single picker** (SP): a problem is considered "single picker" when the picking task in the warehouse is performed by only one operator.
        - **Multiple pickers** (MP): a problem is considered "multi-picker" when the picking task in the warehouse is carried out by two or more operators.
2. **Objective functions**: the objective function represents the subject of optimization. Particularly, we have collected the dimensions measured by all the objective functions identified in the literature of order batching. It is important to note that some objectives are closely related one to each other, and sometimes, minimizing one of them might also minimize another. Specifically, there exists an equivalence between the two most studied objective functions in the literature (distance and time) when the travel velocity is constant. This fact has been pointed out by several authors (Jarvis & Mc. Dowell, 1991; Henn & Wäscher, 2012).

- **Distance** (DI): units of length that operators need to traverse to collect all items in the processed orders.
- **Picking time** (PT): units of time needed to perform the picking task, when collecting items in the orders processed in the warehouse.
- **Cost** (CO): unit of value that measures an economic indicator related to the picking operation in the warehouse.
- **Tardiness** (TA): units of time corresponding to the delay in handling an order with respect to a predefined due date. In this category, we also include the earliness, which indicates the anticipation of serving an order with respect to its due date, measured in units of time.
- **Completion time** (CT): units of time needed to collect all orders that arrived at the warehouse, including picking time and time waiting for the arrival of new orders (waiting time).
- **Turnover time** (TT): units of time that an order remains in the system (i.e., difference between the instant in the time when the order is served and the instant in the time when the order arrives, which needs to be known).
- **Workload balance** (WB): units of effort that indicate the differences among the amounts of work performed by different operators. It is usually measured in time; however, other dimensions could be used, such as: distance traversed, number of orders or batches retrieved, etc.
- **Blocking time** (BT): units of time that a picker waits for before achieving a particular task, blocked by the operation of another picker or machine (i.e., extracting items from a particular picking position; using the depot; etc.). This objective is also known in the literature as congestion.

3. **Decision variables**: the solution to an optimization problem is expressed through the values assigned to a set of variables. In the context of order batching problems, depending on the number of processes/tasks studied, it is possible to solve a single optimization problem (i.e., batching) or more than one at the same time (i.e., batching together with Sequencing/Assigning/Routing/Waiting). Therefore, the variables that represent a solution to an order batching problem might be different depending on the number of tasks studied. We have defined a category for each of the possible processes/tasks studied:
   - **Batching** (B): set of variables that represent the group of orders in batches.
   - **Sequencing** (S): set of variables that represent the sequence in which batches are collected.
   - **Assigning** (A): set of variables that represent the assignment of each batch to a picker.
   - **Routing** (R): set of variables that represent the route to follow by the picker to collect a batch.
   - **Waiting** (W): set of variables that represent the waiting time of each picker before starting a new route.

Notice that we have avoided the inclusion of several tasks such as: sorting, packaging, or scheduling, to ease the taxonomy. However, the taxonomy can be easily extended in the future by including such or other tasks.

In Figure 3, we graphically represent the proposed taxonomy, where the considered constraints, objective functions, and variables are represented. As far as constraints are

concerned, we can observe that the offline version is a special case of the online one. Furthermore, for any particular instant in time, we can transform an online variant into an offline one, supposing that no further orders will arrive to the system. Similarly, in the case of the number of pickers, the single-picker variant is a particular case of the multiple-pickers one. Since the single-picker variant considers that there is only one picker in the warehouse, batches must be sequenced and collected one by one. In this sense, approaches for multiple pickers can be used to solve single picker problems. Therefore, in single-picker variants there are not blocking situations in the aisles, in the picking positions or in the depot. On the other hand, multiple picker variants consider two or more pickers to work simultaneously in the picking-order process. This context reveals not only possible blocking situations, but also other necessities, such as deciding the assignment of batches to pickers or balancing the workload among the pickers.

Among the objective functions compiled, note that we have indicated only the dimension used to measure the objective function, not indicating whether the optimization target is an average value, a total value, or a maximum / minimum value. The objectives are divided into three different groups, the first group (i.e., tardiness, picking time, distance, and cost) contains those objectives present in any online / offline and single-picker / multiple-pickers variant. The second group (i.e., completion time and turnover time) contains those objectives that only make sense when considering an online variant. Finally, the third group (i.e., blocking time and workload balance) contains those objectives which only make sense when considering a multiple-pickers variant.

The taxonomy presented in Figure 3, represents a general framework that can be used to classify all optimization problems currently present in the state of the art of order batching problems. Additionally, it can be extended further in the future by including new constraints, objectives, or tasks with its associated variables to represent them. It is also important to note that, depending on the objective pursued, this taxonomy could also be used to classify single-objective or multi-objective optimization problems.

The taxonomy can be used, through the notation in parentheses introduced for each subcategory, to label order batching problems. Therefore, we propose to identify each optimization problem with a tag composed of four subtags (one per subcategory) separating them with hyphens. For instance, OFF-SP-TA-B would stand for an offline variant of the problem, which considers a single picker and looks to optimize the tardiness through the optimization of the variables related to the batching task.

Finally, given the taxonomy, we have related the categories in the taxonomy with the acronym of the problem and the number of papers which handle each particular variant. Note that some of the variants identified in Figure 3 have never been studied in the literature. Additionally, we have separated those problems that only look for the optimization of the batching task (denoted as "Simple") from those that look for the optimization of two or more tasks (denoted as "Joint").

4. State of the art

In this section, we review the most relevant approaches proposed in the literature of order batching. Particularly, we have organized this review, dividing the analysis of contributions into the categories introduced in the taxonomy presented in Section 3, obtaining four groups: Offline Single Picker (Section 4.1), Offline Multiple Pickers (Section 4.2), Online Single Picker (Section 4.3), and Online Multiple Pickers (Section 4.4). Within each category, we identify different problems, depending on the tasks studied: batching, routing, sequencing, assigning, or waiting. When several tasks are optimized simultaneously, the problem could be considered as a special case of the joint order batching problem.

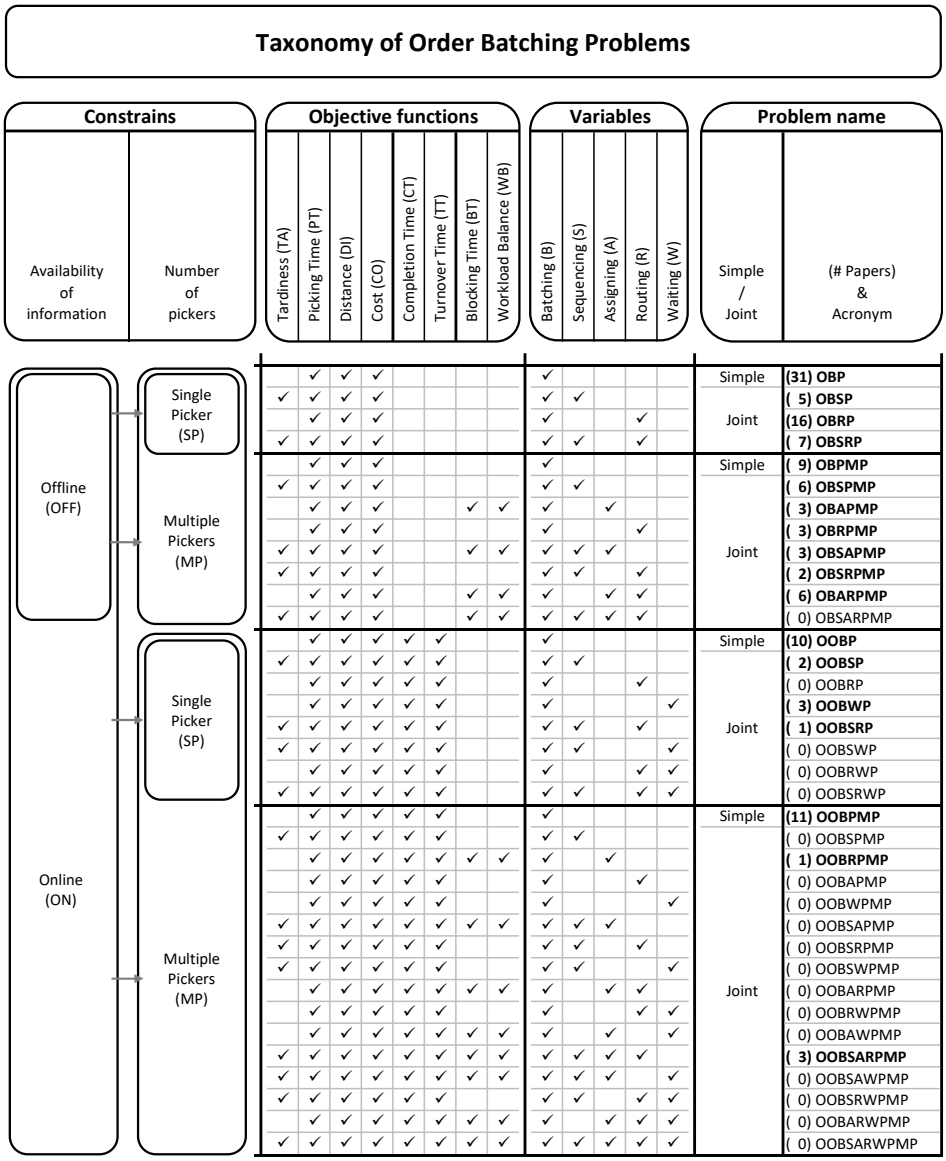


Figure 3: Taxonomy for classifying the order batching optimization problems in the literature.

For each problem identified, we compile in a different table all the contributions found in a journal indexed in the Journal Citation Reports (JCR) or in the Scimago Journal & Country Rank (SJR), sorted in chronological order. In each row of the tables, we report the reference to the paper, the objective function studied, and the routing and batching strategies proposed. Furthermore, for each of the proposals, we denote if the proposed strategies in the paper include heuristic algorithms, exact algorithms, or a combination of both. Notice that sometimes more than one algorithm is used for the same task (routing / batching), and more than one objective function is studied. Finally, to complement this information, we describe any notable aspect of the warehouse considered.

To end this section, we perform a synthesis of the review performed (see Section 4.5).

4.1. Offline / Single picker

In this section, we review the state of the art of offline order batching variants with a single picker. In these variants, all orders to collect are known before the process starts and there is only one picker in the warehouse.

4.1.1. Order Batching Problem (OBP)

We have classified under this category those papers that only consider the optimization of the batching task. In Table 1, we report each identified contribution, the objective function studied, and the algorithms proposed for the routing and batching tasks, together with the type of algorithm (Heuristic/Exact). As we can observe, most of the contributions study the minimization of the distance or the minimization of the picking time as objective function.

The first approach for the OBP was proposed in Elsayed (1981). In this case and other similar ones (Elsayed & Unal, 1989; Pan & Liu, 1995) the routing task was handled by an Automated Storage / Retrieval System (AS/RS), while the batching task was performed with basic heuristics. Several authors studied the influence of the routing strategy on the problem, comparing different routing algorithms (Hwang & Kim, 2005; Dukic & Oluic, 2007; Albareda-Sambola et al., 2009) in combination with the same batching proposal. Most of the algorithms for the routing task are traditional heuristics such as S-shape, Largest Gap, or Combined. It is worth mentioning that Menéndez et al. (2017b) proposed an extension of the Combined method. Similarly, most of the approaches for the batching task are also heuristic algorithms and, there are only a few exceptions of exact algorithms applied for solving the batching task (Muter & Öncan, 2015; Öncan, 2015; Lenoble et al., 2018). The first metaheuristic approach for the batching was a Genetic Algorithm introduced in Hsu et al. (2005).

Finally, with respect to the structure of the warehouse, most of the papers deal with a single-block warehouse, however some of them also considered a two or more blocks warehouse (Scholz & Wäscher, 2017; Yang et al., 2020).

Publications related to OBP in a JCR/SJR indexed journal	Objective Function				Routing		Batching	
	Distance	Picking Time	Cost	Completion Time	Type of Algorithm	Algorithm	Type of Algorithm	Algorithm
Elsayed (1981)	✔				-	AS/RS	H	SEED
Elsayed & Unal (1989)		✔			-	AS/RS	H H	EQ SL
Gibson & Sharp (1992)	✔				H	SS-OW	H H H	FCFS 4D-SFC SMD
Pan & Liu (1995)		✔			-	AS/RS	H	SEED
Rosenwein (1996)		✔			H	SS	H	SEED
De Koster et al. (1999b)		✔			H H	SS LG	H H H H	FCFS C&W EQ SL+C&W+EQ
Hsu et al. (2005)	✔				H	SS	H	GA
Hwang & Kim (2005)		✔			H H H	SS RE MP	H	SLCA
Chen & Wu (2005)	✔				H	SS	M	ARM+MM
Dukic & Oluic (2007)	✔				H H H H H E	SS RE MP LG CP DP	H H	FCFS C&W
Bozer & Kile (2008)	✔				H	SS	H H	FFEBBA NTS
Albareda-Sambola et al. (2009)		✔			H H H	SS LG CO	H	VND
Henn et al. (2010)	✔				H H	SS LG	H H	ILS RBAS
Hsieh & Huang (2011)	✔				H H E	SS MLI DP	H H H H H	KMS SOMBA ARA PSO SOP
Henn & Wäscher (2012)		✔			H H	SS LG	H H	TS ABHC
Menéndez et al. (2015)		✔			H	CO	H	GVNS
Muter & Öncan (2015)			✔		H H E	SS RE MP	E	TCGA

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E); Mixed (M); Not Defined (-). **Routing:** Automatic Storage and Retrieval System (AS/RS); S-Shape One Way (SS-OW); S-Shape (SS); Largest Gap (LG); Return (RE); Mid-Point (MP); Composite (CP); Dynamic Programming (DP); Combined (CO); Maximum Loop Insertion (MLI). **Batching:** Seed (SEED); EQUAL (EQ); Small and Large (SL); First Come First Served (FCFS); 4-dimensional Space Filling Curve (4D-SFC); Sequential Minimum Distance (SMD); Clarke & Wright (C&W); Genetic Algorithm (GA); Single-Link Clustering Algorithm (SLCA); Association Rule Mining (ARM); Mathematical Model (MM); First Fit-Envelope Based Batching Algorithm (FFEBBA); Normalized Time Saving (NTS); Variable Neighborhood Descent (VND); Iterated Local Search (ILS); Rank-Based Ant System (RBAS); K-Means Strategy (KMS); Self-Organization Map Batching Algorithm (SOMBA); Association Rule Algorithm (ARA); Particle Swarm Optimization (PSO); Strict-Order Picking (SOP); Tabu Search (TS); Attribute-Based Hill Climber (ABHC); General Variable Neighborhood Search (GVNS); Tailored Column Generation Algorithm (TCGA).

Continued on the next page.

Publications related to OBP in a JCR/SJR indexed journal	Objective Function				Routing		Batching	
	Distance	Picking Time	Cost	Completion Time	Type of Algorithm	Algorithm	Type of Algorithm	Algorithm
Öncan (2015)	✔				H H E	SS RE MP	E H	MM ILS+TT
Pérez-Rodríguez & Hernández-Aguirre (2015)		✔			H	SS	H H H	EDA GA MA
Koch & Wäscher (2016)	✔				H	SS	H	GGA
Menéndez et al. (2017b)		✔			H H H	SS LG CO	H	MS+VNS
Lenoble et al. (2017)				✔	-	VC	E	MM
Scholz & Wäscher (2017)	✔				H H H H E	SS LG CO AA DP	H	ILS
Cano et al. (2018)	✔				H	SS	H	GA
Lenoble et al. (2018)				✔	-	VLM	H	SA
Žulj et al. (2018)	✔				H H	SS LG	H	ALNS+TS
Van Gils et al. (2018)	✔				H H H E	SS LG RE DP	H H H	FCFS SEED SV
Nicolas et al. (2018)				✔	-	VLM	E H H H H	MM SV SA TS GA
Cano (2019)	✔				H	SS	H	GA
Yang et al. (2020)	✔				H H	SS CO	H E	TS MM
Yang et al. (2021)		✔			-	MRS	H	HGA

Acronym key · Type of algorithm: Heuristic (H); Exact (E); Not Defined (-). **Routing:** S-Shape (SS); Return (RE); Mid-Point (MP); Largest Gap (LG); Combined (CO); Vertical Carousel (VC); Aisle by Aisle (AA); Dynamic Programming (DP); Vertical Lift Module (VLM); Mobile Rack System (MRS). **Batching:** Mathematical Model (MM); Iterated Local Search (ILS); Tabu Thresholding (TT); Estimation of Distribution Algorithm (EDA); Genetic Algorithm (GA); Memetic Algorithm (MA); Grouping Genetic Algorithm (GGA); Multi-Start (MS); Variable Neighborhood Search (VNS); Simulated Annealing (SA); Adaptive Large Neighborhood Search (ALNS); Tabu Search (TS); First Come First Served (FCFS); Seed (SEED); Saving (SV); Hybrid Genetic Algorithm (HGA).

Table 1: Publications related to OBP.

4.1.2. Order Batching and Sequencing Problem (OBSP)

We have classified under this category those papers which consider the optimization of the batching and sequencing tasks. In Table 2, we report each paper identified, the objective function studied, and the algorithms proposed for the routing and batching/sequencing tasks, together with the type of algorithm (Heuristic/Exact). As we can observe, for this problem, minimizing the tardiness (a penalty associated with the orders handled after an expected date/time) is one of the most studied objective functions. Elsayed et al. (1993)

also considered the minimization of earliness together with tardiness.

Most proposals under this category achieve the batching and the sequencing tasks simultaneously with the same algorithm. The well-known Earliest Due Date (EDD) strategy has been used by several authors (Henn & Schmid, 2013; Bustillo et al., 2015; Menéndez et al., 2017a) as a naive but effective technique for constructing an initial solution to the problem, usually improved later with a metaheuristic. Elsayed et al. (1993) introduced an Optimal Release Times strategy (Fry et al., 1987) to introduce time windows (i.e., delays between batches) with the aim of balancing the earliness / tardiness for each batch. Jiang et al. (2018) studied the minimization of the makespan of the last batch (which is equivalent to the completion time). In this case, the authors did not consider due dates, but sequencing was necessary due to the existence of a limited buffer space in the sorting-packing area. Miguel et al. (2022) studied the minimization of the total operational cost as an evolution of a previous work devoted to OBP (Miguel et al., 2019). They considered that the products could be stored at different heights, and the earliness and tardiness are considered factors that influence the cost.

Publications related to OBSP in a JCR/SJR indexed journal	**Objective Function**			**Routing**		**Batching & Sequencing**	
	Tardiness	**Cost**	**Completion Time**	**Type of algorithm**	**Algorithm**	**Type of algorithm**	**Algorithm**
Elsayed et al. (1993)	✔			-	AS/RS	H	PL
Henn & Schmid (2013)	✔			H H E	SS LG MM	H H E	EDD+ILS EDD+ABHC MM
Menéndez et al. (2017a)	✔			H H H	SS LG CO	H	EDD+GVNS
Jiang et al. (2018)			✔	H	SS	H	CSA
Miguel et al. (2022)		✔		H	SS	H	HEA

Acronym key · Type of algorithm: Heuristic (H); Exact (E); Not Defined (-). **Routing:** Automatic Storage and Retrival System (AS/RS); S-Shape (SS); Largest Gap (LG); Combined (CO). **Batching:** Priority List (PL); Earliest Due Date (EDD); Iterated Local Search (ILS); Attribute-Based Hill Climber (ABHC); Cumulative-Seed Algorithm (CSA); Hybrid Evolutionary Algorithm (HEA); General Variable Neighborhood Search (GVNS).

Table 2: Publications related to OBSP.

4.1.3. Order Batching and Routing Problem (OBRP)

We have classified under this category those papers that consider the optimization of batching and routing tasks. In Table 3, we report each paper identified, the objective function studied, and the algorithms proposed for routing and batching tasks, together with the type of algorithm (Heuristic/Exact). As we can observe, the minimization of the distance is the most studied objective function. Gademann & Velde (2005) studied for the first time the minimization of the total travel time for this problem.

Several exact approaches have been introduced for the routing task (Gademann & Velde, 2005; Hong et al., 2012b; Matusiak et al., 2014; Zuniga et al., 2015; Hong & Kim, 2017; Ardjmand et al., 2019; Oxenstierna et al., 2021). Among them, the approach introduced by Matusiak et al. (2014) optimally solved the routing task with an A* algorithm (Hart

et al., 1968). In addition, Ho & Tseng (2006) used a metaheuristic algorithm, for the first time, for the routing task. Zuniga et al. (2015) handled the routing task by modeling the problem as a traveling salesman problem, using a previous formulation (Gutin & Punnen, 2006).

On the other hand, many seed methods have been studied for the batching task. Particularly, Ho & Tseng (2006) and Ho et al. (2008), conducted a deep analysis of 90 and 154 seed variants, respectively, for the batching task. Zuniga et al. (2015) studied the minimization of the total travel time by combining an optimization algorithm and a simulation model. They used the EDD rule to conform and sequence the batches. However, in this case, the objective function studied is not affected by the sequencing task. Therefore, this paper can also be classified as a single-picker offline OBP.

Despite the fact that most papers study a standard warehouse with a single block and a single depot, Matusiak et al. (2014) and Lin et al. (2016) considered the existence of two blocks. Matusiak et al. (2014) also studied the presence of narrow aisles. Kübler et al. (2020) considered too the existence of multiple blocks in the warehouse, and solved an additional problem that consists of the relocation of products in the warehouse. Briant et al. (2020) and Schiffer et al. (2022) studied several real warehouse instances with multiple blocks. Furthermore, Oxenstierna et al. (2021) studied the minimization of the traveled distance in six different irregular warehouse layouts. Finally, Schiffer et al. (2022) considered the existence of multiple depots and a variable multiblock layout.

4.1.4. Order Batching, Sequencing, and Routing Problem (OBSRP)

We have classified under this category those papers that consider the optimization of the batching, sequencing, and routing tasks. In Table 4, we report each paper identified, the objective function studied, and the algorithms proposed for the routing and batching/sequencing tasks, together with the type of algorithm (Heuristic/Exact). As we can observe in Table 4, the minimization of tardiness (Elsayed & Lee, 1996; Azadnia et al., 2013; Chen et al., 2015) and cost (Tsai et al., 2008; Miguel et al., 2019; Pinto & Nagano, 2019) are the most studied objective functions for this problem. Tsai et al. (2008) studied the minimization of the retrieval operation cost, calculated as the sum of the travel cost and the penalties associated to the earliness and tardiness. The most recent approach was introduced in Jiang et al. (2022) by studying, for the first time, the total completion time for this problem.

There are exact approaches for both, the routing and the batching tasks. Notice that the batching is usually performed together with the sequencing. The first approach for the problem was proposed in Elsayed & Lee (1996), where the authors introduced a Mixed-Integer Linear Programming model to formulate the problem as a whole. Additionally, they also achieved an additional task, the storage of products on the shelves in the same picking tour. Tsai et al. (2008) proposed a Genetic Algorithm (GA) to simultaneously handle batching and sequencing tasks and a second GA to optimize the routing. Jiang et al. (2022) proposed a mathematical model, based on a previous one introduced in Manjeshwar et al. (2009), to jointly solve the batching, routing, and sequencing tasks. In this paper, the authors also considered the sorting task after the picking.

All the approaches considered the existence of one block, except Chen et al. (2015) that studied the problem for two blocks in the warehouse.

4.2. Offline / Multiple pickers

In this section, we review the state of the art of offline order batching variants with multiple pickers. In these variants, all orders to collect are known before the process starts, and there are two or more pickers in the warehouse.

Publications related to OBRP in a JCR/SJR indexed journal	Objective Function			Routing		Batching	
	Picking Time	Distance	Cost	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Gademann & Velde (2005)	✔			E	B&P	E	B&P
Ho & Tseng (2006)		✔		H H	LG LG+SA	H	SEED
Ho et al. (2008)		✔		H H	LG LG+SA	H	SEED
Kulak et al. (2012)		✔		H H	SV+2-opt NN+Or-opt	H	TS+CA
Hong et al. (2012b)		✔		H E	SS MM	H H H H H E	FCFS SEED C&W RSBBA LPR MM
Matusiak et al. (2014)			✔	E	A*	H	SA
Zuniga et al. (2015)	✔			E	MM	H	EDD
Cheng et al. (2015)		✔		H	PSO+ACO	H	PSO+ACO
Lin et al. (2016)		✔		H	PSO	H	PSO
Hong & Kim (2017)		✔		H E	SS MM	H H H H H E	FCFS Seed C&W RSBBA LPR MM
Pferschy & Schauer (2018)		✔		H H H E	FIH+3-Opt CIH+3-Opt RIH+3-Opt MM	H E	SEED MM
Ardjmand et al. (2019)		✔		E H H H	MM GA SA GA+SA	E H H H	MM GA SA GA+SA
Kübler et al. (2020)		✔		H	NN+2-opt	H	DE-PSO
Briant et al. (2020)		✔		E	CG	E	CG
Oxenstierna et al. (2021)		✔		E	MM	H	SEED
Schiffer et al. (2022)		✔		E	B&PR	E	B&PR

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E). **Routing:** S-Shape (SS); Largest Gap (LG). **Batching:** Branch and Price (B&P); Branch and Prune (B&PR); Simulated Annealing (SA); Seed (SEED); Saving (SV); Nearest Neighbor (NN); Tabu Search (TS); Clustering Algorithm (CA); First Come First Served (FCFS); Clarke & Wright (C&W); Mathematical Model (MM); Linear Programming Relaxation (LPR); Route Selection–Based Batching Algorithm (RSBBA); Earliest Due Date (EDD); Particle Swarm Optimization (PSO); Ant Colony Optimization (ACO); Genetic Algorithm (GA); Discrete Evolutionary Particle Swarm Optimization (DE-PSO); Column Generation (CG); Farthest Insertion Heuristic (FIH); Cheapest Insertion Heuristic (CIH); Random Insertion Heuristic (RIH).

Table 3: Publication Related to OBRP.

Publications related to OBSRP in a JCR/SJR indexed journal	Objective Function			Routing		Batching & sequencing	
	Tardiness	Cost	Completion Time	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Elsayed & Lee (1996)	✔			E	AS/RS+B&B	H H H	EDD+NSR EDD+SPTR EDD+MCR
Tsai et al. (2008)		✔		H	GA	H	GA
Azadnia et al. (2013)	✔			H	GA	H	GA+MINWAL
Chen et al. (2015)	✔			H	ACO	H	GA
Miguel et al. (2019)		✔		E H	MM MA	E H	MM MA
Pinto & Nagano (2019)		✔		H	GA	H	EDD+GA
Jiang et al. (2022)			✔	H E	SS MM	H E	SEED MM

Acronym key · Type of algorithm: Heuristic (H); Exact (E). **Routing:** S-Shape (SS). **Batching:** Branch and Bound (B&B); Ant Colony Optimization (ACO); Genetic Algorithm (GA); Earliest Due Date (EDD); Seed (SEED); Memetic Algorithm (MA); Mining Association Rules with Weighted Items (MINWAL); Mathematical Model (MM); Nearest Schedule Rule (NSR); Shortest Processing Time Rule (SPTR); Most Common Location Rule (MCR)

Table 4: Publications related to OBSRP.

4.2.1. Order Batching Problem with Multiple Pickers (OBPMP)

This category compiles all works where only the batching is optimized but there exist multiple pickers in the warehouse. In Table 5, we report each paper identified, the objective function studied, and the algorithms proposed for the routing and batching tasks, together with the type of algorithm (Heuristic/Exact). In this case, the minimization of the picking time is the most studied objective function.

Most of the algorithmic proposals are based on heuristic algorithms, however, it is possible to find some exact algorithms for the routing (De Koster et al., 1999a; Gademann et al., 2001; Menéndez et al., 2017c) but also for the batching (Gademann et al., 2001; Yang, 2022). Ruben & Jacobs (1999) studied the influence of the several storage strategies in combination with the batching ones. While Petersen (2000) and Gademann et al. (2001) considered a picking strategy with waves, where a couple of batches are picked simultaneously (i.e., in the same wave) by a group of pickers. Petersen (2000) also explored other picking policies such as: strict order picking, sequential zoning, or batch zoning. Additionally, the impact of storage assignment policies was also reviewed in Yang (2022).

De Koster et al. (1999a) considered a real warehouse with two blocks and narrow aisles. Similarly, other approaches, such as Van Gils et al. (2016), or Cergibozan & Tasan (2020) also considered the existence of two blocks.

Publications related to OBPMP in a JCR/SJR indexed journal	Objective Function			Routing		Batching	
	Distance	Picking Time	Cost	Type of algorithm	Algorithm	Type of algorithm	Algorithm
De Koster et al. (1999a)		✔		H H H E	SS LG CO DP	H H H	FCFS SEED C&W
Ruben & Jacobs (1999)		✔		H	SS-OW	H H H H H	RB FFDA SMD FFEBBA FFCBBA
Petersen (2000)			✔	H	CP	H	FCFS
Gademann et al. (2001)		✔		E	DP	M	B&B+2-Opt
Pan et al. (2015)		✔		-	RCS	H	GGA
Van Gils et al. (2016)	✔			H H H H H	SS LG RN AA LKH	H H	FCFS SEED
Menéndez et al. (2017c)		✔		H E	CO DP	H	PVNS
Cergibozan & Tasan (2020)	✔			H H H	SS MP RE	H H	GA GA+PSO
Yang (2022)			✔	-	RMFS	E	MM

Acronym key · Type of algorithm: Heuristic (H); Exact (E); Mixed (M); Not Defined (-). **Routing:** S-Shape (SS); Largest Gap (LG); Combined (CO); Dynamic programming (DP); S-Shape one Way (SS-OW); Composite (CP); Roller Conveyor System(RCS); Aisle by Aisle (AA); Return (RE); Lin-Kernighan-Helsgaun (LKH); Mid-Point (MP); Robotic Mobile Fulfillment System (RMFS). **Batching:** First Come First Served (FCFS); Seed (SEED); Clarke & Wright (C&W); Random Batching (RB); First-Fit-Decreasing Algorithm (FFDA); Sequential Minimum Distance (SMD); First Fit-Envelope Based Batching Algorithm (FFEBBA); First Fit-Class Based Batching Algorithm (FFCBBA); Branch and Bound (B&B); Group Genetic Algorithm (GGA); Parallel Variable Neighborhood Search (PVNS); Genetic Algorithm (GA); Particle Swarm Optimization (PSO); Mathematical Model (MM).

Table 5: Publications related to OBPMP.

4.2.2. Order Batching and Sequencing Problem with Multiple Pickers (OBSPMP)

This category compiles all works where the batching and sequencing tasks are optimized in a warehouse with multiple pickers. In Table 6, we report each paper identified, the objective function studied, and the algorithms proposed for the routing and batching/sequencing tasks, together with the type of algorithm (Heuristic/Exact). In this case, the minimization of the completion time was the most studied objective function. Also, it is possible to find a multiobjective approach (Huang et al., 2018), which studies the completion time and the workload of pickers (by trying to balance the total number of items per batch and the total picking time per zone).

Among the proposals, it is possible to find heuristic (Hong et al., 2012a; Cano et al., 2021) and exact (Hong et al., 2012a; Žulj et al., 2021) algorithms for the routing and for the batching/sequencing tasks. Also, in Huang et al. (2018), the authors proposed a mixed model that combines a Genetic Algorithm with a Mathematical Model. There are two previous papers (Feng & Hu, 2021; Hofmann & Visagie, 2021) where the routing was

calculated using a Roller Conveyor System (RCS).

For this problem, several special cases of warehouses were considered. Hong et al. (2012a) studied the existence of narrow aisles. Žulj et al. (2021) considered the use of different picking zones with a robot assigned to each zone, and Hofmann & Visagie (2021) studied the existence of a single aisle which contains a conveyor belt. Also, Feng & Hu (2021) studied a fresh food processing warehouse, which handled the activity of cleaning and packing the food before storing it on shelves.

Publications related to OBSPMP in a JCR/SJR indexed journal	Objective Function				Routing		Batching & Sequencing	
	Picking Time	Completion Time	Tardiness	Workload Balance	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Hong et al. (2012a)	✔				E H	MM SS-OW	E H	MM SA
Huang et al. (2018)		✔		✔	-	-	M	GA+MM
Cano et al. (2021)		✔			H	SS	H	GGA
Žulj et al. (2021)			✔		E H	MM OH	E H	MM ALNS+NEH
Feng & Hu (2021)		✔			-	RCS	H H	SEED+GA SEED+SA
Hofmann & Visagie (2021)		✔			-	RCS	H	GH

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E); Mixed (M); Not Defined (-). **Routing:** S-Shape (SS); S-Shape one Way (SS-OW); Optimal Heuristic (OH); Roller Conveyor System (RCS). **Batching:** Mathematical Model (MM); Simulated Annealing (SA); Group Genetic Algorithm (GGA); Adaptive Large Neighborhood Search (ALNS); Nawaz, Enscore, and Ham Algorithm (NEH); Genetic Algorithm (GA); Seed (SEED); Greedy Heuristics (GH).

Table 6: Publications related to OBSPMP.

4.2.3. Order Batching and Assigning Problem with Multiple Pickers (OBAPMP)

We have classified under this category those works where the batching and assigning tasks are optimized in a scenario with multiple pickers. In Table 7, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing and batching/assigning tasks, together with the type of algorithm (Heuristic/Exact). In this case, three different objective functions have been studied: minimization of the distance, minimization of the picking time, and the minimization of the completion time.

In the three approaches identified for this variant, the batching and assigning tasks were simultaneously handled. Matusiak et al. (2017) proposed an Adaptive Large Neighborhood Search algorithm for jointly solving both tasks in a multiple-block scenario. While, Ardjmand et al. (2018) proposed three different methods depending on the size of the instance, including an exact model solved with a solver. Similarly, Wagner & Mönch (2022) proposed an Integer Linear Programming model for the batching and assigning tasks as an extension of the previous model introduced in Gademann & Velde (2005).

4.2.4. Order Batching and Routing Problem with Multiple Pickers (OBRPMP)

We have classified under this category those works where the batching and routing tasks are optimized in a scenario with multiple pickers. In Table 8, we compile the previous

Publications related to OBAPMP in a JCR/SJR indexed journal	Objective Function			Routing		Batching & Assigning	
	Distance	Picking Time	Completion Time	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Matusiak et al. (2017)			✔	H	AA	H	ALNS
Ardjmand et al. (2018)		✔		E E	MM DP	E H H	MM LD+PSO PSA+ACO
Wagner & Mönch (2022)	✔			H H E	SS LG MM	E H	MM VNS+FFD

Acronym key · Type of algorithm: Heuristic (H); Exact (E). **Routing:** Aisle-by-Aisle (AA); Dynamic Programming (DP); S-Shape (SS); Largest Gap (LG). **Batching:** Adaptive Large Neighborhood Search (ALNS); Mathematical Model (MM); Lagrangian Decomposition (LD); Particle Swarm Optimization (PSO); Parallel Simulated Annealing (PSA); Ant Colony Optimization (ACO); Variable Neighborhood Search (VNS); First Fit Decreasing (FFD).

Table 7: Publications related to OBAPMP.

works found for this problem, the objective function studied, and the algorithms proposed for routing and batching tasks, together with the type of algorithm (Heuristic/Exact). In this case, the minimization of the distance, the cost and the completion time have been studied.

Armstrong et al. (1979) studied the minimization of the completion time in a semi-automated warehouse with a conveyorized order-picking system, with one picker per aisle who is in charge of placing the collected products on a conveyor. Additionally, they allowed the possibility of splitting orders in different batches and the existence of the same product in multiple aisles. Yousefi Nejad et al. (2021) studied the minimization of the total cost associated with the picking of orders. They proposed an improvement of the mathematical model introduced in Cortés Achedad et al. (2017) to jointly solve batching and routing tasks. They also proposed three heuristic approaches for larger instances, and a scenario with multiple pickers (2 to 10) and variable capacity of the picking devices. Finally, Atchade-Adelomou et al. (2021) studied the minimization of the total distance traveled by 2 to 4 robots, by simultaneously considering the batching and routing tasks. To that aim, they used a completely novel approach based on a classical hybrid quantum algorithm (Variational Quantum Eigensolver) which was compared in different quantum simulators.

4.2.5. Order Batching, Sequencing and Assigning Problem with Multiple Pickers (OBSAPMP)

We have classified under this category those works where the batching, sequencing, and routing tasks are optimized in a scenario with multiple pickers. In Table 9, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing and batching/sequencing/assigning tasks, together with the type of algorithm (Heuristic/Exact). In this case, only the minimization of tardiness has been studied as an objective function.

Henn (2015) introduced a mathematical model for the problem, but the execution of the model on a solver resulted impossible. Therefore, two variants of the Variable Neighborhood

Publications related to OBRPMP in a JCR/SJR indexed journal	Objective Function			Routing & Batching	
	Distance	Cost	Completion Time	Type of algorithm	Algorithm
Armstrong et al. (1979)			✔	E	BD+MM
Yousefi Nejad et al. (2021)		✔		E H H H	MM GA PSO ABC
Atchade-Adelomou et al. (2021)	✔			H	VQE

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E). **Routing & Batching:** Bender's Decomposition (BD); Mathematical Model (MM); Genetic Algorithm (GA); Particle Swarm Optimization (PSO); Artificial Bee Colony (ABC); Variational Quantum Eigensolver (VQE).

Table 8: Publications related to OBRPMP.

Search methodology were proposed for the batching task. Sequencing and assigning tasks are considered within the neighborhoods of the VNS, either in the local search procedures or in the shake. Scholz et al. (2017) adapted a previous mathematical model (Henn, 2015) for the problem. Also, they proposed a Variable Neighborhood Descent algorithm for jointly solving the batching, sequencing, and assigning tasks. Kuhn et al. (2021) studied the minimization of the total tardiness of all orders in a warehouse with multiple blocks. This time, the authors considered not only batching and sequencing activities, but also delivery operations from the warehouse to the shops. In fact, the tardiness is calculated after the delivery of the orders to the shops, not when the orders are handled to the depot of the warehouse. They proposed several heuristics and a mathematical model for the integrated Order Batching and Vehicle Routing Problem.

Publications related to OBSAPMP in a JCR/SJR indexed journal	Objective Function	Routing		Batching & Sequencing & Assigning	
	Tardiness	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Henn (2015)	✔	H H	SS LG	H	VNS
Scholz et al. (2017)	✔	H	LKH	H H	ESD+VND SEED+VND
Kuhn et al. (2021)	✔	H E	SS MM	H E	C&W+ALNS+LPTR MM

Acronym key · **Type of algorithm**: Heuristic (H); Exact (E). **Routing:** S-Shape (SS); Largest Gap (LG); Lin–Kernighan–Helsgaun (LKH). **Batching:** Variable Neighborhood Search (VNS); Earliest Start Date (ESD); Seed (SEED); Variable Neighborhood Descent (VND); Clarke & Wright (C&W); Adaptive Large Neighborhood Search (ALNS); Largest Processing Time Rule (LPTR); Mathematical Model (MM).

Table 9: Publications related to OBSAPMP.

4.2.6. Order Batching, Sequencing, and Routing Problem with Multiple Pickers (OBSRPMP)

We have classified under this category those works where the batching, sequencing, and routing tasks are optimized in a scenario with multiple pickers. In Table 10, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing/batching/sequencing tasks, together with the type of algorithm (Heuristic/Exact). In this case, the minimization of tardiness and the minimization of distance have been studied.

Particularly, Cano et al. (2020) studied the minimization of several objectives such as: the travel distance, the total tardiness / earliness, and a combination of travel time and tardiness / earliness in an aggregated function. To that aim the authors proposed four different mathematical models inspired in previous works (Scholz et al., 2016; Scholz & Wäscher, 2017; Scholz et al., 2017; Valle et al., 2017). The authors studied a warehouse with multiple blocks and more than one height levels. On the other hand, Cals et al. (2021) studied the minimization of the number of orders with tardiness, proposing a method based on Deep Reinforcement Learning (i.e., Reinforcement Learning and Deep Neural Networks) inspired by the ideas proposed in Zhang et al. (2012) to optimize the batching, routing, and sequencing tasks.

Publications related to OBSRPMP in a JCR/SJR indexed journal	**Objective Function**		**Routing & Batching & Sequencing**	
	Distance	**Tardiness**	**Type of algorithm**	**Algorithm**
Cano et al. (2020)	✔	✔	E	MM
Cals et al. (2021)		✔	H	DRL

Acronym key · Type of algorithm: Heuristic (H); Exact (E). **Routing & Batching & Sequencing:** Mathematical Model (MM); Deep Reinforcement Learning (DRL).

Table 10: Publications related to OBSRPMP.

4.2.7. Order Batching, Assigning, and Routing Problem with Multiple Pickers (OBARPMP)

We have classified under this category those works in which batching, assigning, and routing tasks are optimized in a scenario with multiple pickers. In Table 11, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for routing and batching / assignment tasks, together with the type of algorithm proposed (Heuristic/Exact). In this case, the minimization of the distance and the picking time are the most studied objective functions. Additionally, a biobjective function to minimize the makespan together with the number of pickers is introduced.

Valle et al. (2016) formulated three mathematical models to jointly solve batching and routing problems. The models considered the existence of multiple blocks. Later, Valle et al. (2017) proposed an evolution based on two Branch-and-Cut approaches to jointly solve the batching and routing tasks. Van Gils et al. (2019) also introduced a mathematical model for the problem, adapted from Valle et al. (2017), to solve the batching, routing, and assigning tasks. They considered the existence of two blocks in the warehouse. Another mathematical model run with a solver was proposed in Valle & Beasley (2020) to

find optimal solutions to the joint order batching and routing problem, which also considered the assignation of batches to pickers. Their approach considered two variants of the routing, depending on the reversal constraint (i.e., pickers can perform a U-turn in the parallel aisles or not). They also considered single and multiple blocks warehouse scenarios. Ardjmand et al. (2020) jointly solved the batching, assigning, and routing tasks using a hybrid method which combines Column Generation, Genetic Algorithm, and Artificial Neural Networks. Their approach was compared with the previous proposal introduced in Ardjmand et al. (2018). Finally, Rasmi et al. (2022) studied the minimization of the makespan together with the minimization of the number of active pickers in a biobjective approach. They proposed an Integer Linear Programming model to simultaneously solve the batching, assigning, and routing tasks. Also, they introduced a heuristic approach in which the batching task was solved with a k-means clustering algorithm (Lloyd, 1982), the routing task was tackled with the approach proposed in Ratliff & Rosenthal (1983), and the assigning task was solved with a mathematical model run in a commercial solver. The authors also compared several storage location assignment policies.

Publications related to OBARPMP in a JCR/SJR indexed journal	**Objective Function**			**Routing**		**Batching & Assigning**	
	Distance	**Picking Time**	**Others**	**Type of algorithm**	**Algorithm**	**Type of algorithm**	**Algorithm**
Valle et al. (2016)	✔			E	MM	E	MM
Valle et al. (2017)	✔			E	B&C	E	B&C
Van Gils et al. (2019)		✔		E E H	MM DP LKH	E H	MM ILS
Valle & Beasley (2020)	✔			E	MM	E	MM
Ardjmand et al. (2020)		✔		M	CG+GA+ANN	M	CG+GA+ANN
Rasmi et al. (2022)		✔	✔	E E	MM DP	E M	MM KMS+MM

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E); Mixed (M). **Routing:** Dynamic Programming (DP); Lin–Kernighan–Helsgaun (LKH). **Batching:** Mathematical Model (MM); Branch and Cut (B&C); Iterated Local Search (ILS); Column Generation (CG); Genetic Algorithm (GA); Artificial Neural Networks (ANN); K-Means Strategy (KMS).

Table 11: Publications related to OBARPMP.

4.3. Online / Single picker

In this section, we review the state of the art of online order batching variants with a single picker. In these variants, orders arrive to the system dynamically, i.e., once the picking process has already started. For each contribution, we review the distribution in the arrival of orders (when available). Additionally, it is considered that there is only one picker in the warehouse.

4.3.1. Online Order Batching Problem (OOBP)

We have classified under this category those works where the batching task is optimized in a scenario with dynamic arrival of orders. In Table 12, we compile the previous

works found for this problem, the objective function studied, and the algorithms proposed for the routing and batching tasks, together with the type of algorithm proposed (Heuristic/Exact). In this case, the turnover time and the picking time were the most studied objective functions. However, we can also find approaches studying the minimization of the completion time or cost.

Almost all the reviewed papers tackled the routing and the batching tasks with heuristic algorithms. Tang & Chew (1997) studied the minimization of the average turnover time through the reduction of the waiting time of orders once they are in the system. The arrival of orders follows a Poisson process, so the problem was modeled as an $E_n/G/c$ queueing system, where n denotes the batch size. Chew & Tang (1999) studied the minimization of the average turnover time through the reduction of service time and travel time. Again, the arrival of orders follows a Poisson process, and the problem was modeled as an $E_n/G/c$ queueing system, where n denotes the batch size. Le-Duc & De Koster (2007) studied the minimization of the average throughput time (i.e., the time that the order remains in the system before being served) in a two-block warehouse. The arrival of orders follows a Poisson process and the problem was modeled as a $M/G^k/1$ queueing system, where k denotes the batch size. Schleyer & Gue (2012) studied the minimization of the average throughput time and the optimal batch size for efficient picking. In this work, the authors considered that the arrival of orders is not restricted to be a Poisson process, but any stationary arrival stream of orders. The problem was modeled as a $G/G/1$ queueing system. Other approaches considering the arrival of orders following a Poisson process were Henn (2012); Xu et al. (2014) and Pérez-Rodríguez et al. (2015). Finally, Gil-Borrás et al. (2020b) studied the minimization of the total completion time of all orders, but also reported the maximum turnover time obtained. In fact, it can be considered as an extension of Gil-Borrás et al. (2018).

Publications related to OOBP in a JCR/SJR indexed journal	**Objective Function**					**Routing**		**Batching**	
	Picking Time	**Cost**	**Turnover Time**	**Completion Time**	**Others**	**Type of algorithm**	**Algorithm**	**Type of algorithm**	**Algorithm**
Tang & Chew (1997)			✔			H	SS-OW	H	FCFS
Chew & Tang (1999)			✔			H	SS	H	FCFS
Le-Duc & De Koster (2007)	✔					H	SS	H	FCFS
Schleyer & Gue (2012)	✔				✔	H	SS	E	DTM
Henn (2012)				✔		H H	SS LG	H	ILS
Xu et al. (2014)	✔					H	SS	H	FCFS+VTWB
Pérez-Rodríguez et al. (2015)			✔			H	SS	H	CEDA
Zhang et al. (2018)		✔				H	SS	H H	C&W ILS
Gil-Borrás et al. (2018)			✔			H	SS	H	BVNS
Gil-Borrás et al. (2020b)			✔	✔		H	SS	H	GRASP+VND

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E). **Routing:** S-Shape One Way (SS-OW); S-Shape (SS); Largest Gap (LG). **Batching:** First Come First Served (FCFS); Discrete-Time Models (DTM); Iterated Local Search (ILS); Variable Time Window Batching (VTWB); Continuous Estimation of Distribution Algorithm (CEDA); Clarke & Wright (C&W); Basic Variable Neighborhood Search (BVNS); Greedy Randomized Adaptive Search Procedure (GRASP); Variable Neighborhood Descent (VND).

Table 12: Publications related to OOBP.

4.3.2. Online Order Batching and Waiting Problem (OOBWP)

We have classified under this category those works where the batching and waiting tasks are optimized in a scenario with dynamic arrival of orders. In Table 13, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing, batching, and waiting tasks, together with the type of algorithm proposed (Heuristic/Exact). In this case, the minimization of the picking time, completion time, and cost have been studied.

Bukchin et al. (2012) studied the minimization of the average costs associated with the tardiness and overtime of the pickers. For the first time, they introduced a new waiting method that accurately calculates the departure time of each picker based on previous information. Then, they developed an approximate model to determine the waiting strategy for future arrivals of orders. Giannikas et al. (2017) studied the minimization of the average completion time. However, they considered a variant of the problem which allows the addition of new orders to a batch being collected. They consider a Variable Time-Window strategy based on the number of orders (1, 5, 10, 15, and 20) arriving to the system. Finally, Gil-Borrás et al. (2020a) studied the minimization of the picking time, but they also reported the completion time of collecting all orders. The authors evaluated and compared several time-window strategies: a No-Waiting strategy, a Fixed Time Window strategy based on time (3, 6, and 12 minutes), and a Variable Time Window strategy based on the number of orders arrived to the system (4, 8, 16 orders).

Publications related to OOBWP in a JCR/SJR indexed journal	**Objective Function**			**Routing**		**Batching**		**Waiting**	
	Picking Time	**Cost**	**Completion Time**	**Type of algorithm**	**Algorithm**	**Type of algorithm**	**Algorithm**	**Type of algorithm**	**Algorithm**
Bukchin et al. (2012)		✔		-	-	M	FCFS+MDP	M	FCFS+MDP
Giannikas et al. (2017)			✔	E	DP	H	GA	H	VTW
Gil-Borrás et al. (2020a)	✔		✔	H	SS	H	FCFS GH	H H H	NW FTW VTW

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E); Mixed (M); Not Defined (-). **Routing:** S-Shape (SS); Dynamic Programming (DP). **Batching:** First Come First Served (FCFS); Markov decision process (MDP); Genetic Algorithm (GA); Greedy Heuristic (GH). Waiting: No Waiting (NW); Variable Time Window (VTW); Fixed Time Window (VTW).

Table 13: Publications related to OOBWP.

4.3.3. Online Order Batching, and Routing Problem (OOBRP)

We have classified under this category those works where the batching and routing tasks are optimized in a scenario with dynamic arrival of orders. In Table 14, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing and batching tasks, together with the type of algorithm proposed (Heuristic/Exact). In this case, the minimization of the distance and the cost, have been tackled.

Ene & Öztürk (2012) studied the minimization of the travel cost expressed as a function of the travel time. They proposed two approaches to jointly solve batching and routing tasks. Additionally, in this paper, the authors also studied the storage problem by minimiz-

ing the warehouse transmissions with another Integer Programming model using GAMS (Boisvert et al., 1985). They considered a two-block warehouse. On the other hand, Li et al. (2016) studied the minimization of the total travel distance. They proposed an algorithm based on Ant Colony Optimization for jointly solving the batching and routing task. They considered warehouses with multiple blocks and a very large amount of orders (up to 10,000).

Publications related to OOBRP in a JCR/SJR indexed journal	Objective Function		Routing & Batching	
	Distance	Cost	Type of algorithm	Algorithm
Ene & Öztürk (2012)		✔	E H	MM GA
Li et al. (2016)	✔		H	ACO

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E). **Routing & Batching:** Mathematical Model (MM); Genetic Algorithm (GA); Ant Colony Optimization (ACO).

Table 14: Publications related to OOBRP.

4.3.4. Online Order Batching, Sequencing, and Routing Problem (OOBSRP)

We have classified under this category those works in which batching, sequencing, and routing tasks are optimized in a scenario with dynamic arrival of orders. In Table 15, we compile the only work found for this problem, the objective function studied, and the algorithms proposed for the routing and batching/sequencing tasks. In this case, the minimization of the picking time and the turnover time were tackled.

Specifically, Won & Olafsson (2005) studied the minimization of a combined objective function that considers the minimization of the picking time together with the minimization of the time that orders remain in the warehouse. They proposed a formulation for the joint order batching, sequencing and routing problem. Notice that the warehouse studied in this paper includes a depot at the end of each aisle and the travel distance is assumed to be calculated using the Tchebychev metric (Bozer et al., 1990) instead of the usual rectilinear metric.

4.4. Online / Multiple pickers

In this section, we review the state of the art of online order batching variants with multiple pickers. In these variants, orders arrive to the system dynamically (once the picking process has already started) to a warehouse with two or more pickers.

4.4.1. Online Order Batching Problem with Multiple Pickers (OOBPMP)

We have classified under this category those works where the batching task is optimized in a scenario with dynamic arrival of orders and multiple pickers. In Table 16, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing and batching tasks. In this case, the completion time was the most studied objective function. However, other objectives were also considered in the literature such as picking time, cost, workload balance, or turnover time.

Publications related to OOBSRP in a JCR/SJR indexed journal	Objective Function		Routing & Batching & Sequencing	
	Picking Time	Turnover Time	Type of algorithm	Algorithm
Won & Olafsson (2005)	✔	✔	E	MM

Acronym key · Type of algorithm: Exact (E). **Routing & Batching & Sequencing:** Mathematical Model (MM).

Table 15: Publications related to OOBSRP.

Yu & De Koster (2009) considered picking zones within the warehouse (each of them assigned to a different picker) and the arrival of orders was determined by a Poisson distribution. The warehouse had a random storage policy. The authors tackled the batching task with an approximation model based on the queueing network theory, and they used the S-Shape routing strategy. Van Nieuwenhuyse & De Koster (2009) studied a two-block warehouse where the arrival of orders followed a Poisson process and the problem was modeled as a $G/G/1$ and a $G/G/m$ queueing system. They proposed the use of different batching strategies based on waiting for the arrival of orders. Particularly, they proposed a Fixed Time Window Batching, consisting of waiting for a fixed amount of time, and a Variable Time Window Batching, consisting of waiting while there is available space in the batch. Also, they compared the pick-and-sort and sort-while-pick picking policies. Rubrico et al. (2011) studied a variant of the problem, named Online Rescheduling Problem with multiple pickers, by considering the existence of static and dynamic arrival of orders. Additionally, they introduced the constraint that newly arrived orders were composed of only one type of product. Zhang et al. (2017) studied the minimization the maximum completion time, also known as the turnover time of all orders, but also reported the average idle time per picker and the average workload. Chen et al. (2018) studied the minimization of the service time of a single order. In this case, they considered a multiple-block warehouse with narrow aisles. Also, they studied the possibility that orders could be split in several batches and that batches could be modified during picking. Similarly, Hojaghania et al. (2019) studied the minimization of the maximum turnover time and the idle time of pickers in a warehouse with different zones within the warehouse, each of them assigned to a picker. Zhang et al. (2021) studied a pondered objective function which includes the minimization of completion time needed to pick and delivery the orders, together with the minimization of the total delivery cost. The assignment of batches to pickers follows a first available picker rule. Shavaki & Jolai (2021) studied the minimization of the transportation cost of orders and jointly solved the delivery planning, by proposing two mathematical models, solved with a solver, which included: the assignment of trucks to docks, the departure time, and the route of the truck. Finally, Gil-Borrás et al. (2021) studied the minimization of picking time, the minimization of the completion time, and the minimization of the differences in the workload balance among pickers in a single-block warehouse.

Publications related to OOBPMP in a JCR/SJR indexed journal	Objective Function					Routing		Batching	
	Picking Time	Cost	Turnover Time	Completion Time	Workload Balance	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Yu & De Koster (2009)				✔		H	SS	H	QNT
Van Nieuwenhuyse & De Koster (2009)				✔		H	SS	H H	FTWB VTWB
Rubrico et al. (2011)				✔		H	SS	H	SDI+MRS
Zhang et al. (2017)			✔			H	SS	H	SEED+HRBA
Chen et al. (2018)				✔		H H	SS LG	H H H	GAS FTWB VTWB
Hojaghania et al. (2019)			✔			H	SS	H H	ACO ABC
Gil-Borrás et al. (2019)				✔		H H H	SS LG CO	H	BVNS
Alipour et al. (2020)				✔		H H	SS LG	H	ILS
Zhang et al. (2021)		✔		✔		H	SS	H	C&W
Shavaki & Jolai (2021)		✔				H	SS	H H H	HBS SV GA
Gil-Borrás et al. (2021)	✔			✔	✔	H	SS	H	MS+VND

Acronym key · Type of algorithm: Heuristic (H). **Routing:** S-Shape (SS); Largest Gap (LG); Combined (CO). **Batching:** Queueing Network Theory (QNT); Fixed Time Window Batching (FTWB); Variable Time Window Batching (VTWB); Steepest Descent Insertion (SDI); Multistage Rescheduling strategy (MRS); Seed (SEED); Hybrid Rule-Based Algorithm (HRBA); Green Area Strategy (GAS); Ant Colony Optimization (ACO); Artificial Bee Colony (ABC); Basic Variable Neighborhood Search (BVNS); Iterated Local Search (ILS); Clarke & Wright (C&W); Heuristic Based on Similarity (HBS); Saving (SV); Genetic Algorithm (GA); Multi-Start (MS): Variable Neighborhood Descent (VND).

Table 16: Publications related to OOBPMP.

4.4.2. Online Order Batching and Routing Problem with Multiple Pickers (OOBRPMP)

We have classified under this category those works where the batching and routing tasks are optimized in a scenario with dynamic arrival of orders and multiple pickers. In Table 17, we compile the only previous work found for this problem, the objective function, and the routing and batching strategies used. In this case, Leung et al. (2020) studied the minimization of the total travel time in a real scenario. They proposed a Genetic Algorithm which integrates the solution to the batching and routing tasks. They also considered the existence of multiple pickers (up to 18). Their proposal was integrated into a software system to manage the warehouse.

4.4.3. Online Order Batching, Sequencing, Assigning, and Routing Problem with Multiple Pickers (OOBSARPMP)

We have classified under this category those works where the batching, assigning, and routing tasks are optimized in a scenario with dynamic arrival of orders and multiple pickers.

In Table 18, we compile the previous works found for this problem, the objective function studied, and the algorithms proposed for the routing and batching tasks.

Particularly, Zhang et al. (2016) studied the minimization of the total service time, while maximizing the number of orders delivered without exceeding a predefined due date. In this paper, the sequencing and assigning tasks were inspired by a previous strategy for other problem introduced in Pratap et al. (2015). Particularly, they proposed several heuristic rules that combine the urgency of a batch and the workload balance of idle pickers. The arrival of orders is studied on a 2-hour time horizon. Later, Duda & Stawowy (2019) studied the minimization of the number of pickers together with the minimization of the distance traveled in an online scenario. To that aim, the authors introduced a weighted function which combines the two previous objectives. They proposed a Mixed-Integer Programming model to jointly solve the batching, sequencing, assigning, and picker routing tasks. The authors also proposed a heuristic approach based on Variable Neighborhood Search for solving the problem when the size of the instance is large. They considered an 8-hour time horizon. Finally, Schrotenboer et al. (2019) simultaneously studied the minimization of the total travel time and the picking cost though the use of a combined objective function. They proposed different Mixed-Integer Programming models to jointly solve the batching, sequencing, assigning, and routing tasks. The authors also proposed a constructive procedure together with an Adaptive Large Neighborhood Search heuristic for the problem. Additionally, in this paper, the authors integrated the restocking of returned products into regular order picking routes.

4.5. Synthesis of the review

In Section 4, we have reviewed and classified 122 papers (107 JCR / 15 SJR) related to Order Batching. In Figure 4, we present a bar chart in which all publications are classified per year and group of problem. Additionally, in Figure 5, we show another bar chart in which we can compare the number of papers per order batching variant.

As a first conclusion of the analysis performed, we observe that among the 36 variants of order batching problems identified in the taxonomy introduced in Figure 3, 18 of them have never been tackled in the literature. Among the studied variants, we observe that 48.36% of the papers deal with offline single-picker variants, 26.23% of the papers deal with offline multiple-pickers variants, 13.11% of the papers deal with online single-picker variants, and 12.30% of the papers deal with online multiple-pickers variants.

Furthermore, the offline variants of the problem have been studied further than the online ones (91 papers vs 31 papers). Similarly, single-picker variants have been studied further than multiple-pickers ones (75 papers versus 47 papers). The most studied variant is the offline version of the Order Batching Problem with a single picker, where the problem consists of the batching task only (31 papers).

Publications related to OOBRPMP in a JCR/SJR indexed journal	**Objective Function**	**Routing & Batching**	
	Distance	**Type of algorithm**	**Algorithm**
Leung et al. (2020)	✔	H	GA

Acronym key · **Type of algorithm:** Heuristic (H).
Routing & Batching: Genetic Algorithm (GA).

Table 17: Publications related to OOBRPMP.

Publications related to OOBSARPMP in a JCR/SJR indexed journal	Objective Function					Routing		Batching	
	Distance	Picking Time	Cost	Completion Time	Others	Type of algorithm	Algorithm	Type of algorithm	Algorithm
Zhang et al. (2016)				✔	✔	H	SS	H H	SEED C&W
Duda & Stawowy (2019)	✔				✔	E H	MM VNS	E H	MM VNS
Schrotenboer et al. (2019)		✔	✔			E H	MM ALNS	E H	MM ALNS

Acronym key · **Type of algorithm:** Heuristic (H); Exact (E). **Routing:** S-Shape (SS). **Batching:** Seed (SEED); Clarke & Wright (C&W); Mathematical Model (MM); Variable Neighborhood Search (VNS); Adaptive Large Neighborhood Search (ALNS).

Table 18: Publications related to OOBSARPMP.

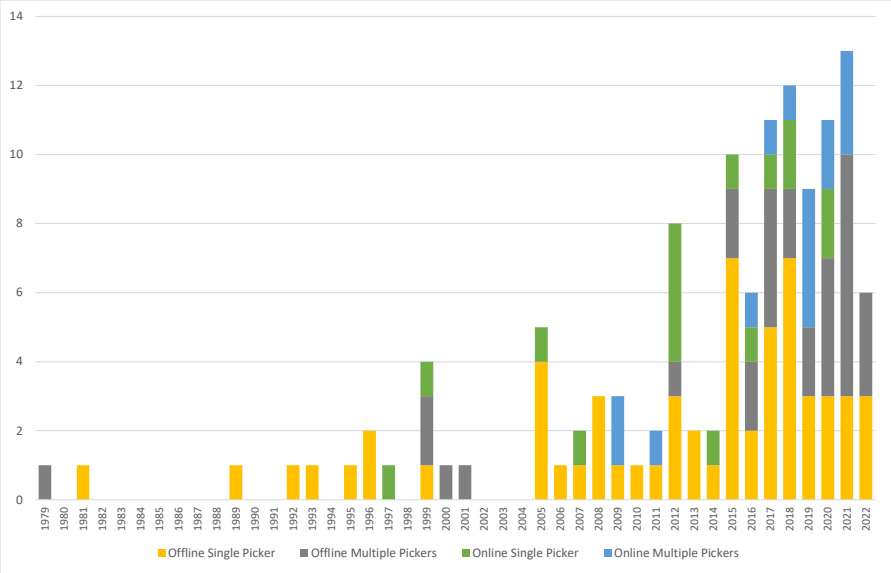


Figure 4: Publications reviewed classified by year and grouped by category.

As far as the warehouse studied is concerned, the most studied warehouse is a rectangular-shaped warehouse with a single depot and a single block (composed of parallel aisles) to perform the picking operation. Particularly, we have found 92 out of the 122 papers using this warehouse model. However, other authors studied different warehouse configurations, such as multiple blocks, irregular shapes, or storage in different heights, among others. Furthermore, 14 papers studied semi-automated warehouses.

All the papers compiled achieve the routing and batching tasks. However, only 29 papers study the sequencing task, 15 papers study the assigning task, and 3 papers study the waiting task.

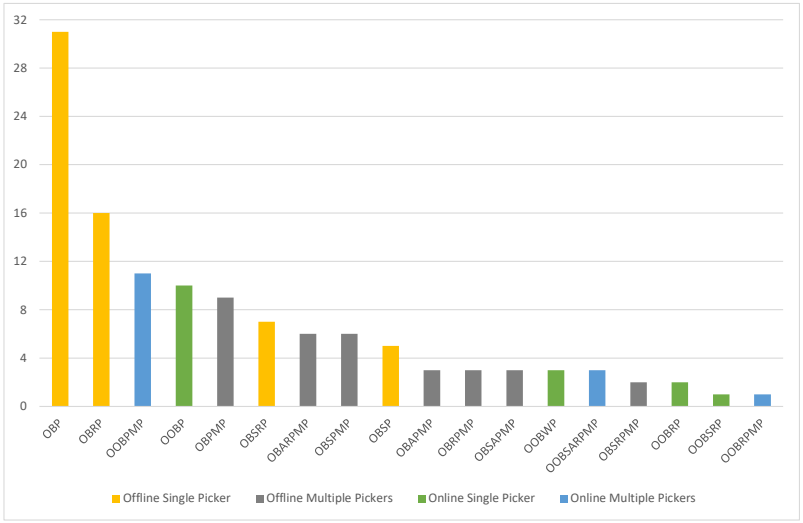


Figure 5: Publications reviewed classified by order batching variant and grouped by category.

A large number of papers explore the use of more than one routing strategy. In particular, all the papers reviewed include any kind of heuristic/metaheuristic strategy for the task. Among them, the S-Shape method is the most popular one (i.e., used in 62 out of 122 papers). On the other hand, 34.42% of the papers additionally propose the use of an exact approach, mainly based on a mathematical model or dynamic programming. As far as the batching task is concerned, metaheuristics are the most common strategies to tackle the task (60.65% of the studied papers included at least a metaheuristic method). However, simpler heuristics such as: seed, savings, and FCFS methods, are also very popular (36.06% of the studied papers included at least a heuristic method). Finally, the 27.04% of the studied papers proposed the use of any kind of exact method (usually a mathematical model). The sequencing and assigning tasks are usually handled together with the batching strategy. The waiting task has been little explored in the literature despite the fact that it has been shown to have a profound impact on the overall performance of the picking algorithms.

5. Conclusions and open issues

In this survey, we review the order batching family of optimization problems. This family compiles a group of well-known optimization problems related to the picking of orders in a warehouse, having in common that they consider the batching policy during the picking process. That is, orders received in a warehouse are grouped into batches before starting the picking process. We have properly defined the family of problems denoted as order batching problems, and we have identified the main tasks that have to be addressed to solve each particular variant of the problem within the order batching family. Then, we have proposed a taxonomy to classify all order batching problems. To the best of our knowledge, this is the first time a taxonomy to classify the order batching variants has been proposed. Following the proposed taxonomy, we have reviewed the previous literature related to order batching and classified all JCR and SJR papers found in the literature using the aforementioned taxonomy, on the basis of the particular problem tackled in each

paper. Finally, for each reference identified in the literature, we have briefly highlighted the strategies proposed for each of the main tasks associated to solve the problem variant handled in the paper. Next, we state our conclusions, open issues, and future research opportunities.

5.1. Conclusions

Order batching problems have been extensively studied by the scientific community in the last forty years, since the batching policy within a warehouse has been demonstrated to be a very effective strategy to perform the task of picking orders. As it happens with many scientific disciplines, practitioners in the field often handle simplified variants of interesting real problems to illustrate the performance of their algorithms instead of more realistic scenarios. Furthermore, it is common to find literature with simple variants of problems but very large data sets (or with very large instances within them) that are often unrealistic for the problem handled. In this sense, we believe that some of the variants related to order batching studied in the state of the art, such as the simple OBP (an offline variant with a single picker) which is the most studied variant in the literature, represents mainly a theoretical problem that helps practitioners to propose and validate new algorithms and ideas further than a real problem on its own. In contrast, online variants with multiple pickers are probably the most general versions of order batching and the closest ones to real-world scenarios related to order batching. It is important to remark that the offline version of order batching problem is a specific case of the online one, which can be considered as the general problem. Similarly, let us remember that the single-picker version of the problem is a specific case of the multiple-pickers one. Therefore, solving more general variants of the problem also provides solutions to the specific ones.

Despite the fact that it is possible to find more than a hundred publications related to order batching in top-level journals, as far as we know, practitioners have never introduced a proper taxonomy to clearly identify the gaps or the particular problem variant they are handling. This fact has obstructed researchers to identify previous works in the literature directly connected to their research, and therefore many articles lack a proper comparison of their findings with other previous proposals.

Proposing a taxonomy which classifies all order batching variants tackled in the previous literature is not an easy task. Any taxonomy might result incomplete when considering an exceptional / specific piece of work, and the criteria included in it are always full of controversy. However, there are some relevant aspects that are clearly identifiable in the literature of order batching problems such as: offline / online, single / multiple pickers, or the objective function being optimized. In this sense, we have tried to propose a taxonomy that gathers the characteristics of an optimization problem: constraints, variables, and objective functions. Furthermore, the proposed taxonomy is easily extensible, mainly by adding new constraints, variables, or objective functions.

From our point of view, there are two main groups to classify any order batching problem: Simple or Joint. We denote as "Simple" any variant of the problem which only handles the batching task (i.e., the optimization is restricted to the values of the variables that determine the batching). Similarly, we denote as "Joint" any variant of order batching that handles the batching task together with one or more additional tasks (i.e., optimization is not restricted to the variables that determine the batching, but also the variables which determine sequencing, routing, assigning, or waiting).

The most outstanding variants of the order batching problems, attending to the relevance, novelty, and number of references related to them, are the simple variants of order batching problems: simple Order Batching Problem, simple Order Batching Problem with Multiple Pickers, simple Online Order Batching Problem, simple Online Order Batching

Problem with Multiple Pickers. Additionally, there are some relevant joint versions such as: joint Order Batching, Sequencing, and Routing Problem, or joint Order Batching and Routing Problem. On the other hand, 18 out of the 36 identified variants of the problem have never been tackled. This is especially relevant in the case of the online variants of the problem.

As a final conclusion, we would like to highlight that order batching problems are a growing family of optimization problems of high economic interest for the industry. Given the large recent interest in related problems, the main objective of this paper is to set the foundations of order batching and organize the current state of the art of this family of problems, so the related literature can grow up properly, avoiding repetitions and establishing a clear comparison framework for future research proposals.

5.2. Open issues and future research opportunities

In this section, we highlight the main open issues related to order batching problems, including the gaps identified in the literature, the most realistic variants of the problem, the most influential tasks to address, and the more promising algorithmic strategies.

In the near future, practitioners interested in any problem related to order batching should start by identifying the particular gap in the taxonomy proposed in this review, which they are trying to cover. Moreover, any previous work on the variant discussed should be included in the literature review, and the new approach should be adequately compared with the previous ones. Since the taxonomy is designed to grow, in case of necessity, we also invite practitioners to extend it with new constraints, objective functions, or variables (tasks), making clear the new contribution to the literature. As we can observe from the taxonomy proposed and the classification of previous works, many variants of order batching remain unstudied, which opens a very large research opportunity. Particularly, most of the joint versions of order batching problems have not yet been studied.

We suggest that the research direction should be moving from the classical and more theoretical variants of order batching to more realistic variants. Particularly, the classical and most studied approach when dealing with order batching is related to the offline version of the problem with a single picker. However, this is only a special case of the general problem, that might consider multiple pickers and an online arrival of orders to the warehouse. We suggest that practitioners in the field focus on these more realistic variants of the problem.

The batching in isolation is an interesting task from a theoretical point of view. However, when dealing with any variant of order batching problems, it is necessary to study it in combination with other tasks, such as routing, sequencing, assigning, and waiting. All the previous work in the literature considers the routing; however, since we should focus on dealing with more realistic variants, further study should be performed in relation to the sequencing and assigning (when multiple pickers are available) and waiting (when there is a dynamic arrival of orders to the warehouse). Furthermore, we have discovered that the latter has a deep influence on the performance of the overall method and is by far the least studied activity. In the future, the inclusion of other tasks such as sorting the products after picking or refilling the shelves could also be considered.

Currently, there is a large collection of algorithms, either heuristics/metaheuristics or exacts to deal with almost any of the tasks related to order batching. However, the effectiveness of the proposed strategy depends on the particular task. Furthermore, some tasks might be as hard as others computationally speaking but, in practice, they might be much smaller than others (e.g., the size of the batching problem is usually larger than the size of the sequencing problem, since the number of orders is much larger than the number of

batches). Moreover, the size of the instance makes some of the tasks smaller enough to be handled with exact methods. Therefore, an effort should be made to develop matheuristic algorithms which include optimal results for particular tasks in combination with other heuristic techniques for the rest of the tasks. Similarly, the routing task, given a classical warehouse design, is optimally solved in a reasonable amount of time.

Finally, there are few studies about multi-objective optimization variants of order batching problems, either considering two or more of the objective functions identified in this paper or coupling the batching task together with other optimization problems.

Appendix A. Classification of the previous papers in the literature

Researchers have been proposing a wide range of optimization problems belonging to the order batching family of problems in rectangular-shaped warehouses with parallel aisles, by adding constraints, modifying the characteristics of the warehouse, or optimizing several objectives at the same time. To contribute to the literature of order batching, we have grouped all references found in the literature and reviewed in this paper under the best-known names and acronyms. Additionally, we have classified the optimization problems identified on the basis of the taxonomy introduced in Section 3. Notice that we only consider papers published in journals indexed in the Journal Citation Reports (JCR) or the Scimago Journal & Country Rank (SJR).

Taxonomy of order batching problems			
Category	**Reference**	**Category**	**Reference**
[Simple] / Order Batching Problem (OBP)			
OFF-SP-DI-B	Elsayed (1981)	OFF-SP-PT-B	Elsayed & Unal (1989)
OFF-SP-DI-B	Gibson & Sharp (1992)	OFF-SP-PT-B	Pan & Liu (1995)
OFF-SP-PT/DI-B	Rosenwein (1996)	OFF-SP-PT-B	De Koster et al. (1999b)
OFF-SP-DI-B	Hsu et al. (2005)	OFF-SP-PT-B	Hwang & Kim (2005)
OFF-SP-DI-B	Chen & Wu (2005)	OFF-SP-DI-B	Dukic & Oluic (2007)
OFF-SP-DI-B	Bozer & Kile (2008)	OFF-SP-PT-B	Albareda-Sambola et al. (2009)
OFF-SP-DI-B	Henn et al. (2010)	OFF-SP-DI-B	Hsieh & Huang (2011)
OFF-SP-PT-B	Henn & Wäscher (2012)	OFF-SP-PT-B	Menéndez et al. (2015)
OFF-SP-CO-B	Muter & Öncan (2015)	OFF-SP-PT-B	Pérez-Rodríguez & Hernández-Aguirre (2015)
OFF-SP-DI-B	Öncan (2015)	OFF-SP-DI-B	Koch & Wäscher (2016)
OFF-SP-PT-B	Menéndez et al. (2017b)	OFF-SP-CT-B	Lenoble et al. (2017)
OFF-SP-DI-B	Scholz & Wäscher (2017)	OFF-SP-DI-B	Cano et al. (2018)
OFF-SP-PT-B	Lenoble et al. (2018)	OFF-SP-DI-B	Žulj et al. (2018)
OFF-SP-DI-B	Van Gils et al. (2018)	OFF-SP-CT-B	Nicolas et al. (2018)
OFF-SP-DI-B	Cano (2019)	OFF-SP-DI-B	Yang et al. (2020)
OFF-SP-PT-B	Yang et al. (2021)		
[Joint] / Order Batching and Sequencing Problem (OBSP)			
OFF-SP-TA-BS	Elsayed et al. (1993)	OFF-SP-TA-BS	Henn & Schmid (2013)
OFF-SP-TA-BS	Menéndez et al. (2017a)	OFF-SP-CT-BS	Jiang et al. (2018)
OFF-SP-CO-BS	Miguel et al. (2022)		
[Joint] / Order Batching and Routing Problem (OBRP)			
OFF-SP-PT-BR	Gademann & Velde (2005)	OFF-SP-DI-BR	Ho & Tseng (2006)
OFF-SP-DI-BR	Ho et al. (2008)	OFF-SP-DI-BR	Kulak et al. (2012)
OFF-SP-DI-BR	Hong et al. (2012b)	OFF-SP-DI-BR	Matusiak et al. (2014)
OFF-SP-PT-BR	Zuniga et al. (2015)	OFF-SP-DI-BR	Cheng et al. (2015)
OFF-SP-DI-BR	Lin et al. (2016)	OFF-SP-DI-BR	Hong & Kim (2017)
OFF-SP-DI-BR	Pferschy & Schauer (2018)	OFF-SP-DI-BR	Ardjmand et al. (2019)
OFF-SP-DI-BR	Kübler et al. (2020)	OFF-SP-DI-BR	Briant et al. (2020)
OFF-SP-DI-BR	Oxenstierna et al. (2021)	OFF-SP-DI-BR	Schiffer et al. (2022)
[Joint] / Order Batching, Sequencing, and Routing Problem (OBSRP)			
OFF-SP-TA-BSR	Elsayed & Lee (1996)	OFF-SP-CO-BSR	Tsai et al. (2008)
OFF-SP-TA-BSR	Azadnia et al. (2013)	OFF-SP-TA-BSR	Chen et al. (2015)
OFF-SP-DI/CO-BSR	Pinto & Nagano (2019)	OFF-SP-CO-BSR	Miguel et al. (2019)
OFF-SP-CT-BSR	Jiang et al. (2022)		
[Simple] / Order Batching Problem with Multiple Pickers (OBPMP)			
OFF-MP-PT-B	De Koster et al. (1999a)	OFF-MP-DI-B	Ruben & Jacobs (1999)
OFF-MP-PT-B	Petersen (2000)	OFF-MP-PT-B	Gademann et al. (2001)
OFF-MP-PT-B	Pan et al. (2015)	OFF-MP-DI-B	Van Gils et al. (2016)
OFF-MP-PT-B	Menéndez et al. (2017c)	OFF-MP-DI-B	Cergibozan & Tasan (2020)
OFF-MP-CO-B	Yang (2022)		
[Joint] / Order Batching and Sequencing Problem with Multiple Pickers (OBSPMP)			

Continued on the next page

Category	Reference	Category	Reference
OFF-MP-PT-BS	Hong et al. (2012a)	OFF-MP-CT-BS	Huang et al. (2018)
OFF-MP-CT-BS	Cano et al. (2021)	OFF-MP-TA-BS	Žulj et al. (2021)
OFF-MP-CT-BS	Feng & Hu (2021)	OFF-MP-CT-BS	Hofmann & Visagie (2021)
[Joint] / Order Batching and Assigning Problem with Multiple Pickers (OBAPMP)			
OFF-MP-CT-BA	Matusiak et al. (2017)	OFF-MP-CT-BA	Ardjmand et al. (2018)
OFF-MP-DI-BA	Wagner & Mönch (2022)		
[Joint] / Order Batching and Routing Problem with Multiple Pickers (OBRPMP)			
OFF-MP-CT-BR	Armstrong et al. (1979)	OFF-MP-DI-BR	Atchade-Adelomou et al. (2021)
OFF-MP-CO-BR	Yousefi Nejad et al. (2021)		
[Joint] / Order Batching, Sequencing and Assigning Problem with Multiple Pickers (OBSAPMP)			
OFF-MP-TA-BSA	Henn (2015)	OFF-MP-TA-BSA	Scholz et al. (2017)
OFF-MP-TA-BSA	Kuhn et al. (2021)		
[Joint] / Order Batching, Sequencing and Routing Problem with Multiple Pickers (OBSRPMP)			
OFF-MP-DI/TA-BSR	Cano et al. (2020)	OFF-MP-TA-BSR	Cals et al. (2021)
[Joint] / Order Batching, Assigning and Routing Problem with Multiple Pickers (OBARPMP)			
OFF-MP-DI-BAR	Valle et al. (2016)	OFF-MP-DI-BAR	Valle et al. (2017)
OFF-MP-PT-BAR	Van Gils et al. (2019)	OFF-MP-DI-BAR	Valle & Beasley (2020)
OFF-MP-CT-BAR	Ardjmand et al. (2020)	OFF-MP-CT+NP-BAR	Rasmi et al. (2022)
[Simple] / Online Order Batching Problem (OOBP)			
ON-SP-TT-B	Tang & Chew (1997)	ON-SP-TT-B	Chew & Tang (1999)
ON-SP-CT-B	Le-Duc & De Koster (2007)	ON-SP-CT-B	Schleyer & Gue (2012)
ON-SP-CT-B	Henn (2012)	ON-SP-CT-B	Xu et al. (2014)
ON-SP-TT-B	Pérez-Rodríguez et al. (2015)	ON-SP-TT-B	Zhang et al. (2018)
ON-SP-TT-B	Gil-Borrás et al. (2018)	ON-SP-CT-B	Gil-Borrás et al. (2020b)
[Simple] / Online Order Batching and Waiting Problem (OOBWP)			
ON-SP-CO-BW	Bukchin et al. (2012)	ON-SP-CT-BW	Giannikas et al. (2017)
ON-SP-PT-BW	Gil-Borrás et al. (2020a)		
[Simple] / Online Order Batching, and Routing Problem (OOBRP)			
ON-SP-CO-BR	Ene & Öztürk (2012)	ON-SP-DI-BR	Li et al. (2016)
[Simple] / Online Order Batching, Sequencing, and Routing Problem (OOBSRP)			
ON-SP-PT+TT-BSR	Won & Olafsson (2005)		
[Simple] / Online Order Batching Problem with Multiple Pickers (OOBPMP)			
ON-MP-TT-B	Yu & De Koster (2009)	ON-MP-TT-B	Van Nieuwenhuyse & De Koster (2009)
ON-MP-CT-B	Rubrico et al. (2011)	ON-MP-CT-B	Zhang et al. (2017)
ON-MP-PT-B	Chen et al. (2018)	ON-MP-TT-B	Hojaghania et al. (2019)
ON-MP-CT-B	Gil-Borrás et al. (2019)	ON-MP-CT-B	Alipour et al. (2020)
ON-MP-PT-B	Zhang et al. (2021)	ON-MP-PT-B	Shavaki & Jolai (2021)
ON-MP-PT-B	Gil-Borrás et al. (2021)		
[Joint] / Online Order Batching and Routing Problem with Multiple Pickers (OOBRPMP)			
ON-MP-PT-BR	Leung et al. (2020)		
[Joint] / Online Order Batching, Sequencing, Assigning, and Routing Problem with Multiple Pickers (OOBSARPMP)			
ON-MP-PT+NO-BSAR	Zhang et al. (2016)	ON-MP-DI+NP-BSAR	Duda & Stawowy (2019)
ON-MP-PT+CO-BSAR	Schrotenboer et al. (2019)		

Table A.1: Classification of the journal papers related to Order Batching.

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*, 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*, 101–107.

Ardjmand, E., Bajgiran, O. S., & Youssef, E. (2019). Using list-based simulated annealing and genetic algorithm for order batching and picker routing in put wall based picking systems. *Applied Soft Computing*, *75*, 106–119.

Ardjmand, E., Ghalehkhondabi, I., Young II, W. A., Sadeghi, A., Sinaki, R. Y., & Weckman, G. R. (2020). A hybrid artificial neural network, genetic algorithm and column

generation heuristic for minimizing makespan in manual order picking operations. *Expert Systems with Applications*, *159*, 113566.

Ardjmand, E., Shakeri, H., Singh, M., & Bajgiran, O. S. (2018). Minimizing order picking makespan with multiple pickers in a wave picking warehouse. *International Journal of Production Economics*, *206*, 169–183.

Armstrong, R. D., Cook, W. D., & Saipe, A. L. (1979). Optimal batching in a semi-automated order picking system. *Journal of the operational research society*, *30*, 711–720.

Atchade-Adelomou, P., Alonso-Linaje, G., Albo-Canals, J., & Casado-Fauli, D. (2021). qrobot: A quantum computing approach in mobile robot order picking and batching problem solver optimization. *Algorithms*, *14*.

Azadnia, A. H., Taheri, S., Ghadimi, P., Mat Saman, M. Z., & Wong, K. Y. (2013). Order batching in warehouses by minimizing total tardiness: a hybrid approach of weighted association rule mining and genetic algorithms. *The Scientific World Journal*, *2013*, 246578.

Blanchard, D. (2010). *Supply chain management best practices*. John Wiley & Sons.

Boisvert, R. F., Howe, S. E., & Kahaner, D. K. (1985). Gams: A framework for the management of scientific software. *ACM Transactions on Mathematical Software (TOMS)*, *11*, 313–355.

Bozer, Y. A., & Kile, J. W. (2008). Order batching in walk-and-pick order picking systems. *International Journal of Production Research*, *46*, 1887–1909.

Bozer, Y. A., Schorn, E. C., & Sharp, G. P. (1990). Geometric approaches to solve the chebyshev traveling salesman problem. *IIE transactions*, *22*, 238–254.

Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A. L., & Ogier, M. (2020). An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, *285*, 497–512.

Bukchin, Y., Khmelnitsky, E., & Yakuel, P. (2012). Optimizing a dynamic order-picking process. *European Journal of Operational Research*, *219*, 335–346.

Bustillo, M., Menéndez, B., Pardo, E. G., & Duarte, A. (2015). An algorithm for batching, sequencing and picking operations in a warehouse. In *2015 International Conference on Industrial Engineering and Systems Management* (pp. 842–849). IEEE.

Cals, B., Zhang, Y., Dijkman, R. M., & Van Dorst, C. (2021). Solving the online batching problem using deep reinforcement learning. *Computers & Industrial Engineering*, *156*, 107221.

Cano, J. A. (2019). Parameters for a genetic algorithm: An application for the order batching problem. *IBIMA Business Review*, *2019*, 802597.

Cano, J. A., Correa-Espinal, A. A., & Gómez-Montoya, R. A. (2018). Solución del problema de conformación de lotes en almacenes utilizando algoritmos genéticos. *Información tecnológica*, *29*, 235–244.

Cano, J. A., Correa-Espinal, A. A., & Gómez-Montoya, R. A. (2020). Mathematical programming modeling for joint order batching, sequencing and picker routing problems in manual order picking systems. *Journal of King Saud University-Engineering Sciences*, *32*, 219–228.

Cano, J. A., Cortés Achedad, P., Campo, E. A., & Correa-Espinal, A. A. (2021). Solving the order batching and sequencing problem with multiple pickers: a grouped genetic algorithm. *International Journal of Electrical and Computer Engineering*, *11*, 2516–2524.

Cergibozan, Ç., & Tasan, A. S. (2020). Genetic algorithm based approaches to solve the order batching problem and a case study in a distribution center. *Journal of Intelligent Manufacturing*, (pp. 1–13).

Chen, F., Wang, H., Qi, C., & Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers and Industrial Engineering*, *66*, 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*, 640–685.

Chen, M.-C., & Wu, H.-P. (2005). An association-based clustering approach to order batching considering customer demand patterns. *Omega*, *33*, 333–343.

Chen, T.-L., Cheng, C.-Y., Chen, Y.-Y., & Chan, L.-K. (2015). An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. *International Journal of Production Economics*, *159*, 158–167.

Cheng, C.-Y., Chen, Y.-Y., Chen, T.-L., & Jung-Woon Yoo, J. (2015). Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem. *International Journal of Production Economics*, *170*, 805–814.

Chew, E. P., & Tang, L. C. (1999). Travel time analysis for general item location assignment in a rectangular warehouse. *European Journal of Operational Research*, *112*, 582–597.

Cortés Achedad, P., Gómez-Montoya, R. A., Muñuzuri, J., & Correa-Espinal, A. A. (2017). A tabu search approach to solving the picking routing problem for large-and medium-size distribution centres considering the availability of inventory and k heterogeneous material handling equipment. *Applied Soft Computing*, *53*, 61–73.

Coyle, J. J., Bardi, E. J., & Langley, C. J. (1996). *The management of business logistics* volume 6. West Publishing Company Minneapolis/St. Paul.

De Koster, R. B. M., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European journal of operational research*, *182*, 481–501.

De Koster, R. B. M., Roodbergen, K. J., & Van Voorden, R. (1999a). Reduction of walking time in the distribution center of De Bijenkorf. *Lecture Notes in economics and mathematical systems. New Trends in Distribution Logistics*, *480*, 215–234.

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

Drury, J. (1988). Towards more efficient order picking. *IMM monograph*, *1*.

Duda, J., & Stawowy, A. (2019). A VNS Approach for Batch Sequencing and Route Planning in Manual Picking System with Time Windows. *Lecture Notes in Computer Science. International Conference on Variable Neighborhood Search*, *12010*, 167–177.

Dukic, G., & Oluic, C. (2007). Order-picking methods: improving order-picking efficiency. *International Journal of Logistics Systems and Management*, *3*, 451–460.

Elsayed, E. A. (1981). Algorithms for optimal material handling in automatic warehousing systems. *The International Journal of Production Research*, *19*, 525–535.

Elsayed, E. A., & Lee, M. K. (1996). Order processing in automated storage/retrieval systems with due dates. *IIE transactions*, *28*, 567–577.

Elsayed, E. A., Lee, M. K., Kim, S., & Scherer, E. (1993). Sequencing and batching procedures for minimizing earliness and tardiness penalty of order retrievals. *The International Journal of Production Research*, *31*, 727–738.

Elsayed, E. A., & Unal, O. I. (1989). Order batching algorithms and travel-time estimation for automated storage/retrieval systems. *The International Journal of Production Research*, *27*, 1097–1114.

Ene, S., & Öztürk, N. (2012). Storage location assignment and order picking optimization in the automotive industry. *The international journal of advanced manufacturing technology*, *60*, 787–797.

Feng, X., & Hu, X. (2021). A heuristic solution approach to order batching and sequencing for manual picking and packing lines considering fatiguing effect. *Scientific Programming*, *2021*, 8863391.

Fry, T. D., Armstrong, R. D., & Blackstone, J. H. (1987). Minimizing weighted absolute deviation in single machine scheduling. *IIE transactions*, *19*, 445–450.

Gademann, N., Van Den Berg, J. P., & Van Der Hoff, H. H. (2001). An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE transactions*, *33*, 385–398.

Gademann, N., & Velde, V. S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, *37*, 63–75.

Giannikas, V., Lu, W., Robertson, B., & Mc. Farlane, D. (2017). An interventionist strategy for warehouse order picking: Evidence from two case studies. *Inter. Journal of Production Economics*, *189*, 63–76.

Gibson, D. R., & Sharp, G. P. (1992). Order batching procedures. *European Journal of Operational Research*, *58*, 57–67.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2018). New VNS Variants for the Online Order Batching Problem. *Lecture Notes in Computer Science*, *11328*, 89–100.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2019). Basic VNS for a variant of the Online Order Batching Problem. *Lecture Notes in Computer Science*, *12010*, 17–36.

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*, 295–325.

Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., & Duarte, A. (2021). A heuristic approach for the online order batching problem with multiple pickers. *Computers & Industrial Engineering*, *160*, 107517.

Gu, J., Goetschalckx, M., & Mc. Ginnis, L. F. (2007). Research on warehouse operation: A comprehensive review. *European journal of operational research*, *177*, 1–21.

Gudehus, T. (1973). Principles of order picking: operations in distribution and warehousing systems. (in german), w. *Girardet, Essen, Germany*, .

Gutin, G., & Punnen, A. P. (2006). *The traveling salesman problem and its variations* volume 12. Springer Science & Business Media.

Hahn, S., & Scholz, A. (2017). *Order Picking in Narrow-Aisle Warehouses: A Fast Approach to Minimize Waiting Times*. Technical Report Otto-von-Guericke University Magdeburg, Faculty of Economics and Management.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, *4*, 100–107.

Henn, S. (2012). Algorithms for on-line order batching in an order picking warehouse. *Computers and Operations Research*, *39*, 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*, 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*, 82–105.

Henn, S., & Schmid, V. (2013). Metaheuristics for order batching and sequencing in manual order picking systems. *Computers & Industrial Engineering*, *66*, 338–351.

Henn, S., & Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, *222*, 484–494.

Ho, Y. C., Su, T. S., & Shi, Z. B. (2008). Order-batching methods for an order-picking warehouse with two cross aisles. *Computers & Industrial Engineering*, *55*, 321–347.

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 Production Research*, *44*, 3391–3417.

Hofmann, F. M., & Visagie, S. E. (2021). The effect of order batching on a cyclical order picking system. In *International Conference on Computational Logistics* (pp. 252–268). Springer.

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*, *28*, 2789–2811.

Hong, S. (2019). A performance evaluation of bucket brigade order picking systems: Analytical and simulation approaches. *Computers & Industrial Engineering*, *135*, 120–131.

Hong, S., Johnson, A. L., & Peters, B. A. (2012a). Batch picking in narrow-aisle order picking systems with consideration for picker blocking. *European Journal of Operational Research*, *221*, 557–570.

Hong, S., Johnson, A. L., & Peters, B. A. (2012b). Large-scale order batching in parallel-aisle picking systems. *IIE Transactions*, *44*, 88–106.

Hong, S., & Kim, Y. (2017). A route-selecting order batching model with the S-shape routes in a parallel-aisle order picking system. *European Journal of Operational Research*, *257*, 185–196.

Hsieh, L.-F., & Huang, Y.-C. (2011). New batch construction heuristics to optimise the performance of order picking systems. *Intern. Journal of Production Economics*, *131*, 618–630.

Hsu, C.-M., Chen, K.-Y., & Chen, M.-C. (2005). Batching orders in warehouses by minimizing travel distance with genetic algorithms. *Computers in Industry*, *56*, 169–178.

Huang, M., Guo, Q., Liu, J., & Huang, X. (2018). Mixed model assembly line scheduling approach to order picking problem in online supermarkets. *Sustainability*, *10*, 3931.

Hwang, H., & Kim, D. G. (2005). Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. *International Journal of Production Research*, *43*, 3657–3670.

Il-Choe, K., & Sharp, G. P. (1991). *Small parts order picking: design and operation*. Technical Report School of Industrial and Systems Engineering. Georgia Institute of Technology Atlanta, EEUU.

Jarvis, J. M., & Mc. Dowell, E. D. (1991). Optimal product layout in an order picking warehouse. *IIE transactions*, *23*, 93–102.

Jiang, X., Sun, L., Zhang, Y., & Hu, X. (2022). Order batching and sequencing for minimising the total order completion time in pick-and-sort warehouses. *Expert Systems with Applications*, *187*, 115943.

Jiang, X., Zhou, Y., Zhang, Y., Sun, L., & Hu, X. (2018). Order batching and sequencing problem under the pick-and-sort strategy in online supermarkets. *Procedia computer science*, *126*, 1985–1993.

Koch, S., & Wäscher, G. (2016). A grouping genetic algorithm for the Order Batching Problem in distribution warehouses. *Journal of Business Economics*, *86*, 131–153.

Kübler, P., Glock, C. H., & Bauernhansl, T. (2020). A new iterative method for solving the joint dynamic storage location assignment, order batching and picker routing problem in manual picker-to-parts warehouses. *Computers & Industrial Engineering*, *147*, 106645.

Kuhn, H., Schubert, D., & Holzapfel, A. (2021). Integrated order batching and vehicle routing operations in grocery retail–a general adaptive large neighborhood search algorithm. *European Journal of Operational Research*, *294*, 1003–1021.

Kulak, O., Sahin, Y. F., & Taner, M. E. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, *24*, 52–80.

Le-Duc, T., & De Koster, R. B. M. (2007). Travel time estimation and order batching in a 2-block warehouse. *European Journal of Operational Research*, *176*, 374–388.

Lenoble, N., Frein, Y., & Hammami, R. (2017). Optimization of order batching in a picking system with carousels. *IFAC-PapersOnLine*, *50*, 1106–1113. 20th IFAC World Congress, 20th World Congress of the International Federation of Automatic Control.

Lenoble, N., Frein, Y., & Hammami, R. (2018). Order batching in an automated warehouse with several vertical lift modules: Optimization and experiments with real data. *European Journal of Operational Research*, *267*, 958–976.

Leung, K. H., Lee, C. K. M., & Choy, K. L. (2020). An integrated online pick-to-sort order batching approach for managing frequent arrivals of b2b e-commerce orders under both fixed and variable time-window batching. *Advanced Engineering Informatics*, *45*, 101–125.

Li, J., Huang, R., & Dai, J. B. (2016). Joint optimisation of order batching and picker routing in the online retailer's warehouse in China. *International Journal of Production Research*, *55*, 447–461.

Lin, C.-C., Kang, J.-R., Hou, C.-C., & Cheng, C.-Y. (2016). Joint order batching and picker Manhattan routing problem. *Computers and Industrial Engineering*, *95*, 164–174.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, *28*, 129–137.

Manjeshwar, P. K., Damodaran, P., & Srihari, K. (2009). Minimizing makespan in a flow shop with two batch-processing machines using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, *25*, 667–679.

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

Matusiak, M., De Koster, R. B. M., & Saarinen, J. (2017). Utilizing individual picker skills to improve order batching in a warehouse. *European Journal of Operational Research*, *263*, 888–899.

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

Menéndez, B., Pardo, E. G., Alonso-Ayuso, A., Molina, E., & Duarte, A. (2017b). 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. (2017c). Parallel variable neighborhood search for the min-max order batching problem. *International Transactions in Operational Research*, *24*, 635–662.

Miguel, F. M., Frutos, M., Méndez, M., & Tohmé, F. (2022). Order batching and order picking with 3d positioning of the articles: solution through a hybrid evolutionary algorithm. *Mathematical Biosciences and Engineering*, *19*, 5546–5563.

Miguel, F. M., Frutos, M., Tohmé, F., & Rossit, D. (2019). A memetic algorithm for the integral OBP/OPP problem in a logistics distribution center. *Uncertain Supply Chain Management*, *7*, 203–214.

Misni, F., & Lee, L. S. (2017). A review on strategic, tactical and operational decision planning in reverse logistics of green supply chain network design. *Journal of Computer and Communications*, *5*, 83–104.

Mohring, U., Baumann, M., & Furmans, K. (2020). Discrete-time analysis of levelled order release and staffing in order picking systems. *Logistics research*, *13*, 1.

Muter, I., & Öncan, T. (2015). An exact solution approach for the order batching problem. *IIE Transactions*, *47*, 728–738.

Nicolas, L., Yannick, F., & Ramzi, H. (2018). Order batching in an automated warehouse with several vertical lift modules: Optimization and experiments with real data. *European Journal of Operational Research*, *267*, 958–976.

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*, 142–155.

Oxenstierna, J., Malec, J., & Krueger, V. (2021). Layout-agnostic order-batching optimization. In *International Conference on Computational Logistics* (pp. 115–129). Springer.

Pan, J. C.-H., & Liu, S. Y. (1995). A comparative study of order batching algorithms. *Omega*, *23*, 691–700.

Pan, J. C.-H., Shih, P.-H., & Wu, M.-H. (2015). Order batching in a pick-and-pass warehousing system with group genetic algorithm. *Omega*, *57*, 238–248.

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

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

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

Petersen, C. G. (2000). An evaluation of order picking policies for mail order companies. *Production and operations management*, *9*, 319–335.

Pferschy, U., & Schauer, J. (2018). Order batching and routing in a non-standard warehouse. *Electronic Notes in Discrete Mathematics*, *69*, 125–132.

Pinto, A. R. F., & Nagano, M. S. (2019). An approach for the solution to order batching and sequencing in picking systems. *Production Engineering*, *13*, 325–341.

Pratap, S., Nayak, A., Cheikhrouhou, N., & Tiwari, M. K. (2015). Decision support system for discrete robust berth allocation. *IFAC-PapersOnLine*, *48*, 875–880.

Rasmi, S. A. B., Wang, Y., & Charkhgard, H. (2022). Wave order picking under the mixed-shelves storage strategy: A solution method and advantages. *Computers & Operations Research*, *137*, 105556.

Ratliff, H. D., & Rosenthal, A. S. (1983). Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. *Operations Research*, *31*, 507 – 521.

Rosenwein, M. B. (1996). A comparison of heuristics for the problem of batching orders for warehouse selection. *International Journal of Production Research*, *34*, 657–664.

Ruben, R. A., & Jacobs, F. R. (1999). Batch construction heuristics and storage assignment strategies for walk/ride and pick systems. *Management Science*, *45*, 575–596.

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*, 62 – 71.

Schiffer, M., Boysen, N., Klein, P. S., Laporte, G., & Pavone, M. (2022). Optimal picking policies in e-commerce warehouses. *Management Science*, .

Schleyer, M., & Gue, K. R. (2012). Throughput time distribution analysis for a one-block warehouse. *Transportation Research Part E: Logistics and Transportation Review*, *48*, 652–666.

Scholz, A., Henn, S., Stuhlmann, M., & Wäscher, G. (2016). A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, *253*, 68–84.

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*, 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*, 491–520.

Schrotenboer, A. H., Wruck, S., Vis, I. F. A., & Roodbergen, K. J. (2019). Integration of returns and decomposition of customer orders in e-commerce warehouses. *arXiv preprint CoRR*, *abs/1909.01794*.

Shavaki, F. H. N., & Jolai, F. (2021). A rule-based heuristic algorithm for joint order batching and delivery planning of online retailers with multiple order pickers. *Applied Intelligence*, *51*, 3917—3935.

Tang, L. C., & Chew, E. P. (1997). Order picking systems: Batching and storage assignment strategies. *Computers & Industrial Engineering*, *33*, 817 – 820.

Tian, X., Zhou, L., & Yang, J. (2019). Research on two-stage order picking sequencing for intensive shelf. In *MATEC Web of Conferences* (p. 02003). EDP Sciences volume 296.

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

Tsai, C. Y., Liou, J. H., & Huang, T. M. (2008). Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. *International Journal of Production Research*, *46*, 6533–6555.

Valle, C. A., & Beasley, J. E. (2020). Order batching using an approximation for the distance travelled by pickers. *European Journal of Operational Research*, *284*, 460–484.

Valle, C. A., Beasley, J. E., & Da Cunha, A. S. (2016). Modelling and Solving the Joint Order Batching and Picker Routing Problem in Inventories. *Lecture Notes in Computer Science. Combinatorial Optimization: 4th International Symposium, ISCO*, *9849*, 81–97.

Valle, C. A., Beasley, J. E., & Da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, *262*, 817–834.

Van Gils, T., Braekers, K., Ramaekers, K., Depaire, B., & Caris, A. (2016). Improving order picking efficiency by analyzing the combination of storage, batching, zoning and routing policies. *Lecture Notes in Computer Science. Computational Logistics. ICCL 2016*, *9855*, 427–442.

Van Gils, T., Caris, A., Ramaekers, K., & Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, *277*, 814–830.

Van Gils, T., Ramaekers, K., Braekers, K., Depaire, B., & Caris, A. (2018). Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. *International Journal of Production Economics*, *197*, 243–261.

Van Nieuwenhuyse, I., & De Koster, R. B. M. (2009). Evaluating order throughput time in 2-block warehouses with time window batching. *Inter. Journal of Production Economics*, *121*, 654–664.

Van Nieuwenhuyse, I., De Koster, R. B. M., & Colpaert, J. (2007). Order batching in multi-server pick-and-sort warehouses. *Katholieke Universiteit Leuven, Department of Decision Sciences and Information Management*, *180*, 367–8869.

Wagner, S., & Mönch, L. (2022). A variable neighborhood search approach to solve the order batching problem with heterogeneous pick devices. *European Journal of Operational Research*, *304*, 461–475.

Won, J., & Olafsson, S. (2005). Joint order batching and order picking in warehouse operations. *International Journal of Production Research*, *43*, 1427–1442.

Xu, X., Liu, T., Li, K., & Dong, W. (2014). Evaluating order throughput time with variable time window batching. *International Journal of Production Research*, *52*, 2232–2242.

Yang, J., Zhou, L., & Liu, H. (2021). Hybrid genetic algorithm-based optimisation of the batch order picking in a dense mobile rack warehouse. *Plos one*, *16*, e0249543.

Yang, N. (2022). Evaluation of the joint impact of the storage assignment and order batching in mobile-pod warehouse systems. *Mathematical Problems in Engineering*, *2022*.

Yang, P., Zhao, Z., & Guo, H. (2020). Order batch picking optimization under different storage scenarios for e-commerce warehouses. *Transportation Research Part E: Logistics and Transportation Review*, *136*, 101897.

Yousefi Nejad, M. A., Ebadi Torkayesh, A., Malmir, B., & Neyshabouri Jami, E. (2021). Robust possibilistic programming for joint order batching and picker routing problem in warehouse management. *International Journal of Production Research*, *59*, 4434–4452.

Yu, M. 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*, 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*, 271 – 284.

Zhang, J., Wang, X., & Huang, K. (2016). Integrated on-line scheduling of order batching and delivery under b2c e-commerce. *Computers & Industrial Engineering*, *94*, 280 – 289.

Zhang, J., Wang, X., & Huang, K. (2018). On-line scheduling of order picking and delivery with multiple zones and limited vehicle capacity. *Omega*, *79*, 104 – 115.

Zhang, J., Zhang, X., & Zhang, Y. (2021). A study on online scheduling problem of integrated order picking and delivery with multizone vehicle routing method for online-to-offline supermarket. *Mathematical Problems in Engineering*, *2021*.

Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., & Hu, K. (2012). Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Computers & operations research*, *39*, 1315–1324.

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* (p. 01015). EDP Sciences volume 272.

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*, 653–664.

Žulj, I., Salewski, H., Goeke, D., & Schneider, M. (2021). Order batching and batch sequencing in an AMR-assisted picker-to-parts system. *European Journal of Operational Research*, .

Zuniga, C. A., Olivares-Benitez, E., Tenahua, A. M., & Mujica, M. A. (2015). A methodology to solve the order batching problem. *IFAC-PapersOnLine*, *48*, 1380–1386.

Chapter 4

Online Order Batching Problem with a Single Picker

The Online Order Batching Problem with a Single Picker is a variant of the Order Batching Problem, where the arrival of products to the warehouse is dynamic and there is only one picker in the warehouse. As the result of the research performed in this Doctoral Thesis, two articles have been published for this variant of the problem, which are presented next. Then, for each publication, we compile the bibliographic details of the publication (complete reference, journal, ranking index, category, and ranking score) and, finally, a copy of the published article is attached.

The article titled "*New VNS Variants for the Online Order Batching Problem*" [89] compiles the first research carried out in this Doctoral Thesis and presents the preliminary work carried out to solve the Online Order Batching Problem with a Single Picker. The objective function studied in this paper was the minimization of the maximum time that an order remains in the system. In this case, the arrival of orders occurs in a period of 4 hours. The problem was solved for a single picker in a rectangular warehouse of a single block, with five parallel aisles and a total of 90 stored products. The set of instances used for the problem was a reduced set of 16 instances widely used in the literature [4]. As we have already seen in the previous sections, to solve this problem, several associated subtasks have to be solved. In this work, we introduced a new batching algorithm, based on the Basic Variable Neighborhood Search (BVNS) methodology [197]. The proposal was compared to three well-known algorithms widely used in the literature for the batching task: First-Come-First-Serve (FCFS), Seed [212, 239], and Saving Clarke & Wright [42]. In all cases, the routing was addressed with the S-Shape heuristic algorithm. The results obtained were very promising since, for the same execution time, we obtained an improvement of more than 30% over the compared methods. This article is attached in Section 4.1.

The article "*GRASP with Variable Neighborhood Descent for the Online Order Batching Problem*" [92] was the second work carried out in this Doctoral Thesis and can be considered as an evolution of the previous work, to solve the Online Order Batching Problem with a single picker. In this work, we studied the minimization of the total completion time of all orders, but also reported the maximum turnover time. The arrival of orders occurs in a period of 4 hours, and the problem was solved for a rectangular warehouse of a single block and a single picker. We used two different sets of instances widely used in the literature [4, 125]. These sets of instances include five different warehouse configurations with a variant number of parallel aisles. In addition, we presented a mathematical model to define the problem addressed in its offline version. Then, we evaluated the maximum execution time of the model for different number of orders for some instances, in order to analyze the intractability of the problem. Then, we tackled the batching task with a method that combined the Greedy Randomized Adaptive

Search Procedure (GRASP) [76] to construct initial solutions, with a variable Neighborhood Descent (VND) [197] approach to improve the solutions. The implementation of GRASP used a greedy heuristic based on the complete capacity of the batch. The VND included the use of three different neighborhoods. The routing task was handled with an S-shape algorithm. The approach proposed was compared to the best previous approach in the state of the art, the Iterated Local Search (ILS) introduced in [125]. The combination of GRASP+VND resulted in an improvement of more than 2.5% for minimizing the completion time and an improvement of more than 6.5% for minimizing the maximum turnover time, with respect to the previous algorithm in the state of the art. This article is attached in Section 4.2.

4.1 New VNS Variants for the Online Order Batching Problem

Gil-Borrás S., Pardo E.G., Alonso-Ayuso A., Duarte A. (2019) New VNS Variants for the Online Order Batching Problem. In: Sifaleras A., Salhi S., Brimberg J. (eds) Variable Neighborhood Search. ICVNS 2018. Lecture Notes in Computer Science, vol 11328. (p. 89-100) Springer, Cham.

Published in "**Lecture Notes in Computer Science**"

ISSN: 0302-9743 / 1611-3349

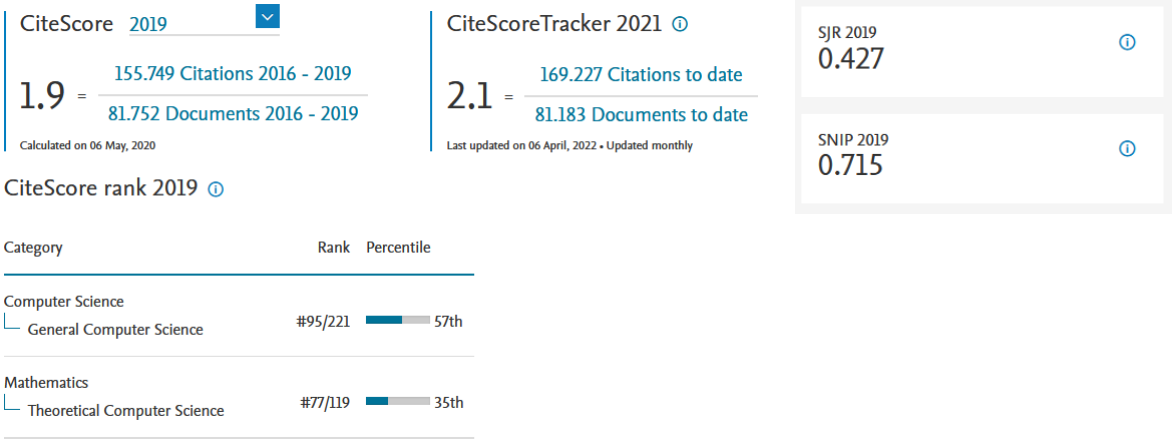
https://doi.org/10.1007/978-3-030-15843-9_8

CiteScore 2019: 1.9

SJR 2019: 0.427

CiteScore rank 2019

- Computer Science, General Computer Science. Ranking 95/221 (Q2).
- Mathematics, Theoretical Computer Science. Ranking 77/119 (Q3).



New VNS Variants for the Online Order Batching Problem

Sergio Gil-Borrás[1], Eduardo G. Pardo[1(✉)], Antonio Alonso-Ayuso[2], and Abraham Duarte[2]

[1] Dept. Sistemas Informáticos, Universidad Politécnica de Madrid, Madrid, Spain
sergio.gil.borras@alumnos.upm.es, eduardo.pardo@upm.es
[2] Department of Computer Science, Universidad Rey Juan Carlos, Móstoles, Spain
{antonio.alonso,abraham.duarte}@urjc.es

Abstract. The Order Batching Problem (OBP) can be considered a family of optimization problems related to the retrieval of goods in a warehouse. The original and most extended version of the problem consists in minimizing the total time needed to collect a group of orders. However, this version has been evolved with many other variants, where the restrictions and/or the objective function might change. In this paper, we deal with the Online Order Batching Problem (OOBP) version, which introduces the novelty to the OBP of considering orders that have arrived to the warehouse once the retrieval of previous orders has started. This family of problems has been deeply studied by the heuristic community in the past. Notice, that solving any variant of the OBP include two important activities: grouping the orders into batches (batching) and determining the route to follow by a picker to retrieve the items within the same batch (routing). We review the most outstanding proposals in the literature for the OOBP variant and we propose a new version of a competitive Variable Neighborhood Search (VNS) algorithm to tackle the problem.

Keywords: Online Order Batching Problem · Batching · Variable Neighborhood Search

1 Introduction

The storage of goods in warehouses has associated many tasks such as receiving the goods, storing them or retrieving the products from the shelves of the warehouse, when a new order arrives. Many of those tasks can be enunciated as optimization problems with the aim of saving time, space or work load, among others. The Order Batching Problem (OBP) can be considered a family of optimization problems, more than a single problem, related to the operation of retrieval of goods in a warehouse, when the policy of retrieval is based on order batching.

This work has been partially founded by Ministerio de Economía y Competitividad with grant refs. TIN2015-65460-C2-2-P., MTM2015-63710-P.

A. Sifaleras et al. (Eds.): ICVNS 2018, LNCS 11328, pp. 89–100, 2019.
https://doi.org/10.1007/978-3-030-15843-9_8

The order batching, then, consist in grouping a set of orders together (conforming a batch) and assigning the batch to a person (the picker) who retrieves all the orders within the same batch on a single tour through the warehouse. This policy has been proved to be very effective in contrast with the traditional strict-order picking policy, where each order that arrives to the warehouse is assigned to a picker, who collects exclusively the items from that order on each tour. Some authors point out that it is possible to reduce the travel time up to 35% if the routes followed by the pickers are designed adequately [4]. Additionally, if the batching and routing are considered simultaneously, the save of time can be even larger.

The original and most extended version of the problem, usually known as Order Batching Problem (OBP) consists in minimizing the total time needed to collect a group of orders. However, this version has been evolved with many other variants, where the restrictions and/or the objective function might change. Notice, that solving any variant of the OBP might include two important activities: grouping the orders into batches (batching), and finding the route to follow by the picker to collect the items within the same batch (routing). Additionally, some variants of the OBP also consider a third activity: determining the next batch to be processed (sequencing) once the batches have already been conformed.

In this paper, we deal with the Online Order Batching Problem (OOBP), which is a version of the OBP that introduces the novelty of considering orders that have arrived to the warehouse once the process of retrieval of previous orders has already started. The objective function of the problem is to minimize the maximum time that an order remains in the system. This is usually known in the related literature as the turnover time. To tackle this problem we propose the use of the methodology Variable Neighborhood Search (VNS), particularly, the Basic Variable Neighborhood Search (BVNS) variant and we compare our approach with the classical approaches in the literature for other variants of the OBP.

The rest of the paper is organized as follows: in Sect. 2 we review the most outstanding proposals for the problem in the literature, and we describe in detail the methods that will be used in our experiments as a comparative framework. In Sect. 3 we propose a new version of a competitive Variable Neighborhood Search algorithm to tackle the problem. In Sect. 4 we perform the experiments in order to compare our proposal with the traditional methods for the OBP family of problems. Finally, in Sect. 5 we present our conclusions future research lines.

2 State of the Art

The Order Batching family of problems has been deeply studied by the heuristic community in the past. There are remarkable references based on different metaheuristics for most of the best-known variants of problems within the OBP literature: the classical OBP [1,11,15,16]; the Min-max OBP [7,13]; the OBSP [2,14]; and also to the OOBP tackled in this paper [10,20,22].

However, the first remarkable methods for most of the previous problems are not the metaheuristic approaches but the simpler heuristic procedures based on greedy functions [12]. Those methods were constructive procedures based on simple ideas and have been used as a baseline in many comparisons.

As far as we know, those methods have not been either used or compared in the context of the OOBP. Next, we present a brief description of the most remarkable ones that will be used later in the Sect. 4. Particularly, we consider: the First Come First Served algorithm (Sect. 2.1); the Seed algorithm (Sect. 2.2); and, the Clarke & Wright Savings algorithm (Sect. 2.3).

2.1 First Come First Served Algorithm

The First Come First Served (FCFS) algorithm is probably the simplest heuristic algorithm designed for the OBP. The algorithm receives a list of orders and returns a list of batches. First, the received list of orders is sorted according to the arrival time of each order, in such a way that the oldest order comes first. Then the list of orders is traversed one by one, assigning the next order to be processed to the next available batch. If the order fits in the current batch it is inserted in that batch. Otherwise, a new batch is created with that order, becoming this new batch the current one which will be target of the next considered order. This process is repeated with all the orders until the end of the list. Once all the orders have a batch assigned, the set of batches generated in the algorithm is returned.

2.2 Seed Algorithm

The algorithms known as "seed algorithms" are a group of methods based on a common strategy: a "seed" (in this case an order) is first chosen and assigned to a batch. Then, other available orders might be added to the same batch, as far as the capacity constraint is not violated. Therefore, for each "seed method" it will be necessary to determine how to choose the seed order, and how to choose the additional orders suitable to be assigned to a particular batch with an assigned seed. In this case, the strategy used to select a "seed order" considers the idea introduced in [18] consisting in selecting the available order with the largest number of products. Then, once the seed has been chosen, the strategy used to aggregate orders to the same batch is the one introduced in [21] consisting in selecting the order with lowest absolute difference of its Center Of Gravity (COG) to the seed. Where the COG of an order is defined as the average of the aisle numbers where the items of that order are located. The considered procedure applies a "cumulative mode" (i.e., the seed is renewed each time a new batch is created). The method, then, consist in selecting one seed order, assign it to a batch and trying to complete the batch following the criteria of the difference of COG. Once the batch is full, the method selects a new seed and so on until all the orders have been assigned to a batch.

2.3 Clark & Wright Savings Algorithm

The "Clark & Wright savings" algorithm is inspired in the idea presented in [3] in the context of vehicles routing. It is based on computing the save of time derived from collecting two orders separately versus collecting them together in the same route. The algorithm creates a square matrix with a size equals to the number of orders. Then, each row/column corresponds with one order. The crossing position of a column and a row will store the save/loss of time of collecting the two orders related, separately or together. Additionally, the diagonal of the matrix would store the time of collecting each order in isolation. Notice that this is a symmetric matrix, therefore only one half of the matrix (above or under the diagonal) is needed. For instance, the saving of collecting orders 1 and 2 would be computed as follows: $saving = t_1 + t_2 - t_{1,2}$ where t_1 and t_2 represent the time needed to collect orders 1 and 2 separately, and $t_{1,2}$ the time needed to collect them together. Then all the pairs of orders are stored in a list sorted depending on their savings, in a decreasing way. Next, the list is scanned trying first to allocate together the pairs which produce a largest saving. Notice that several situations might happen: if both orders have not been previously allocated in a batch and they fit together, they are assigned to the same batch; if one of the batches have already been allocated, then the other one will be assigned to the same batch if it fits. Otherwise the procedure will continue with the next pair; finally, if both orders involved have previously been placed in other batches the procedure will jump again to the following pair. We refer the reader to [4] for further details.

3 Algorithmic Proposal

In this section we present our algorithmic proposal to tackle the OOBP. In particular, we propose the use of the methodology Variable Neighborhood Search (VNS) [17]. VNS was originally proposed by Mladenović and Hansen in 1997 as a revolutionary idea to escape from a local optimum, based on the concept of change of the neighborhood structure. Then, the general idea behind the method is to reach local optimum by using a local search procedure and then, change the neighborhood structure (once the current solution found can not be further improved) in order to give the local search the opportunity of looking for a new local optimum in the new neighborhood.

There original idea has been notably evolved with many variants. Probably, the most remarkable ones are: Reduced VNS (RVNS) which perform a stochastic search within a neighborhood; Variable Neighborhood Descent (VND) which perform a deterministic search within the considered neighborhoods; Basic VNS (BVNS) which combines stochastic and deterministic exploration in one neighborhood; and General VNS (GVNS) which combines stochastic and deterministic exploration within a set of neighborhoods. Other well-known approaches are: Skewed VNS (SVNS); and Variable Neighborhood Decomposition Search (VNDS). For a detailed description and tutorials of all those methods we refer the reader to [8,9,17]. Other recent variants include: Variable Formulation Search

(VFS) [19], Parallel Variable Neighborhood Search [6,13] and Multi-Objective Variable Neighborhood Search [5].

In this paper we make use of the BVNS algorithm. In Algorithm 1 we present a pseudocode of this method. It receives three parameters to start the search: (i) an initial solution S generated with an external method; (ii) a value k_{max} which determines the maximum number of neighborhoods to explore; and (iii) the maximum allowed running time (t_{max}). The method explores the neighborhood of the current solution trying to obtain a better one. In order to do that, BVNS has three stages that run consecutively. The first stage is the perturbation of the current solution, performed in order to escape from the current local optimum, reaching a solution in a new neighborhood. As a second stage the method make use of a local search procedure, which is able to find a local optimum within the current neighborhood. The third stage, represented by the procedure `Neighborhoodchange`, determines if it is necessary to change the neighborhood to be explored, depending on whether the solution provided to the local search has been improved or not. This method updates the value of the variable k, which indicates the number of perturbations to be performed to the current solution in the `Shake` procedure. The value $k = 1$ indicates that an improvement has been performed, otherwise the value of k is incremented in a predefined amount (typically 1 unit).

Algorithm 1. `BVNS`$(S, k_{\max}, t_{\max})$

```
1: repeat
2:     k ← 1
3:     while k ≤ k_max do
4:         S' ← Shake(S, k)
5:         S'' ← LocalSearch(S')
6:         k ← NeighborhoodChange(S, S'', k)
7:     end while
8: until t < t_max
9: return S
```

A more detailed description of the method used to generate the initial solution can be found in the Sect. 3.1. Similarly, the description of the `Shake` and `LocalSearch` procedures are presented, respectively, in Sect. 3.2 and Sect. 3.3. Notice that we do not provide a detailed description of the `NeighborhoodChange` procedure since it follows an standard implementation.

The algorithm is executed repeatedly until the maximum allowed time is reached. In each iteration, the number of perturbations performed to the solution, before the local search, is indicated by the value of the variable k. The variable k starts at 1, indicating that the first neighborhood to be explored is the closer one. This value is increased every time that the local search does not improve the current solution, until it reaches the value of k_{max}. Then, the variable k is reset to its initial value 1 and the procedure is repeated again until the maximum allowed time is reached.

3.1 Constructive Procedure

We have used a random algorithm as a constructive method in order to provide an initial solution to the BVNS algorithm. The algorithm receives a list of orders as an input parameter. The list of orders is randomly scanned. In each iteration, an order is randomly selected and it is placed in the next available batch. When the selected order no longer fits in the current batch, a new batch is created with this order. Then, the next order will be placed in this new batch and the process is repeated until the order list is fully scanned and all the orders have a batch assigned. Once the process is finished, the procedure returns a list of batches as a solution. In Algorithm 2 we present a pseudocode of this procedure.

Algorithm 2. `Constructive`(L_{orders})

```
1: S ← NewBatchList()
2: B ← NewBatch()
3: repeat
4:     o ← ChooseRandomOrder(L_orders)
5:     L_orders ← L_orders \ o
6:     if Fits(B, o) then
7:         Add(B, o)
8:     else
9:         Add(S, B)
10:        B ← NewBatch()
11:        Add(B, o)
12:    end if
13: until L_orders = ∅
14: return S
```

3.2 Shake Procedure

The perturbation procedure chosen for this problem consist in exchanging two orders from different batches. The procedure receives as input parameters an initial solution S and the parameter k that indicates the number of times the perturbation will occur. In each perturbation two random batches are selected. Then, two orders also selected at random within the selected batches are exchanged. Notice that the exchange must produce a feasible solution (i.e., it does not exceed the maximum capacity of each batch), otherwise it should be repeated. This process will be repeated as many times as the parameter k indicates. At the end of this procedure, a solution in a different neighborhood will be returned. In Algorithm 3 we present a pseudocode of this procedure.

3.3 Local Search Procedure

The local search procedure proposed to be used within the BVNS, as well as the shake procedure, is based in the one-to-one exchange move. This procedure receives an initial solution S, as an input parameter, and it returns the

Algorithm 3. `Shake`(S, k)

```
repeat
    repeat
        B_i ← ChooseRandomBatch(S)
        B_j ← ChooseRandomBatch(S)
    until B_i ≠ B_j
    o_i ← ChooseRandomOrder(B_i)
    o_j ← ChooseRandomOrder(B_j)
    if Fits(B_i \ o_i, o_j) and Fits(B_j \ o_j, o_i) then
        B_i ← B_i \ o_i
        Add(B_i, o_j)
        B_j ← B_j \ o_j
        Add(B_j, o_i)
        k ← k − 1
    end if
until k = 0
return S
```

local optimum within the neighborhood of the solution. The procedure explores every order o in all the batches trying to find a feasible interchange with other order that improves the current solution. If an improve move is performed, then the procedure starts again from the new solution found, performing another whole iteration, otherwise it carries on until all candidate interchanges have been explored without improvement and returns the best solution found. In Algorithm 4 we present the pseudocode of this procedure.

Algorithm 4. `LocalSearch`(S)

```
repeat
    improved ← false
    for all o_i ∈ S do
        for all o_j ∈ S do
            S' ← Exchange(S, o_i, o_j)
            if f(S') < f(S) then
                S ← S'
                improved ← true
                break
            end if
        end for
    end for
until improved = false
return S
```

4 Results

We compare our proposal with the classical greedy constructive procedures presented in Sect. 2. The experiments were run 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, and all the codes were developed in Java 8.

4.1 Instances

An instance to test any algorithm for the OOBP needs to consider the following aspects: the warehouse layout; the orders; and the distribution followed by the arrival of the orders.

We have selected and adapted a set of instances previously referred in the literature for the OBP to test our proposal. In particular, we have selected a subgroup of instances from those reported in [1] which have been reference instances in the OBP literature in the last few years. This data set contains instances related to four real warehouses of rectangular shape. Each warehouse has two transversal aisles, one at the front and one at the back of the warehouse and a variable number of parallel aisles. In each side of the parallel aisles there are products stored. Every warehouse has only one depot located at the front-cross aisle either at the left corner or at the center of the aisle. In the Fig. 1 we present an example of the layout of the considered warehouse. Particularly, this example warehouse has 2 crossing aisles and 5 parallel aisles, with 9 picking positions in each side of the parallel aisles, totalizing 90 picking positions. In this case, the depot is placed in the center of the front cross aisle.

The number of orders per instance varies among the following values [50, 100, 150, 200, 250]. The distribution of the products in the warehouse follows either an ABC distribution or a random one. We have selected 16 representative instances from the Warehouse 1 for our comparison. In this subset, we have selected 4 different instances for each number of orders [100, 150, 200, 250]. Notice that we have avoided the use of the smallest type of instances (i.e., the ones composed by 50 orders) since a small number of orders do not create enough congestion in the delivery of orders and, therefore, the instances become trivial for the OOBP.

Finally, we have adapted the instances by determining distribution of the delivery instant of the orders to the warehouse. We have divided each set of orders into two groups: offline/online. The first group is formed by 15 orders which will be already available at the beginning of execution. The rest of the orders will arrive to the warehouse following an uniform distribution along the time horizon of 4 h.

4.2 Comparison with the State of the Art

The BVNS algorithm has been successfully compared with three different algorithms in the state of the art. Particularly, we have selected three classical and well-known greedy constructive procedures, widely used in the OBP literature:

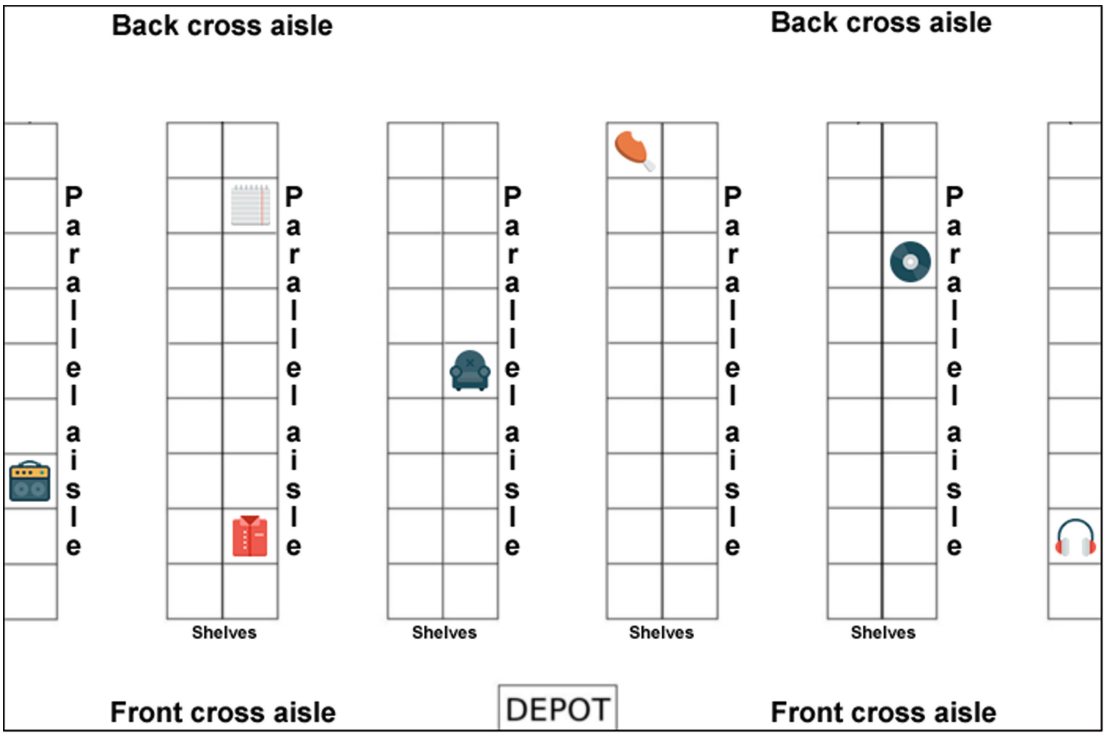


Fig. 1. Warehouse layout.

the First Come First Served (FCFS) algorithm, a variant of the Seed algorithm, and finally the Savings C&W algorithm. These three algorithms were described in the Sect. 2.

Before to perform the comparison of the BVNS with the algorithms in the state of the art, we have carried on some preliminary experiments, to empirically adjust the value of the parameter k_{max} of the BVNS. In this case, we have selected $k_{max} = 15$ for the final configuration of the BVNS. Also, the value of t_{max} was set to 10 s. Therefore, every 10 s, the algorithm starts again from a new solution constructed with the procedure described in Sect. 3.1. Notice, that every construction considers all the orders already arrived to the warehouse and not collected yet.

In Table 1 we present the average value of the objective function (O.F.), which in this case is, for each instance, the maximum time that an order remains in the system before being served; the average deviation with respect to the best solution found in the experiment (Dev.(%)); the number of best solutions found in the experiment (#Best); and the running time of the CPU in seconds (CPUt(s)). Notice, that for each instance, the minimum running time is four hours. These four hours is the time that the order dispenser will use to deliver all the orders in the instance to the system. The final execution time will depend on the time that each algorithm takes to distribute those orders into batches and on the quality of the solution.

As it is shown in Table 1 BVNS is the best algorithm of the comparison since it was able to find the largest number of best solutions found (15 out of

Table 1. Average results of the comparison with the state-of-the-art algorithms.

	O.F.	Dev.(%)	#Best	CPUt(s)
BVNS	3682	0.37%	15	17508
FCFS	5897	55.36%	0	19717
Savings C&W	10621	187.70%	0	24353
Seed	5081	38.45%	1	18232

16 instances) and the smallest deviation of the compared algorithms, in shorter running times. In Table 2 we present the detailed results per instance.

Table 2. Results per instance of the compared algorithms.

	BVNS			FCFS			Savings C&W			Seed		
	O.F. (s)	Dev (%)	CPU t(s)	O.F. (s)	Dev (%)	CPU t(s)	O.F. (s)	Dev (%)	CPU t(s)	O.F. (s)	Dev (%)	CPU t(s)
100_000	**1491**	0.00%	16196	1856	24.50%	16149	2399	60.94%	16522	1974	32.44%	15989
100_030	**1522**	0.00%	15430	1924	26.46%	15416	2276	49.54%	15416	1847	21.39%	15416
100_060	**1771**	0.00%	16103	1927	8.78%	15857	2841	60.39%	16055	2218	25.23%	15996
100_090	**1371**	0.00%	15498	1492	8.84%	15478	1945	41.87%	15597	2008	46.41%	15484
150_000	**2602**	0.00%	15551	4461	71.41%	17565	10843	316.67%	24491	3802	46.11%	16163
150_030	**1181**	0.00%	14619	1492	26.36%	14512	2263	91.60%	14916	1311	11.01%	14584
150_060	**3078**	0.00%	16209	5413	75.89%	18829	11488	273.28%	24984	4074	32.38%	16524
150_090	**1068**	0.00%	14366	1522	42.47%	14323	1432	34.10%	14285	1150	7.67%	14416
200_000	**7135**	0.00%	21409	10871	52.36%	25490	20998	194.30%	35617	9679	35.65%	23344
200_030	1255	5.90%	15480	1497	26.33%	15981	4624	290.19%	19136	**1185**	0.00%	15334
200_060	**5400**	0.00%	19888	8926	65.29%	23584	19491	260.92%	34003	7975	47.68%	21259
200_090	**1498**	0.00%	15445	2709	80.85%	17148	6829	355.82%	21218	2350	56.85%	15305
250_000	**12202**	0.00%	25727	19750	61.86%	33437	28913	136.96%	42601	15657	28.32%	28479
250_030	**2446**	0.00%	15821	5629	130.14%	19315	11356	364.31%	25067	4710	92.57%	17018
250_060	**12028**	0.00%	25840	18201	51.33%	32017	30147	150.64%	43962	15551	29.29%	28299
250_090	**2869**	0.00%	16542	6683	132.97%	20375	12096	321.65%	25784	5801	102.22%	18107

5 Conclusions

In this paper we deal with the Online Order Batching Problem, as a variant of the well-known family of problems related to the Order Batching. This variant considers that there are orders which arrive to the warehouse once the retrieving process has already started. Those orders are immediately processed and introduced in a batch in order to be collected. The problem looks for minimizing the maximum time that an order remains in the system before being served.

To tackle this problem we have proposed several heuristics within the Basic Variable Neighborhood Search framework. Particularly, we propose to start the

search with a random solution and then we define a neighborhood, based on interchange moves, explored by a local search procedure which follows a first improvement strategy. The proposed method has been compared successfully with classical greedy methods in the state of the art, previously used for other variants of the OBP.

In a future research we propose the extension of our algorithm by defining new neighborhoods to be combined in a Variable Neighborhood Descent or in a General Variable Neighborhood Search procedure. Additionally, we also propose to extend the comparison performed, by considering not only the classical greedy constructive methods in the literature, but also the latest metaheuristic-based methods.

References

1. Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., De Blas, C.S.: Variable neighborhood search for order batching in a warehouse. Asia-Pac. J. Oper. Res. **26**(5), 655–683 (2009)
2. Azadnia, A.H., Taheri, S., Ghadimi, P., Mat Saman, M.Z., Wong, K.Y.: Order batching in warehouses by minimizing total tardiness: a hybrid approach of weighted association rule mining and genetic algorithms. Sci. World J. **2013**, (2013)
3. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. **12**(4), 568–581 (1964)
4. De Koster, R., Roodbergen, K.J., Van Voorden, R.: Reduction of walking time in the distribution center of de bijenkorf. In: Speranza, M.G., Stähly, P. (eds.) New Trends in Distribution Logistics. Lecture Notes in economics and mathematical systems, vol. 480, pp. 215–234. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-642-58568-5_11
5. Duarte, A., Pantrigo, J.J., Pardo, E.G., Mladenovic, N.: Multi-objective variable neighborhood search: an application to combinatorial optimization problems. J. Glob. Optim. **63**(3), 515–536 (2015)
6. Duarte, A., Pantrigo, J.J., Pardo, E.G., Sánchez-Oro, J.: Parallel variable neighbourhood search strategies for the cutwidth minimization problem. IMA J. Manag. Math. **27**(1), 55–73 (2013)
7. Gademann, N., Velde, V.D.S.: Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Trans. **37**(1), 63–75 (2005)
8. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
9. Hansen, P., Mladenović, N., Moreno-Pérez, J.A.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010)
10. Henn, S.: Algorithms for on-line order batching in an order picking warehouse. Comput. Oper. Res. **39**(11), 2549–2563 (2012)
11. Henn, S., Koch, S., Doerner, K.F., Strauss, C., Wäscher, G.: Metaheuristics for the order batching problem in manual order picking systems. Bus. Res. **3**(1), 82–105 (2010)
12. Koster, M.B.M.D., der Poort, E.S.V., Wolters, M.: Efficient orderbatching methods in warehouses. Int. J. Prod. Res. **37**(7), 1479–1504 (1999)
13. Menéndez, B., Pardo, E.G., Sánchez-Oro, J., Duarte, A.: Parallel variable neighborhood search for the min-max order batching problem. Int. Trans. Oper. Res. **24**(3), 635–662 (2017)

14. Menéndez, B., Bustillo, M., Pardo, E.G., Duarte, A.: General variable neighborhood search for the order batching and sequencing problem. Eur. J. Oper. Res. **263**(1), 82–93 (2017)
15. Menéndez, B., Pardo, E.G., Alonso-Ayuso, A., Molina, E., Duarte, A.: Variable neighborhood search strategies for the order batching problem. Comput. Oper. Res. **78**, 500–512 (2017)
16. Menéndez, B., Pardo, E.G., Duarte, A., Alonso-Ayuso, A., Molina, E.: General variable neighborhood search applied to the picking process in a warehouse. Electron. Notes Discret Math. **47**, 77–84 (2015)
17. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
18. Pan, C.H., Liu, S.Y.: A comparative study of order batching algorithms. Omega **23**(6), 691–700 (1995)
19. Pardo, E.G., Mladenović, N., Pantrigo, J.J., Duarte, A.: Variable formulation search for the cutwidth minimization problem. Appl. Soft Comput. **13**(5), 2242–2252 (2013)
20. Pérez-Rodríguez, R., Hernández-Aguirre, A., Jöns, S.: A continuous estimation of distribution algorithm for the online order-batching problem. Int. J. Adv. Manuf. Technol. **79**(1), 569–588 (2015)
21. Rosenwein, M.B.: An application of cluster analysis to the problem of locating items within a warehouse. IIE Trans. **26**(1), 101–103 (1994)
22. Rubrico, J., Higashi, T., Tamura, H., Ota, J.: Online rescheduling of multiple picking agents for warehouse management. Robot. Comput.-Integr. Manuf. **27**(1), 62–71 (2011)

4.2 GRASP with Variable Neighborhood Descent for the online order batching problem

Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte A. (2020) GRASP with Variable Neighborhood Descent for the online order batching problem. Journal of global optimization. 78, (p. 295–325).

Published in "**Journal of global optimization**"

ISSN: 0925-5001 / 1573-2916

https://doi.org/10.1007/s10898-020-00910-2

Impact factor (JCR 2020): 2.207

Journal Citation Indicator (JCI): 0.82

Rank by Journal Impact Factor

- Mathematics, Applied.
  Ranking 60/265 (Q1).
- Operations research & management science.
  Ranking 49/84 (Q3).

Rank by Journal Citation Indicator (JCI)

- Mathematics, Applied.
  Ranking 130/309 (Q2).
- Operations research & management science.
  Ranking 33/99 (Q2).

Rank by Journal Impact Factor

Journals within a category are sorted in descending order by Journal Impact Factor (JIF) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

EDITION: Science Citation Index Expanded (SCIE)
CATEGORY: MATHEMATICS, APPLIED
60/265

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	60/265	Q1	77.55
2019	56/261	Q1	78.74
2018	62/254	Q1	75.79
2017	69/252	Q2	72.82
2016	36/255	Q1	86.08

EDITION: Science Citation Index Expanded (SCIE)
CATEGORY: OPERATIONS RESEARCH & MANAGEMENT SCIENCE
49/84

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	49/84	Q3	42.26
2019	42/83	Q3	50.00
2018	43/84	Q3	49.40
2017	45/84	Q3	47.02
2016	31/83	Q2	63.25

Rank by Journal Citation Indicator (JCI)

Journals within a category are sorted in descending order by Journal Citation Indicator (JCI) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

CATEGORY: MATHEMATICS, APPLIED
130/309

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	130/309	Q2	58.09
2019	122/309	Q2	60.68
2018	120/308	Q2	61.20
2017	86/304	Q2	71.88

CATEGORY: OPERATIONS RESEARCH & MANAGEMENT SCIENCE
33/99

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	33/99	Q2	67.17
2019	30/99	Q2	70.20
2018	29/98	Q2	70.92
2017	22/96	Q1	77.60

Journal of Global Optimization (2020) 78:295–325
https://doi.org/10.1007/s10898-020-00910-2

GRASP with Variable Neighborhood Descent for the online order batching problem

Sergio Gil-Borrás[1] · Eduardo G. Pardo[2] · Antonio Alonso-Ayuso[2] · Abraham Duarte[2]

Received: 24 May 2019 / Accepted: 16 April 2020 / Published online: 11 May 2020

Abstract
The Online Order Batching Problem (OOBP) is a variant of the well-known Order Batching Problem (OBP). As in the OBP, the goal of this problem is to collect all the orders that arrive at a warehouse, following an order batching picking policy, while minimizing a particular objective function. Therefore, orders are grouped in batches, of a maximum predefined capacity, before being collected. Each batch is assigned to a single picker, who collects all the orders within the batch in a single route. Unlike the OBP, this variant presents the peculiarity that the orders considered in each instance are not fully available in the warehouse at the beginning of the day, but they can arrive at the system once the picking process has already begun. Then, batches have to be dynamically updated and, as a consequence, routes must too. In this paper, the maximum turnover time (maximum time that an order remains in the warehouse) and the maximum completion time (total collecting time of all orders received in the warehouse) are minimized. To that aim, we propose an algorithm based in the combination of a Greedy Randomized Adaptive Search Procedure and a Variable Neighborhood Descent. The best variant of our method has been tested over a large set of instances and it has been favorably compared with the best previous approach in the state of the art.

Keywords Warehouse management · Online order batching problem · Order batching · Turnover time · Heuristics

1 Introduction

The picking of items in a warehouse, as a part of the supply chain management, follows a picking policy which determines how the picking of items is performed in the warehouse. It is possible to find different picking policies in the literature such as: single-order picking,

This research was partially funded by the projects: MTM2015-63710-P, RTI2018-094269-B-I00, TIN2015-65460-C2-2-P and PGC2018-095322-B-C22 from Ministerio de Ciencia, Innovación y Universidades (Spain); by Comunidad de Madrid and European Regional Development Fund, Grant Ref. P2018/TCS-4566; and by Programa Propio de I+D+i de la Universidad Politécnica de Madrid (Programa 466A).

Eduardo G. Pardo
eduardo.pardo@urjc.es

Extended author information available on the last page of the article

batching and sort-after-picking, single-order picking with zoning, batching with zoning, among others. Order Batching can be considered as a family of picking policies which are based on grouping the orders received in the warehouse into batches, prior to start the picking process. Once the batches have been conformed, all the items within the orders of the same batch are picked together in the same picking route. There are many optimization problems related to the process of picking items in a warehouse when the picking policy is order batching.

Within this family of problems, the most classical version is usually referred to as Order Batching Problem (OBP) [44], which consists of minimizing the total time needed to collect a group of orders received in a warehouse. This version has raised a relevant interest in the scientific community. The OBP has been proved to be $\mathcal{NP}$-hard for general instances [12]. Nonetheless, it is solvable in polynomial time if each batch does not contain more than two orders [12]. Unfortunately, real warehouse instances does not usually fall into this category. Consequently, it has been heuristically approached in the last years by using both, heuristics and metaheuristics. The First-Come First-Served (FCFS) strategy might be the first heuristic approach implemented in warehouses to assign orders to batches. This strategy has been widely used due to its simplicity. Other important heuristic methods are *seed methods* [13,22,32] and *saving methods* [40]. In [4] it is possible to find a survey of those methods where the authors proposed a classification.

More recently, it is also possible to find metaheuristic-based approaches in the literature. The first metaheuristic algorithm applied to the problem was described in [23], where a Genetic Algorithm is introduced. Later, in [1], an algorithm based on the Variable Neighborhood Search methodology is proposed; specifically, the authors considered several neighborhoods in a Variable Neighborhood Descent (VND) scheme. Henn et al. [19] proposed an Iterated Local Search and a Rank-Based Ant System algorithms and in [21] Henn improved previous results by introducing two additional algorithms: Tabu Search and Attribute-Based Hill Climber. In [30], Oncan proposed an Iterated Local Search algorithm with a Tabu Thresholding method as the local search procedure. As far as we know, the latest proposal and the current state of the art for the problem, was made in [26] where the authors proposed a multi-start algorithm based on the Variable Neighborhood Search (VNS) methodology. In each iteration of this algorithm it starts by generating a different initial solution, which is improved using a Basic VNS. Then a post-optimization strategy based on General VNS is applied.

The OBP has been extended with the inclusion of different constraints or alternative objective functions. The two more outstanding ones are: the Order Batching and Sequencing Problem (OBSP) which introduces a constraint related to the due date of each order (see for instance [20,25]). Also, the Min–Max Order Batching Problem (Min–Max OBP) consists of looking for a balance in the workload of a group of pickers, when considering multiple pickers to collect the batches. We refer the reader to [11,28] to review the state of the art of this problem.

All the previous approaches, within the Order Batching family, can be considered as static variants. However, the development of online e-commerce and the reduction of delivery times guaranteed by sellers make this *static* approach of the OBP very restrictive and unrealistic. In real contexts, orders are entering continuously in the system and it is necessary to modify the initial batch allocation, to be able to attend the new orders within the window of time committed to the client. This dynamic version of the problem is known as the Online Order Batching Problem (OOBP). Therefore, the list of orders arrived to the warehouse is updated online and the algorithms have to create batches and routes without having the complete information of all the orders that need to be collected in a working day, nor the arrival time

of future orders not yet in the warehouse. Note that a fundamental difference of this problem compared to its static version is that the method of creating batches and routes is working the whole day. In the static version, the system provides an initial solution that is valid for the entire working day. In case that the order list is updated, the system must resolve a new problem from the beginning with the current set of orders. In the dynamic approach presented in this paper, the system is working continuously and every time a new order enters in the system, it is incorporated into the list of orders and considered immediately for obtaining efficient solutions.

As it is the case of the OBP, in the online version it is also possible to find different variants of the problem. As far as we know, the first contribution related to the OOBP can be traced batch to 1997, when a variant of the of the OOBP with multiple pickers was tackled in [44]. In this paper, the objective function consists of minimizing the turnover time. The authors proposed a simple FCFS method as a batching strategy and a Traversal (S-Shape) routing algorithm to determine the route for the pickers. Later, in 2007 [45] the OOBP was studied for multiple-block warehouses. In this case, the objective function tackled minimizes the average customer order throughput time, which considers the time dedicated to three of the main activities involved in the process of the orders (batching, picking, and sorting). Furthermore, the authors of this paper studied the influence of several factors of the problem in the determination of the optimal solution, such as the batch size or the allocation of the workers. Also, they compared different strategies for sorting the items within the same batch, such as Sort-While-Pick and the Pick-And-Sort. In 2011, Rubrico et al. [41] tackled the Online Rescheduling Problem with multiple pickers. They proposed two heuristic methods based on the Steepest Descent Insertion strategy, and on the Multistage Rescheduling strategy. Both combined with the S-Shape routing algorithm. In this case, the authors minimized the maximum total travel distance traversed by each picker. A combination of the study of the influence of multiple blocks and multiple pickers is tackled in [2]. The authors, studied the time window (Fixed Time Window vs Variable Time Window) which determine whether to wait for new orders or to try to sort again the orders into batches. This time, S-Shape and Largest Gap routing algorithms were compared, and the objective function is to minimize the maximum total travel distance traversed by each picker. Other recent variants of the problem tackles the OOBP integrated with the scheduling of the delivery, trying to minimize the service time of an order [47]. Later, in [46] the same problem was tackled, but considering multiple pickers. This time, the workload of the pickers is also studied. In [48], new constraints are introduced in the same problem. This time the authors considered multiple delivery zones and a capacity limit in the vehicle used to make the delivery.

The variant of the OOBP studied in this paper was first tackled in [18]. In this case the authors considered a single-block warehouse and only one picker. To tackle the problem, they proposed a heuristic algorithm based on Iterated Local Search (ILS) to minimize the maximum completion time of the customer orders, which arrive to the system within a certain time period. However, other objective functions, such as the maximum and the average time that an order remains in the system, were also reported. The proposed method was compared with a classical FCFS algorithm and with a Clarke and Wright method. This variant of the problem was also tackled later in [34], where the authors proposed a variant of the well-known Estimation of Distribution Algorithm (EDA) to tackle the problem. This time, the authors reported the average distance traversed by the picker as objective function. However, they did not compare their proposal with the latest algorithm in the state of the art for the problem [18], but with a previous version from the same authors based on Tabu Search methodology proposed in [21] and originally designed for the static version of the problem. Additionally,

the proposed EDA was not able to improve the results by the Tabu Search. Again, these authors used the S-Shape method as a routing strategy.

It is well documented that order picking operations are one of the most important and costly processes in a warehouse [3,7]. Moreover, if the two decisions concerning the order picking (i.e., batching and routing) are simultaneously considered, the associated benefits can be substantially increased. According to [6], it is possible to reduce the travel time up to 35%, simply by properly designing the routes of the order pickers. Therefore, a key element that strongly affects the performance of these algorithms is the sequence used by each picker to retrieve the items in each batch. This problem classifies as a Steiner Traveling Salesman Problem (see [5]) embedded in a special kind metric space with properties that can be exploited to develop powerful heuristics. Ratliff and Rosenthal [37] proposed a polynomial optimal procedure based on dynamic programming for routing in a rectangular warehouse. The procedure is computationally efficient for warehouses with no cross aisles; however, the efficiency decreases when the warehouse has cross aisles and, in some cases, the proposed routes seems to be *illogical* to the pickers who, as a result, deviate their routes from the specified ones [12]. Alternatively, different routing heuristics have been proposed in the literature (see [15,35,39]). Petersen in [35] carried out a number of numerical experiments to compare six routing methods, concluding that *composite* [35,36] and *largest gap* were the best options among the studied methods. Recent approaches [26] empirically found that the *Combined* method, originally proposed in [38] was the strategy which performed better in single-block rectangular-shaped warehouses.

In this paper, we propose a novel algorithm to tackle the online order batching problem. Particularly, our proposal is based on two well-known metaheuristics: Greedy Randomized Adaptive Search Procedure (GRASP) [10] and Variable Neighborhood Descent [29]. The former is used as a general framework to build efficient starting points for the VND, which is in charge of improving the solution provided by GRASP. The VND used here is a classical basic implementation of the VNS framework. The proposed method outperforms previous attempts in the state of the art.

The rest of the paper is organized as follows: in Sect. 2 we present the variant of the OOBP problem tackled in detail. In Sect. 3 we introduce the different algorithms proposed for solving the problem. Section 4 presents the main computational results obtained, when the algorithms are applied to different warehouse layouts. Finally, the main conclusions of the work and the outline of future research plans are collected in Sect. 5.

2 Problem definition

In this paper we tackle the online order batching problem with a single picker in a one-block warehouse. As it was described in Sect. 1, this problem consists of collecting all the orders made by customers which arrive to a warehouse, minimizing a predefined objective function.

An order is a list of items demanded by a customer which are stored in the warehouse. Orders are grouped in batches of a predefined maximum capacity, before being collected. All the orders in the same batch are collected together, by the same picker, in a single route. In this sense, an order can not be split into more than one batch.

An important issue, in the variant tackled in this paper, is that the orders that have to be managed in a working day, are not fully available at the beginning of the day, but they arrive to the warehouse while pickers are working. This is why the problem is considered online.

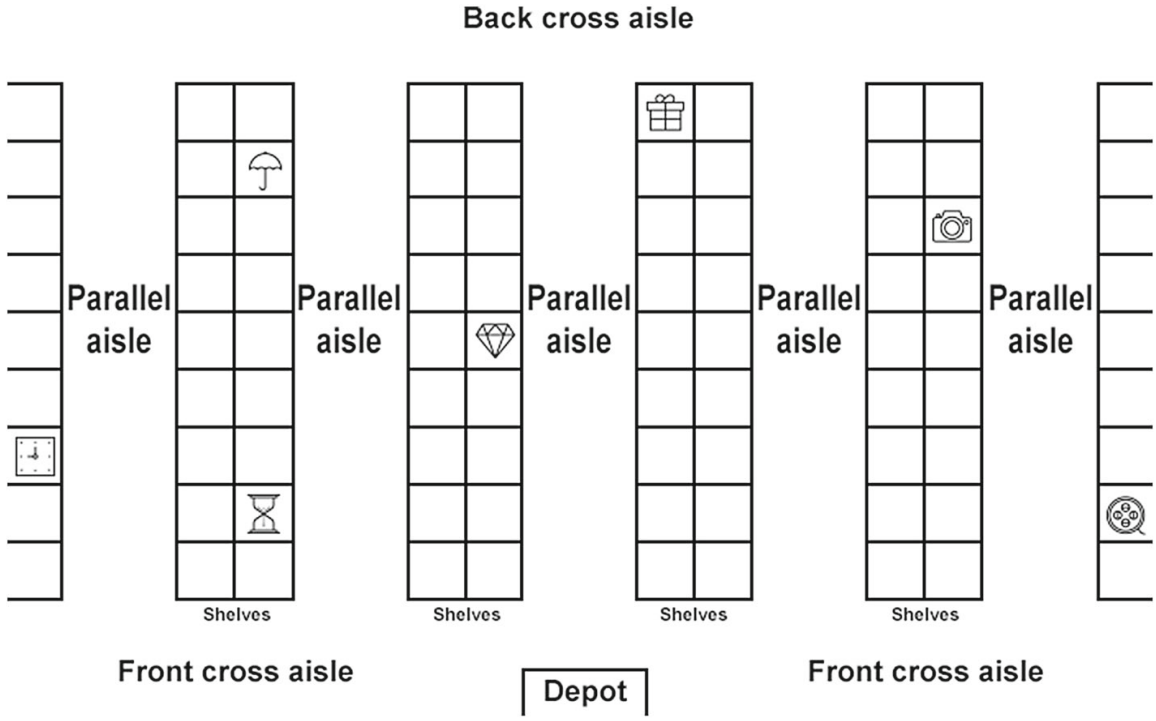


Fig. 1 Warehouse layout

The OOBP studied here considers only warehouses with a rectangular layout. This is, the warehouse is composed of two cross aisles (one at the front, and one at the back) and a variable number of parallel aisles. An example of this warehouse layout is depicted in Fig. 1. In this example, the warehouse has five parallel aisles formed by shelves at each side of the aisle, to store items. Particularly, each parallel aisle consist of 18 picking positions considering the shelves at both sides of the aisle.

Pickers start and finish their routes in a specific place of the warehouse, called depot. This depot is the place where the collected items must be handed once they have been retrieved. The depot is always placed in the front cross aisle, either in the middle of the aisle or in the leftmost corner.

The OOBP consist of three different tasks: batching, selecting, and routing. The batching task consists of grouping the orders already available in the system into batches. Then, once the batches are conformed, another algorithm has to select which, among all the conformed batches, is going to be collected next. Finally, a route to collect the orders within the selected batch need to be built. A picker will then start the retrieving process of all items in the batch, following the route previously built. When the picker finishes the collection of items and delivers them into the depot, a new batch and route is assigned to the picker to carry on his/her work.

In this paper we consider the minimization of two different objective functions related to the OOBP: minimizing the maximum completion time of all batches and minimizing the maximum turnover time of any order. These objective functions have been previously reported in the literature, however, they are considered separately (i.e., they can not be considered in a multi-objective optimization problem, since they are not necessary in conflict).

For a better understanding of these two objective functions, in Fig. 2 we show the life cycle of an order o_i through the timeline. Notice that, since this problem is considered online, the timeline is not bounded. This means that orders are arriving to the system continuously (i.e., 24 h a day/7 days a week) and we just observe what happens in the system in a particular

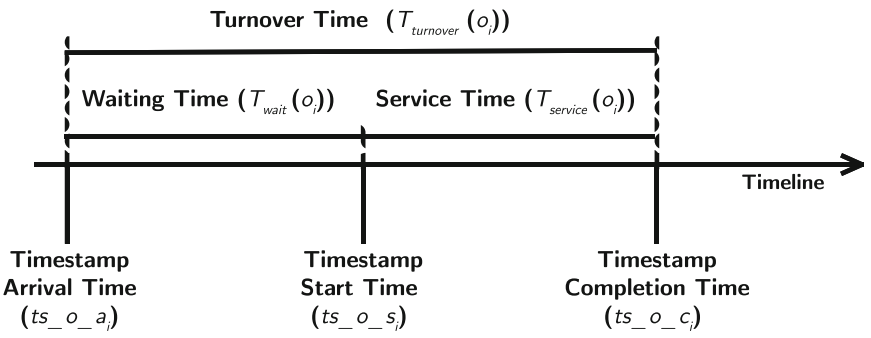


Fig. 2 Life cycle of an order o_i through the timeline

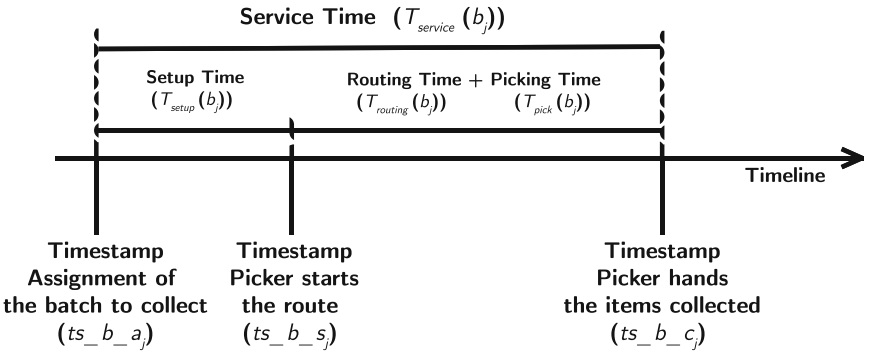


Fig. 3 Life cycle of a batch b_j through the timeline

chunk of time, in order to be able to observe the behavior of an algorithm and to compare it with other algorithms.

In this timeline we have highlighted three important timestamps (ts) in the life cycle of any order o_i: the arrival time ($ts\_o\_a_i$), the starting time ($ts\_o\_s_i$), and the completion time ($ts\_o\_c_i$). The arrival time represents the moment in the time when the order arrives to the warehouse. The starting time represents the moment in the time when the picker starts the route to collect the batch that contains the order. Finally, the completion time represents the moment in the time when the batch, that contains the order, is fully processed and the items in the orders collected are handed in the depot. With those three moments in the time at hand, it is also possible to define three periods of time in the life cycle of the order: the waiting time, $T_{wait}(o_i)$, the service time, also known as makespan, $T_{service}(o_i)$, and the turnover time, $T_{turnover}(o_i)$. The waiting time represents the time that an order remains in the system before being collected. It can be calculated as follows: $T_{wait}(o_i) = ts\_o\_s_i - ts\_o\_a_i$. The service time is the time needed to collect an order once the picker has started the route. It can be calculated as follows: $T_{service}(o_i) = ts\_o\_c_i - ts\_o\_s_i$. Finally, the turnover time is the time that an order remains in the system, either waiting, or being collected. It can be calculated as follows: $T_{turnover}(o_i) = ts\_o\_c_i - ts\_o\_a_i$.

It is important to highlight that, in the context of the OOBP, orders are not collected individually, but they are collected in batches (i.e., a group of orders that are collected together in a single route). Therefore, in Fig. 3 we represent a general schema of the life cycle of a batch through the timeline.

In this timeline we have highlighted three important timestamps in the life cycle of any batch b_j: the assignment of a batch to the picker ($ts\_b\_a_j$), the starting time when the picker initiates the route to collect the items in the orders assigned to the batch ($ts\_b\_s_j$), and the completion time of the batch ($ts\_b\_c_j$), when the picker hands the items collected in the route, into the depot. With those three moments in the time at hand, it is also possible to

Table 1 Parameters and variables for the OBP

Parameters		
n	$\rightarrow$	Number of customer orders available at the system in a given timestamp
m	$\rightarrow$	Upper bound of the number of batches (a straightforward value is $m = n$)
$v_{routing}$	$\rightarrow$	Routing velocity: number of length units that the picker can traverse in the warehouse per unit of time
v_{pick}	$\rightarrow$	Number of items that the picker can search and pick per time unit
w_i	$\rightarrow$	Number of items of order o_i for $1 \leq i \leq n$
W	$\rightarrow$	Maximun number of articles that can be included in a batch (device capacity)
Variables		
x_{ji}	$\rightarrow$	$\begin{cases} 1, & \text{if order } o_i \text{ is assigned to batch } b_j, \\ 0, & \text{otherwise} \end{cases}$

define different periods of time in the life cycle of any batch: the service time, $T_{service}(b_j)$, the setup time (T_{setup}), the routing time $T_{routing}(b_j)$, and the picking time, $T_{pick}(b_j)$. The setup time represents the time that the system and the picker need to prepare the picking cart, to receive and analyze the list of assigned orders (the batch) and to perform any other administrative task before the route starts. It is usually considered constant for any batch, and it is defined in the problem instance. Once the picker is ready to departure, the routing and pick times are the times needed by the pick to traverse the aisles, looking for the items in the orders assigned, and to extract (search and pick) any item from the shelves, respectively. The sum of the setup, routing and pick times, is known as the service time. Notice that the service time is the time available for the algorithms to compose a new disposition of batches, considering only the orders that have arrived to the warehouse in a previous moment in the time that have not been collected yet. Therefore, the service time can be calculated as follows:

$$T_{service}(b_j) = T_{routing}(b_j) + T_{pick}(b_j) + T_{setup}, \quad \forall j \in \{1, \ldots, m\}.$$

Next, we formally define the OBP based in the formulation presented in [18]. Notice that the OOBP is equal to the OBP if we consider a particular instant in the time (i.e., it only takes in consideration the orders already in the system, presupposing that no more orders will arrive later). First, in Table 1, we introduce the parameters and variables needed to define the problem correctly. Then, the objective functions and constraints which define the OBP variants tackled in this paper are presented.

The routing time is determined by routing algorithm. For the sake of simplicity, we consider a function $\mathrm{d}(b_j)$ that receives the orders in the batch b_j and returns the traveled distance to collect those orders. This function depends on the routing strategy that will be presented in Sect. 3.3. Therefore, considering a velocity $v_{routing}$, the routing time is defined as follows:

$$T_{routing}(b_j) = \frac{\mathrm{d}(b_j)}{v_{routing}}, \quad \forall j \in \{1, \ldots, m\}.$$

Considering that w_i is the number of items in the order o_i assigned to b_j, and v_{pick} is the number of items that the picker is able to search and pick per unit of time, the picking time

for a batch b_j can by defined as follows:

$$T_{pick}(b_j) = \sum_{i=1}^{n} \frac{w_i x_{ji}}{v_{pick}}, \quad \forall j \in \{1, \ldots, m\}.$$

Finally, T_{setup}, the setup time, is considered a parameter and it is specified in the particular instance.

The first objective is to minimize the maximum completion time of any order and it is given by:

$$\min \max_{j \in \{1,\ldots,m\}} \big(ts\_b\_s_j + T_{service}(b_j)\big). \tag{1}$$

It is worth mentioning that this objective is determined by the moment in the time in which the picker delivers the last batch.

The second objective function considered in this paper is to minimize the maximum turnover time of the received orders. The turnover time of an order o_i, $T_{turnover}(o_i)$, can be calculated as follows:

$$T_{turnover}(o_i) = \sum_{j=1}^{m} \big(ts\_b\_s_j + T_{service}(b_j)\big) x_{ji} - ts\_o\_a_i, \quad \forall\, i \in \{1, \ldots, n\}.$$

And then, the second objective function can be expressed as:

$$\min \max_{i \in \{1,\ldots,n\}} T_{turnover}(o_i). \tag{2}$$

Note that the value of this objective function is determined by the turnover time of the order that remains longer in the system.

The set of feasible solutions, in both cases, is given by the following constraints:

– Constraints in (3) guarantee that each order is assigned only to one batch:

$$\sum_{j=1}^{m} x_{ji} = 1, \quad \forall\, i \in \{1, \ldots, n\}. \tag{3}$$

– Constraints in (4) guarantee that the maximum capacity of each batch is not exceeded:

$$\sum_{i=1}^{n} w_i x_{ji} \leq W, \quad \forall\, j \in \{1, \ldots, m\}. \tag{4}$$

– Constraints in (5) guarantee that the batch b_j starts to be collected, once the batch b_{j-1} has been collected:

$$ts\_b\_s_j \geq ts\_b\_s_{j-1} + T_{service}(b_{j-1}), \quad \forall\, j \in \{2, \ldots, m\}. \tag{5}$$

– Constraints in (6) guarantee that the route for collecting a batch b_j can not start before the timestamps (moments in the time) when the orders o_i assigned to that batch have arrived to the system:

$$ts\_b\_s_j \geq ts\_o\_a_i x_{ji}, \quad \forall\, i \in \{1, \ldots, n\}, \text{ and } \forall\, j \in \{1, \ldots, m\}. \tag{6}$$

– Constraints in (7) and (8) state the non negativity of $ts\_b\_s_j$ and $ts\_o\_s_i$, respectively:

$$ts\_b\_s_j \geq 0, \quad \forall j \in \{1, \ldots, m\}. \tag{7}$$

$$ts\_o\_s_i \geq 0, \quad \forall i \in \{1, \ldots, n\}. \tag{8}$$

Finally, constraints in (9) state that variables x_{ji} are binary:

$$x_{ji} \in \{0, 1\}, \quad \forall\, j \in \{1, \ldots, m\} \text{ and } \forall\, i \in \{1, \ldots, n\}. \tag{9}$$

Despite of the fact that this model, proposed in [18], represents the OBP instead of the OOBP, it helps to understand the objective of the OOBP. Additionally, this formulation is a non-linear mixed integer programming model that can not be used to solve real instances using a solver. This partially supports the suitability of the use of heuristic algorithms in this context, as we propose in this paper.

It is important to remark that, in the case of the turnover time, the comparison of the value of the objective function of two solutions presents some additional difficulties. Let us introduce the concept of slot of time as the time between two consecutive departures of the picker. Notice that the construction of a solution in the context of the OOBP in a particular moment in the time consist of: i) grouping the orders available in the warehouse into batches, and ii) determining the sequence in which those batches should be collected. Then, this partial solution is evaluated and compared with other comparable partial solutions (i.e., those computed in the same slot of time). When the picker is ready for a new departure, the first batch sorted in the sequence of the best solution found during the slot of time is assigned to the picker. The process is then repeated.

Any order newly arrived to the warehouse must wait until the partial solution under construction is completed. Then, it can be included in the construction process of the next partial solution (no matters if the slot of time has not finished). Additionally, once all the orders have been assigned to a batch, the batch containing the oldest non-collected order (i.e., the first order arrived to the warehouse among the not collected ones) is selected to be assigned to the picker once the next slot of time starts.

3 Algorithms

In this section we describe our algorithmic proposal to tackle the OOBP. First, in Sect. 3.1 we describe the method in charge of the simulation of the general processes that happen in the warehouse. These methods include: considering the orders provided by a dispatcher; performing the batching of the considered orders; choosing the next batch to be collected; determining the route to collect the batch; and determining the departure moment of the picker. Once the general schema is at hand, in Sect. 3.2 we present the algorithms proposed to tackle the batching task. This section is divided into Sect. 3.2.1 where we introduce the Greedy Randomized Adaptive Search Procedure [10] as the constructive method, and Sect. 3.2.2 where we present the Variable Neighborhood Descent [29] used as a local search within the GRASP. Finally, in Sect. 3.3, we detail the routing strategy used.

3.1 General schema

The objective functions of the OOBP studied in this paper consists of either minimizing the total time elapsed in collecting all the orders arrived to a warehouse in a predefined time horizon, or alternatively, minimizing the turnover time. Due to the online nature of this problem, in order to compute these objective functions, it is necessary to have a general algorithm able to manage all the processes involved. We call this algorithm, the orchestration method and it is presented in Algorithm 1.

The orchestration algorithm receives two input parameters: the time horizon for the reception of orders ($maxTime$) and the list of pending orders at the beginning of the process ($listOrders$). Notice that this list of orders, in an online system, might refer to as the orders arrived at night time, or the orders pending to be collected from the previous working day. However, we consider that all the work from the previous day is already done.

The method runs while there are orders pending to be collected or the maximum allowed time has not been reached (step 4). Then, it checks if there are new orders arrived to the system (step 5) and updates the list of pending orders. Once this list has been updated, the batching algorithm is run (step 6) and the new solution is compared to the best found solution until the moment. Then, if there is a picker available and the solution contains batches not collected yet (step 8), the most suitable batch from the best solution ($bestSolution$) is chosen and the routing algorithm (step 10) will construct a route to collect the batch. Then, the picker will collect all the orders within the selected batch (step 11). Finally, the list of pending orders is updated, by removing the orders collected (step 12). Notice that the algorithm do not await until the picker comes back from its route but it is continuously running in order to have the best possible solution available as soon as the picker becomes available again.

Algorithm 1 Orchestration method

1: **Procedure** Orchestration($maxTime, listOrders$)
2: $pendingOrders \leftarrow listOrders$
3: $bestSolution \leftarrow \emptyset$
4: **while** (`getTime()` $< maxTime$) || ($pendingOrders \neq \emptyset$) **do**
5: $pendingOrders \leftarrow pendingOrders \cup$ `getNewOrders()`
6: $solution \leftarrow$ `batchingAlgorithm`($pendingOrders$)
7: `update`($bestSolution, solution$)
8: **if** `isPickerAvailable()` **then**
9: $batch \leftarrow$ `selectBatchAlgorithm`($bestSolution$)
10: $route \leftarrow$ `routingAlgorithm`($batch$)
11: `collect`($batch, route$)
12: `remove`($pendingOrders, batch$)
13: **end if**
14: **end while**

There are three remarkable methods within the orchestration procedure presented in Algorithm 1. The batching algorithm (`batchingAlgorithm`), which conforms the batches to be collected, is described in Sect. 3.2. The routing algorithm (`routingAlgorithm`), which determines the route that a picker must follow to collect a batch, is described in Sect. 3.3. Finally, the selection algorithm (`selectBatchAlgorithm`), which determines the next batch of the solution to be assigned to an available picker. In this case, we do not dedicate a whole section to the algorithm, since it follows a very simple heuristic. Particularly, this method selects the batch which contains the oldest order in the system. This method is commonly named as FIRST in the related literature [18].

3.2 Batching algorithm

As it was previously mentioned, the batching is one of the key procedures in the context of the OOBP. It consists of grouping all the orders received in a warehouse in a set of batches of a maximum predefined size with the aim of minimizing a particular objective function (typically the distance needed to collect all the orders).

In this paper we propose a batching algorithm based on the combination of a GRASP procedure (presented in Sect. 3.2.1 and used as a constructive method) and a VND (presented in Sect. 3.2.2 and used as a local search within the GRASP).

3.2.1 Greedy Randomized Adaptive Search Procedure

The Greedy Randomized Adaptive Search Procedure was introduced in [10] as a multi-start method to find high-quality solutions to hard optimization problems. Each iteration is composed of two steps: (1) construction and (2) improvement. The general schema of GRASP is presented in Algorithm 2. The construction phase combines the greediness of a particular greedy function and the randomization of some decisions during the construction. The constructive procedure within GRASP (step 4) proposed in this paper is explained next. On the other hand, the improvement phase (step 5), usually based on a local search method, is in this case based on a metaheuristic procedure. Particularly, we have replaced the local search with a VND method, explained in detail in Sect. 3.2.2. Therefore, in each iteration, the GRASP method constructs an efficient solution, and this solution is further improved with a VND procedure. This GRASP schema is repeated until the method runs out of time, or the maximum number of iterations is reached.

Algorithm 2 Greedy Randomized Adaptive Search Procedure

1: **Procedure** GRASP($maxTime$)
2: $bestSolution \leftarrow \emptyset$
3: **repeat**
4: $\quad solution' \leftarrow$ `Constructive()`
5: $\quad solution'' \leftarrow$ `LocalSearch`($solution'$)
6: $\quad bestSolution \leftarrow$ `Update`($bestSolution, solution''$)
7: **until** `getTime()` $> maxTime$
8: **return** $bestSolution$

The GRASP constructive method proposed in this paper is presented in Algorithm 3 and it is based on two basic principles: a greedy function that selects a group of candidate items to be added to the solution in the next iteration, and a particular randomization of the decisions made by that function. The method starts from an empty solution (step 2) and in each iteration it adds a new order to the solution. All available orders are initially inserted in the so called Candidate List (CL) (step 3). We propose the use of a greedy function (`f`) based on the weight of the orders in the CL in such a way that heaviest order is considered first (ties are broken at random). Therefore, the GRASP constructive sorts all the orders, already in the system but not yet assigned to a batch, in a descending way with respect to its weight. Then, a threshold th is calculated (step 6) based on the maximum ($\arg\max$ `f`(CL)) and minimum ($\arg\min$ `f`(CL)) weight of any order in the CL, and a random value $\alpha \in U[0, 1]$ (step 4). This threshold is used to determine the percentage of the best candidates to be included in a new list, called Restricted Candidate List (RCL) (step 7). Finally, an order is chosen at random from this RCL (step 8), and it is included in the partial solution being constructed in this iteration. The chosen order is inserted in the first batch with enough available space (step 9) and it is removed from the CL (step 10).

Notice that once a new order is added to the solution, it is necessary to determine the batch where it will be allocated. In the first iteration, a new empty batch is created. Then, in the following iterations, the algorithm tries to insert the selected order in this batch and

if it does not fit in the batch, then another empty batch is created to allocate this order, and so on. Notice that the sequence of batches used to try the insertion of the order is the same sequence in which the batches are created.

Classical stopping criteria for the GRASP are related to a particular number of iterations or a predefined time horizon. However, in this paper, we have run the GRASP method as long as there is not an idle picker. This means that each time that a new batch is assigned to a picker, the GRASP procedure is run again, but this time it does not consider any order which is being collected (i.e., it is in the assigned batch) or it has been collected before.

It is worth mentioning that due to the online context of this problem, the CL is being updated every time that the GRASP procedure starts again, with the latest orders arrived to the system.

Algorithm 3 Constructive procedure

1: **Procedure** Constructive($listOrders$)
2: $solution \leftarrow \emptyset$
3: $CL \leftarrow listOrders$
4: $\alpha \leftarrow$ `getRandomValue()`
5: **while** $CL \neq \emptyset$ **do**
6: $\quad th \leftarrow \arg\max\ \mathtt{f}(CL) - \alpha(\arg\max\ \mathtt{f}(CL) - \arg\min\ \mathtt{f}(CL))$
7: $\quad RCL \leftarrow$ `buildRestrictedCandidateList`(th, CL)
8: $\quad order \leftarrow$ `randomOrderSelection`(RLC)
9: $\quad$ `insertOrder`$(solution, order)$
10: $\quad CL \leftarrow CL \setminus \{order\}$
11: **end while**
12: **return** $solution$

3.2.2 Variable Neighborhood Descent

Variable Neighborhood Search is a metaheuristic proposed by Mladenović and Hansen in 1997 as a general method to solve hard optimization problems. The authors introduced the idea of changing the neighborhood structure within the search in order to reach different local optima. There are many variants of VNS. Some of the best-known are: Reduced VNS (RVNS), Variable Neighborhood Descent (VND), Basic VNS (BVNS), General VNS (GVNS), Skewed VNS (SVNS), and Variable Neighborhood Decomposition Search (VNDS) [16,17,29]. However, more recent approaches have appeared in the last few years, such as: Variable Formulation Search (VFS) [33], Parallel Variable Neighborhood Search (PVNS) [9,28], or Multi-Objective Variable Neighborhood Search (MO-VNS) [8].

In this paper, we propose the use of a VND procedure as a local search within the GRASP methodology. VND was proposed in the context of Variable Neighborhood Search in [29], as a general strategy to systematically explore a group of neighborhoods. The obtained result of a VND procedure is a local optimum with respect to all the neighborhood structures considered. The final order of the studied neighborhoods determines the performance of the method. Typically, neighborhood structures are sorted from the smallest and fastest to explore, to the largest one and slowliest to explore. However, this rule must be empirically tested when considering a particular problem and the associated neighborhood structures.

We use a standard and basic implementation of VND, which can be considered as the classical VND method. In Algorithm 4 we present the pseudocode of the VND procedure proposed in this paper. Particularly, this algorithm receives an initial solution as starting point

and it considers three different neighborhood structures (named in the pseudocode as N_1, N_2, and N_3) explored by a local search procedure. This local search procedure follows a first improvement strategy. The particular neighborhoods proposed are detailed ahead. The method will initially explore the first neighborhood (N_1) in step 7 and then it will determine if an improvement has been made (step 13) or not. If a neighborhood is not able to improve the current solution, then the method will jump to the next available neighborhood (step 17) until the maximum number of neighborhoods is reached (step 19). When the exploration of a neighborhood improves the current solution, the best solution found is updated and the method starts again from the first neighborhood (step 15).

Algorithm 4 Variable Neighborhood Descent

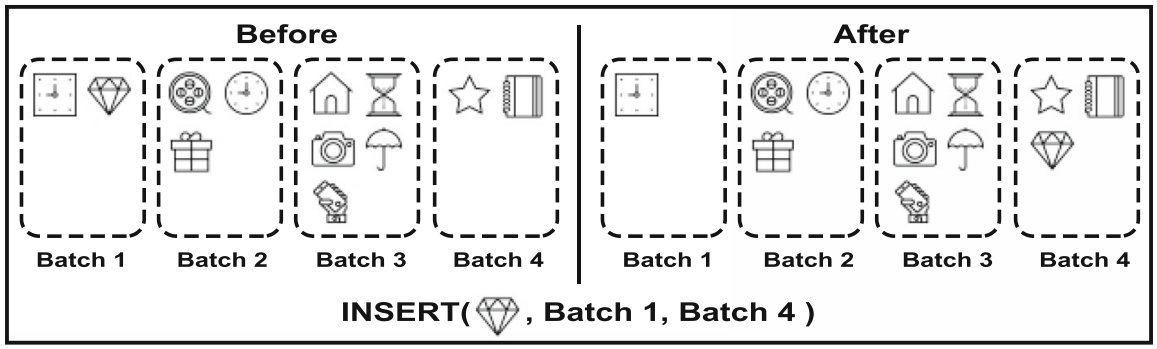
1: **Procedure** VND($solution$)
2: $k \leftarrow 1$
3: $k_{max} \leftarrow 3$
4: $bestSolution \leftarrow solution$
5: **repeat**
6: **if** $k == 1$ **then**
7: $solution' \leftarrow$ `LocalSearch`($bestSolution, N_1$)
8: **else if** $k == 2$ **then**
9: $solution' \leftarrow$ `LocalSearch`($bestSolution, N_2$)
10: **else if** $k == 3$ **then**
11: $solution' \leftarrow$ `LocalSearch`($bestSolution, N_3$)
12: **end if**
13: **if** `evaluate`($solution'$) < `evaluate`($bestSolution$) **then**
14: $bestSolution \leftarrow solution'$
15: $k = 1$
16: **else**
17: $k = k + 1$
18: **end if**
19: **until** $k > k_{max}$
20: **return** $bestSolution$

We propose in this paper three different neighborhood structures to tackle the OOBP. These neighborhood structures are named `Insert`, `Swap1`, and `Swap2` respectively, and they are graphically shown in Fig. 4.

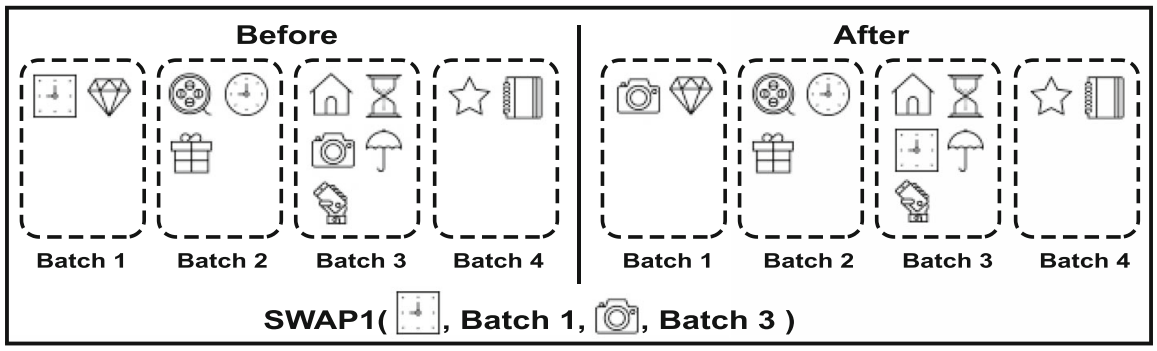
The `Insert` neighborhood, represented by an example in Fig. 4a, considers all possible solutions reached by the insertion of any order in the solution into all the available batches. In the example depicted in Fig. 4a it is represented the insertion of a diamond (originally allocated in Batch 1) into Batch 4. We depict the configuration of the batches before and after the `Insert` operation.

The `Swap1` neighborhood, represented by an example in Fig. 4b, considers all possible solutions reached by the interchange of any pair of orders in a different batch in the solution, into all the available batches. In the example depicted in Fig. 4b it is represented the exchange of a clock (originally allocated in Batch 1) with a photo camera (originally allocated in Batch 4). We depict the configuration of the batches before and after the `Swap1` operation.

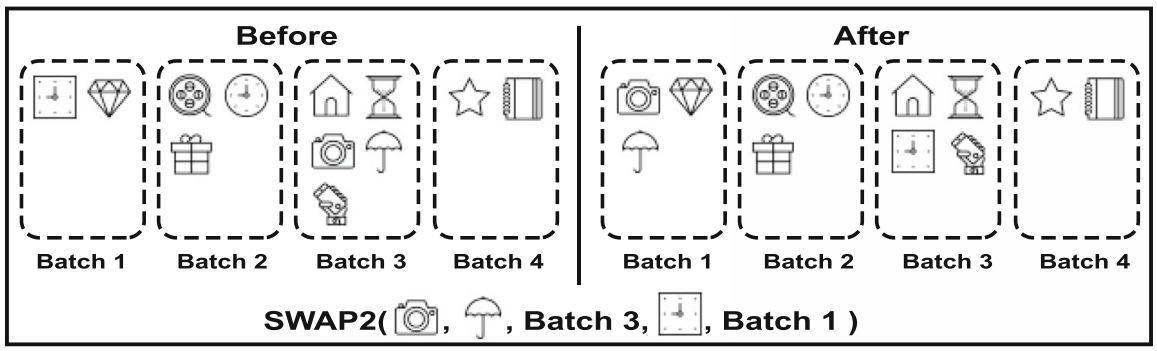
Finally, the `Swap2` neighborhood, represented by an example in Fig. 4c, considers all possible solutions reached by the exchange of every pair of two orders within the same batch, with any single order in any other batch. In the example depicted in Fig. 4c it is represented the exchange of a photo camera and an umbrella (originally allocated in Batch 3) with a clock (originally allocated in Batch 1). We depict the configuration of the batches before and after the `Swap2` operation.



a Insert 1x0.



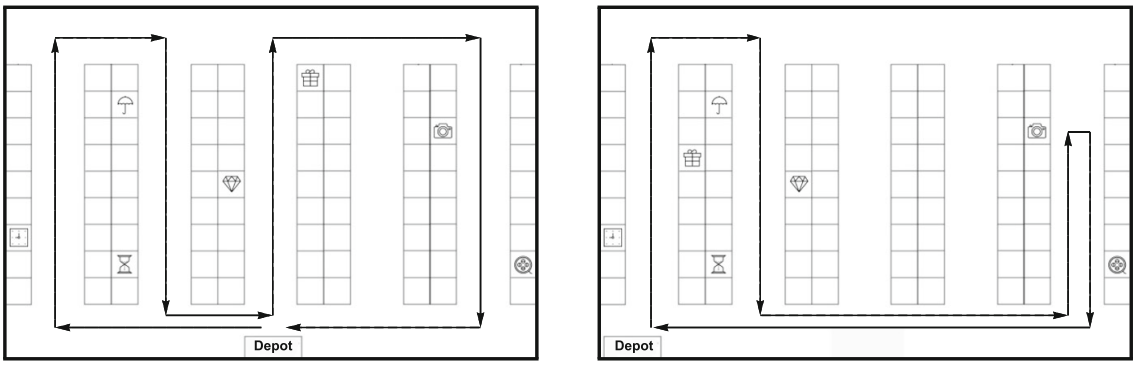
b Swap1 1x1.



c Swap2 2x1.

Fig. 4 Neighborhood structures

Notice that it is mandatory that the resulting batches do not violate the maximum capacity restriction on any batch. Otherwise the operation is considered unfeasible and, in this case the obtained solution after a move is not considered as part of the studied neighborhood.



a Warehouse layout with the depot placed in the center of the front cross aisle.

b Warehouse layout with the depot placed in the left corner of the front cross aisle.

Fig. 5 Route examples calculated with the S-Shape strategy

3.3 Routing algorithm

The routing algorithm determines the route that a picker must follow in order to collect all the items within the orders of the same batch. This route always starts and ends in the same point, the depot. The depot is placed in the front cross aisle, either in the center or in the left corner of the aisle.

In the literature there are a variety of algorithms to solve the routing problem in a warehouse. This problem can be considered a variant of the well-known Traveling Salesman Problem. There are heuristic, metaheuristic, and exact algorithms for solving the problem. In our case, we use the S-Shape heuristic method introduced in [5,14] that is widely used in the literature of the OBP and OOBP due to its simple implementation and fast performance. Additionally, the obtained routes are easily understandable by the pickers.

Given a batch, the S-Shape method identifies those parallel aisles where there are items to collect (at least one item). Then, each of those aisles are completely traversed from one cross aisle (either the front or the back cross aisle) to the opposite cross aisle. The first parallel aisle to be traversed is the leftmost aisle that contains an item to be collected. Then, the picker will enter only in the parallel aisles that contain at least one item that need to be collected. This process is repeated until the last aisle with items is traversed. Then, the picker returns to the depot using the front cross aisle. Notice that if the picker has to traverse an odd number of parallel aisles, the picker will enter in the last aisle from the frontal cross aisle. In this case, the picker will travel up to the most distant item and then he/she will perform an U-turn returning to the front cross aisle.

In Fig. 5 we show an example of two different routes designed using the S-Shape algorithm through a rectangular warehouse. Particularly, in Fig. 5a it is shown an example of a warehouse layout with the depot placed in the center of the front cross aisle. In this case, the route determined by the S-Shape algorithm traverses 4 different aisles before coming back to the depot. In Fig. 5b it is shown a different example of a warehouse layout where the depot is placed in the left corner of the front cross aisle. In this case, the picker must traverse an odd number of parallel aisles. Then, the two first parallel aisles are fully traversed and, in the last one, the picker performs an U-turn when it reaches the last item to collect.

4 Computational results

In this section we present the computational results obtained with the algorithms proposed in Sect. 3. First, in Sect. 4.1 we present the instances used in the experimentation. Then, we describe in Sect. 4.2 the configuration of the dispatcher of orders which determines the moment in the time of the arrival of each order to the system. In Sect. 4.3 we perform a set of preliminary experiments to adjust the parameters of the proposed methods and, additionally, to show the merit of the different parts of the final algorithm. Finally, in Sect. 4.4 we compare our best variant with the current state of the art of the OOBP for the tackled version of the problem.

The experiments were run 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, and all the codes were developed in Java 8.

4.1 Instances

In order to test the algorithms proposed in this paper, we have selected two sets of instances derived from the previous literature in the state of the art of the OOBP. Notice, that an instance for the OOBP is formed by a group of parameters derived from: the warehouse characteristics, the list of orders, and the arrival scheduler. The warehouse is also defined by a certain group of parameters such as: the number of aisles, the length of each aisle, the sorting policy, etc. The list of orders indicates the products that must be collected to satisfy the demand by the customers. Finally, the arrival scheduler determines the moment in the time when a particular order arrives to the warehouse.

All the instances within the selected data sets present common characteristics, as far as the warehouse shape is concerned: rectangular shape, one block with two cross aisles (one at the front, one at the back) and a variable number of parallel aisles. This warehouse distribution is the same previously presented in Fig 1. However, despite of the fact that all instances have the same warehouse structure we consider different warehouse types, which differ in other parameters such as: the number of parallel aisles, the length of the aisles, the width of the aisles, the number of picking positions per aisle, etc. For each warehouse type, additionally, we study different lists of orders. These lists vary in the number of orders and each order varies in the number and composition of items.

The first data set used in the experimentation was introduced in [1] and it was originally defined for the static version of the OBP. The summary of the characteristics of this dataset is reported in Table 2. This dataset includes four different warehouse configurations (denoted as W1, W2, W3, and W4). For each warehouse configuration there are lists of orders whose size ranges from 50 to 250 orders. From the total number of instances originally proposed by the authors (2400), a selection of instances was made in [26] in order to establish a reduced dataset. In this paper, the authors found that using the whole dataset did not provide significant differences when compared to using only a reduced selection of instances. In this paper, we have used 64 instances from the reduced data set in order to perform our experiments.

The second data set used in our experiments was originally proposed in [18]. This data set is composed of 1600 instances. Again, a selection of instances was made in [26] in order to have a reduced data set which can be handled in a reasonable amount of time. In this case, we have also selected 64 diverse instances for our experiments. In Table 3 we present the main characteristics associated to this warehouse (W5). This time, the size of the lists of orders ranges from 40 to 100 orders.

Table 2 Warehouse characteristics [1]

	W1	W2	W3	W4
Storage policy	Random/ABC			
Depot position	Center/corner			
Order size	U(1, 7)	U(2, 10)	U(5, 25)	U(1, 36)
Item weight	1	1	1	U(1, 3)
Order picker capacity (weight)	12	24	150	80
Number of parallel aisles	4	10	25	12
Number of items per aisle	2×30	2×20	2×25	2×16
Total number of items	240	400	1250	384
Parallel aisle length (m)	50	10	50	80
Centre distance between two aisles (m)	4.3	2.4	5	15

Table 3 Warehouse characteristics [18]

	W5
Storage policy	Random/ABC
Depot position	Center
Order size	U(5, 25)
Item weight	1
Order picker capacity (weight)	30/45/60/75
Number of parallel aisles	10
Number of items per aisle	2×45
Total number of items	900
Parallel aisle length (m)	45
Centre distance between two aisles (m)	5

Notice that the two datasets selected have been widely used in the context of the OBP [4,24–28,42,43] and the OOBP [34,47,49]. Also, it is important to notice that the experiments in the context of the OOBP are performed in real time. This means that the time horizon of the arrival of orders is the minimum time that the algorithm needs to be run. Therefore, in the context of the OOBP it is not possible to consider a very large number of instances nor very long time horizons for the arrival of orders. In Table 4 we describe all the important parameters used in our experimentation. Among others, we include the parameters related to the arrival of orders to the warehouse. This configuration is common for the 2 data sets considered. In brief, the time horizon for the arrival of orders has been set to 4 h and the statistical distribution of the time instants of the arrivals is determined by an exponential distribution.

When considering the turnover time (maximum time that an order remains in the system) as objective function, it is necessary to set a moment in the time as a reference time. This allows the algorithm to calculate the amount of awaiting time of a particular order in the system with respect to that moment in the time. Otherwise, since the algorithm is continuously exploring new solutions, it is not possible to compare a solution with other one obtained in a previous moment, since the reference point in the time varies. This reference time is fixed and then updated each time that a picker receives a new batch to collect.

Table 4 Configuration of the parameters in the experiments

Arrival period	4 h
Dispatcher distribution	Exponential
Velocity of the picker (LU/min)	48
Time pick by item (items/min)	6
Setup time (min)	3
Time window fit (s)	400
Initial order in queue	0
Random seed	50
Routing strategy	S-shape
Start picking strategy	As soon as picker is available
Batch selection strategy	Batch with the oldest order

Table 5 Lambda values

#Orders	40	60	80	100	150	200	250
λ	0.167	0.250	0.333	0.417	0.625	0.833	1.042

4.2 Order dispatcher

The arrival of orders to the warehouse in the context of the OOBP 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 to calculate the batch configuration and later the picking route. The time horizon of the arrival of orders to the warehouse is set to 4 h.

In order to simulate the arrival time for each order, we follow a Poisson point process. Since the time horizon is set to 4 h ($t = 4$). 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.

4.3 Preliminary experiments

In order to determine the best configuration of our algorithm and, also to test the contribution of each part of the algorithm, we have performed a set of preliminary experiments. To that aim, we have selected a subgroup of instances from the total datasets. Particularly, we have selected 16 out of 128 instances to perform the preliminary experiments. These instances have been selected in order to include in the reduced dataset the most diverse instances.

4.3.1 Determination of the best GRASP parameters

The GRASP constructive algorithm introduced in Sect. 3 has two main parameters: (i) the value of the α parameter, and (ii) the number of constructions. As far as the value of α is concerned, it determines the voracity in the selection of orders to be included in the Restricted Candidate List. In this paper, we use a greedy criterion based on the weight of the orders (the heaviest order is considered first).

Table 6 Performance of GRASP with different α values

	0.0	0.2	0.4	0.6	0.8	1.0	Random
Dev. (%)	0.84	0.64	0.59	0.53	0.62	1.03	0.42
#Best	3	3	1	3	1	1	4

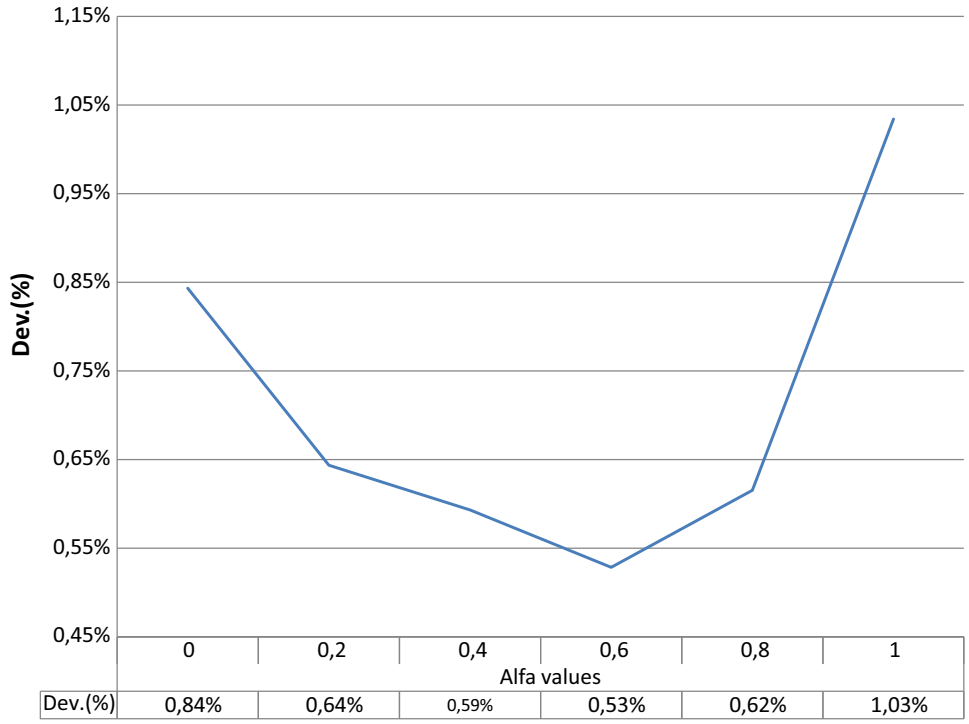


Fig. 6 Representation of the average deviation obtained with different α values

In the first preliminary experiment we test different values of α in order to determine the most suitable value. In Table 6 we report the average deviation with respect to the best value found in the experiment, and the number of best solutions found for different values of the parameter α. Notice, that when $\alpha = 1.0$ the inclusion of orders in the Restricted Candidate List is fully random. On the other hand, when $\alpha = 0.0$, the inclusion of orders in the Restricted Candidate List is fully greedy. The deviation obtained with the different values of α is (Alfa) represented in Fig. 6. As it is shown in the figure, the evolution of the average deviation to the best value found decreases until $\alpha = 0.6$ and increases from this point. However, the differences among the different values of α studied are very small. Therefore, we have also included in the Table 6 the results obtained when α takes a different and random value in each GRASP iteration. As it is possible to see in the table, the best solutions found in the experiment are obtained when α is set to a random value. Notice that this experiment has been performed by running the GRASP constructive algorithm in isolation for a time limit of 60 s.

Next, we test the influence of the second parameter related to GRASP (i.e., number of constructions). In this case, we are not interested in fixing the number of constructions of the algorithm, since it will depend on the available time between the routes of the picker. However, it is interesting to review the influence of the constructive procedure in the quality of the initial solution.

In Table 7 we report the evolution in the deviation of the GRASP constructive method with respect to the best value of the experiment, when considering from 10 to 10,000,000 of

Table 7 Evolution of deviation when increasing the number of GRASP constructions

	10	100	1,000	10,000	100,000	1,000,000	10,000,000
Dev. (%)	4.66	3.22	2.60	1.82	1.07	0.56	0.00
#Best	0	0	0	0	0	1	15
CPUt (s)	0.01	0.02	0.12	0.92	9.11	90.35	900.96

Table 8 Different neighborhood orders within the VND

	$\{N_3, N_1, N_2\}$	$\{N_3, N_2, N_1\}$	$\{N_3, N_2, N_1\}$	$\{N_2, N_1, N_3\}$
Dev. (%)	0.67	1.11	0.93	1.06
#Best	4	3	4	6
CPUt (s)	14.59	8.98	10.60	12.77

constructions. Particularly, we find big differences in the improvement of the deviation from 10 to 100 constructions, but this improvement is reduced when we increase the number of constructions. However, the time increases considerably. In order to relate this values with the real execution of the algorithm it is important to notice that an average picking route might last from 250 to 1200 s.

4.3.2 VND neighborhoods

The VND algorithm proposed in Sect. 3 includes three different neighborhoods (see Sect. 3.2.2), named as N_1, N_2, and N_3. One of the key decisions with respect to the neighborhoods included in a VND is determining the order in which they are placed within the algorithm. Many researchers set the neighborhood order depending on it size (smaller neighborhoods come first since they are faster to traverse). However, we have considered to place in first position not only the smaller neighborhood (N_3) but also the second smallest (N_2). With this at hand, we have reported the performance of the possible different orders of the neighborhoods within the VND (see Table 8).

As we can see in the Table 8 the best combination of neighborhoods is N_3, N_1, and N_2, with an average deviation of 0.67%. However the combination with the largest number of Best values is N_2, N_1, and N_3. In this case, we have decided to use the combination with the smallest deviation (N_3, N_1, and N_2) for our final configuration.

4.3.3 Contribution of each neighborhood

A key parameter when designing a VND is checking that all the neighborhoods included in the algorithm contribute to the process of search. In case any neighborhood do not show improvements it should be removed and the saved time used for other tasks. To test this fact, we have reported the number of improvements made with each neighborhood when they are combined in the VND method. Taking in consideration the preliminary data set, we have provided a random solution as starting point for the VND. In Table 9 we report the number of improvements performed by the local search which traverses each neighborhood. As it is shown in the table, all the neighborhoods produce improvements in the search. Particularly,

Table 9 Improvements produced by each local search within the VND

	#improvements	%improvements
N_1	24	35.82
N_2	26	38.81
N_3	17	25.37

Table 10 Comparison of the VND with respect to the local search

	VND	LS-1	LS-2	LS-3
Dev. (%)	0.12	5.42	8.34	10.21
#Best	15	1	0	0
CPUt (s)	22.89	4.03	13.83	1.13

N_1 and N_2 are the neighborhoods which produce the largest number of improvements (24 and 26 respectively). However, the difference with respect to N_3 is not very remarkable.

4.3.4 Comparison between VND and local search procedures

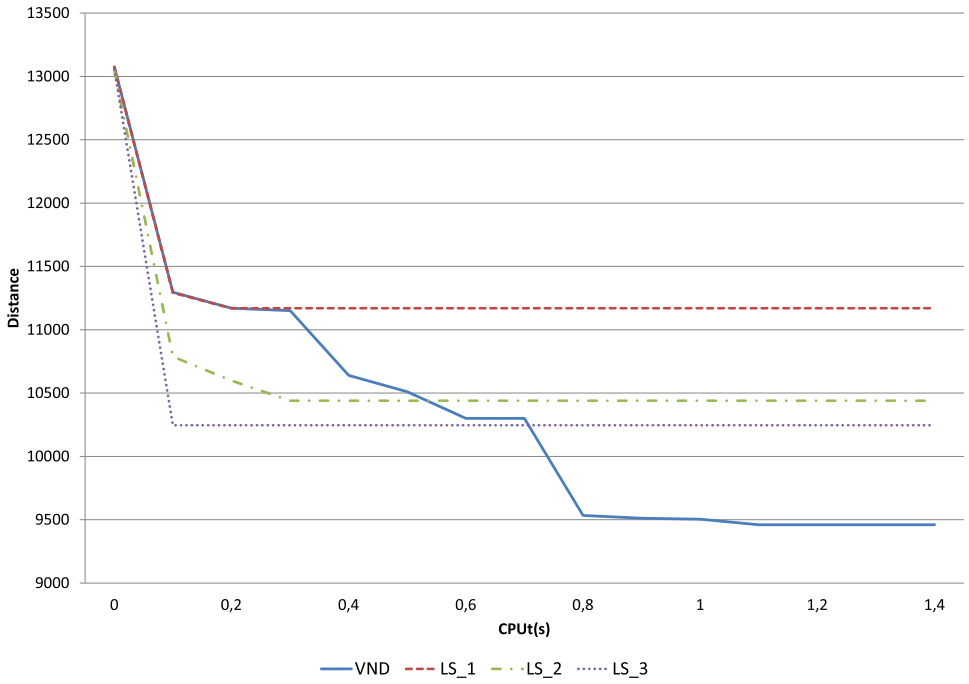
A local search procedure runs until no further improvements can be made in a neighborhood for a particular objective function, i.e., the obtained solution is a local optimum. On the other hand, the combination of different local search procedures within a VND must provide a local optimum with respect to all the neighborhood structures considered. Additionally, it is expected that the quality of the solution found by a VND procedure is better than the solution found by each local search in isolation.

In Table 10 we report the average deviation, the number of best solutions found, and the CPU time of four different algorithms. On one hand, we have run the VND procedure starting from a random solution, which was provided as an input parameter. This random solution was also provided to each of the local search procedures separately. In this case, it is possible to notice that the VND procedure is able to find a much better solution than each local search in isolation. However, as it was expected, the CPU time of the VND is larger than the CPU time of each local search independently.

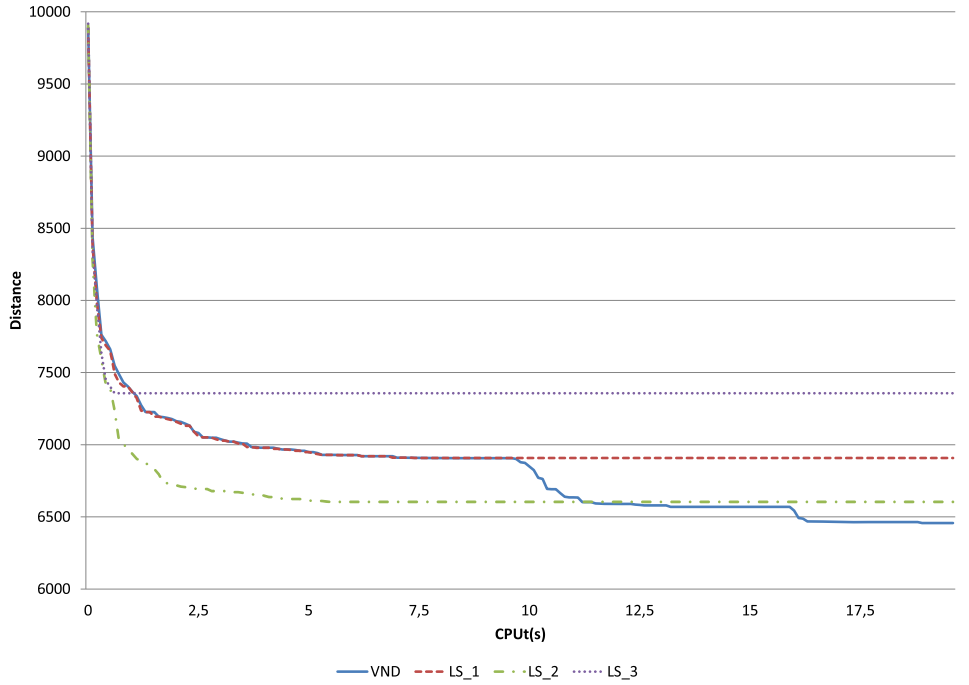
To complement the results presented in Table 10 we present the evolution in the objective function value over the time of each local search procedure compared to the VND method. In this experiment we have selected two representative instances from the preliminary dataset: one small (80 orders, represented in Fig. 7a) and one large (250 orders, represented in Fig. 7b). As it is shown in the figures, the performance of the VND is the best among all the methods compared, for the two instances considered. However the local search procedures converge to a local optimum faster than the VND.

4.3.5 Performance of the static exact approach in the state of the art

Our last preliminary experiment is devoted to illustrate the suitability of using heuristic approaches in the context of the OOBP, instead of using an exact approach. To that aim we have tested the performance of the mathematical formulation introduced in Sect. 2 over the preliminary data set. It is important to notice that the mathematical model proposed in the state of the art was designed for the static version of the problem (i.e., all the orders are available at the beginning of the process). Additionally, the model is incomplete, and it can



a Instance $W5\_abc2\_80\_46$ composed of 80 orders.



b Instance $W2\_250\_090$ composed of 250 orders.

Fig. 7 Evolution of the Local Search procedures and the VND method over two particular instances

not be directly executed in a solver, since the function "d" (which calculates the distance of a given solution) is not defined in the original paper where the model was proposed. In order to overtake this difficulty we have used the distance formulation proposed in [31].

We run the mathematical model for each instance on Gurobi 9.0.0rc2 (win64). Since the sizes of the instances considered are quite large for the model, we reduced the size of each instance by selecting a reduced number of orders (choosen at random) from each instance. We started in 5 orders per instance and increased the number of orders per instance one

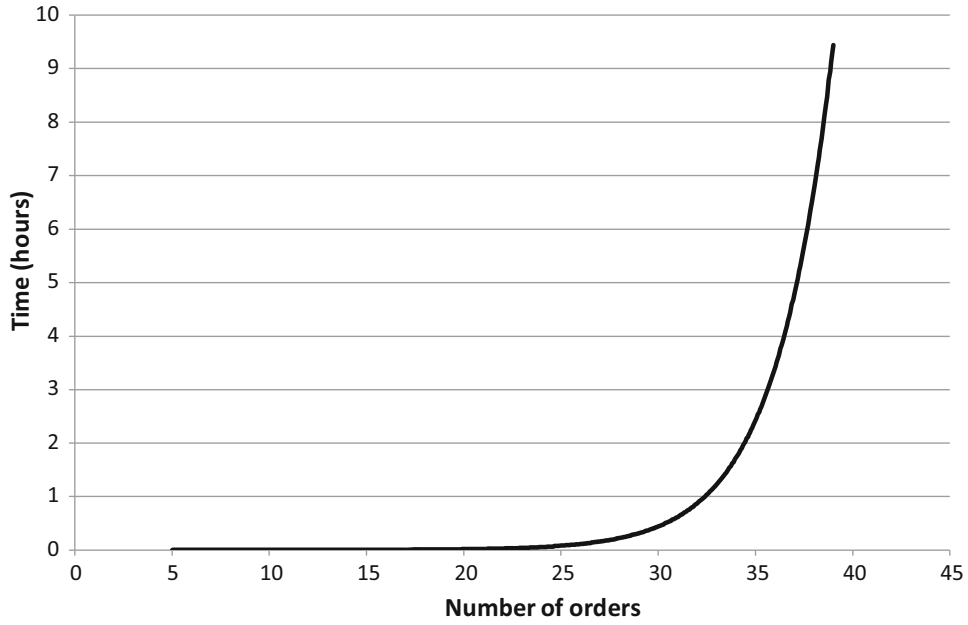


Fig. 8 Average performance of the mathematical formulation introduced in [18]

by one. As expected, the mathematical model was able to solve small-size instances to the optimal solution. However, when reaching the instance sizes considered in this paper (larger than 40 orders per instance) the time needed exceeded the time horizon considered.

In Fig. 8 we report the nonlinear regression obtained when considering the results provided by the solver, which shows the trend in the increase of time when the size of the instance also increases.

4.4 Final experiments

Once we have tested the different components of our algorithms, we have configured our best variant of the GRASP + VND method with the following parameters: the number of iterations of GRASP + VND is not predefined. The procedure is run as many times as possible between the departure of the picker to collect a batch, and the arrival of the picker to the depot after collecting all the items, plus the setup time. The α value in the GRASP constructive procedure has been set to random and it is reset in each iteration of the method. The three neighborhoods proposed in Sect. 3.2.2 have been included in the VND following the order: N_3, N_1, and N_2.

The best configuration of our algorithm GRASP + VND (GR+VND in the tables) is compared with the best method in the state of the art, the Iterated Local Search (ILS) proposed in [18], and denoted in the final experiments as ILS. This final comparison has been performed over the two datasets previously presented. We report the summary of the results in Tables 11 and 12.

In Table 11 we consider the minimization of the maximum completion time as objective function. This objective function was proposed in [18] and it returns the moment in the time when the last processed order is completed. In other words, this objective function equals the total time needed to collect all the orders arrived to the warehouse. The results for the other objective function considered in this paper are reported in Table 12. In this case, the comparison with the state of the art considers the minimization of the maximum turnover time of an order as objective function. This is the maximum time that an order stays in the

Table 11 Comparison with the state of the art considering the minimization of the maximum completion time as objective function

	Albareda		Henn		Total	
	GR + VND	ILS [18]	GR + VND	ILS [18]	GR + VND	ILS [18]
Dev. (%)	0.10	3.38	0.11	2.41	0.11	2.90
#Best	60	4	55	11	115	15

Table 12 Comparison with the state of the art considering the minimization of the maximum turnover time of an order as objective function

	Albareda		Henn		Total	
	GR + VND	ILS [18]	GR + VND	ILS [18]	GR + VND	ILS [18]
Dev. (%)	0.29	8.14	0.63	5.80	0.46	6.97
#Best	61	3	52	12	113	15

system (i.e. the difference between the arrival time to the warehouse and the time when it is completed).

Despite of the fact that we provide both objective functions, we consider that the most interesting one, attending to real scenarios for this problem, is the turnover time (see Table 12). In fact, our algorithms were only configured to minimize this objective function (not the maximum completion time). We only report the maximum completion time for the solutions found when minimizing the maximum turnover time. The main reason of including the maximum completion time is for the sake of clarity, since this objective function was the main objective function proposed in [18].

Considering the minimization of the maximum completion time reported in Table 11, we appreciate that the GRASP + VND is the best overall method, since it achieves a 0.11% of deviation with respect to the best values found in the experiment, while ILS achieves a 2.90% of deviation. Additionally, the number of best solutions found by GRASP + VND (115) is much larger than the number of best solutions found by ILS (15).

As far as the minimization of the maximum turnover time is concerned, we report the obtained values by each algorithm in Table 12. Again, GRASP + VND appears to be the best method. In this case, the obtained deviation (0.46%) is considerably better than the one by ILS (6.97%). The number of best found solutions by GRASP + VND (113) is also much larger than the number of best solutions found by ILS (15).

To complete the comparison of the results, we have carried out a test of mean difference of the results obtained by the two methods. Even when the hypothesis of normality is not verified, the size of the sample used (128 in total, 64 from Albareda and 64 from Henn) allows us to use the t-test for the difference of means in paired samples (note that both methods have been tested on the same instances). The result could not be more significant: with a p-value less than 10^{-10}, the null hypothesis $H_0 : \mu_{ILS} \leq \mu_{GV}$ can be rejected, where μ_{ILP} and μ_{GV} are the mean of the objective function (turnover or distance) by ILS and GRASP + VND, respectively. Therefore, it can be accepted that the results proposed by the GRASP + VND are statistically better than those provided by the ILS. We have performed these tests for the two sets of instances (Albareda and Henn) separately, and the results are similar, with p-values practically null in all cases.

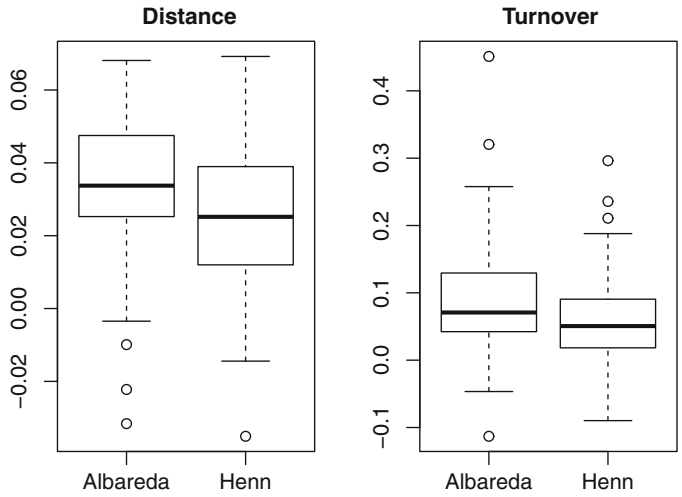


Fig. 9 Relative difference between objectives ILS and GRASP + VND

To conclude, we have analyzed the relative improvement of the GRASP + VND versus ILS. Figure 9 shows the distribution of the ratio $\frac{z_{ILP}-z_{GV}}{z_{GV}}$ for the two objectives and the two sets of instances, Albareda and Henn. The two boxplots on the right show the results for the turnover and the two on the left for the distance. For each of the objectives, the relative improvement is shown separately for Albareda and Henn instances. Note that negative values imply that ILS provides better results than GRASP + VND. From these boxplots, we can conclude again that in most cases GRASP + VND provides better solutions; in those few cases where the ILS solution is better, it does not exceed 2% improvement compared to almost 7% that GRASP + VND can improve the solution. It is interesting to note that GRASP + VND achieves lower improvements in Henn instances, which are precisely where the ILS was tested.

In order to facilitate future comparisons, we report the best values found in our experiments, for each of the considered instances. Particularly, in Table 13 in Appendix A, we report the results for the dataset introduced in [1], and the results for the data set introduced in [18] in Table 14 in Appendix B. All this information, together with the set of instances used in our experiments are available at http://grafo.etsii.urjc.es/optsicom/oobp/.

5 Conclusions

In this paper, we have tackled the online order batching problem with a single picker and in a single-block warehouse. This problem is a variant of the well-known family of problems commonly referred to as Order Batching Problem. It is based on the order batching picking strategy as an efficient way of retrieving the orders arrived in a warehouse, i.e., orders are grouped into batches before they are collected. Additionally, every batch is collected in a single route of a picker.

The OOBP presents the difficulty that all the orders that must be retrieved are not available at the beginning of the process, but they arrive to the warehouse at any time. For the sake of simplicity, we have considered a time horizon of 4 h for the arrival of orders, however, the picker might need longer times to collect all the orders arrived in that time horizon.

To achieve our goals we have proposed several heuristic algorithms which are combined within a GRASP and VND metaheuristics. Particularly, we propose a GRASP algorithm as a general framework to tackle the problem, and we use the VND metaheuristic as a local search procedure. The constructive phase of the GRASP is based on the weight of the orders as a greedy criterion (i.e., the heaviest, the first). The α parameter of GRASP has been

set to a random value on each iteration. The VND method proposed includes three different neighborhood structures that have been empirically sorted. Finally, we have used the S-Shape algorithm as the routing strategy to calculate the best route for collecting the orders grouped in each batch.

It is worth mentioning that in this paper we have considered two different objective functions: (i) minimize the maximum completion time; and (ii) minimize the maximum turnover time. The former, minimizes how long it takes to collect all the orders arrived to the system. This objective function is widely used in many papers. The latter minimizes the maximum time that an order remains in the system. As far as we understand the problem, this objective function represents the most realistic scenario for this variant of the OBP. However, we have reported and compared with previous proposals in the state of the art, the values obtained for both objective functions. The proposed algorithms improve the previous results in the state of the art in both cases, finding improvements of more than 3% and 6%, on average, for each objective function, respectively. The statistical tests performed corroborated in both cases that there are significant differences among the results found.

Our findings indicated that the running time for this problem is not one of the biggest issues, since the time horizon on a real scenario is very large. There are also many small factors that must be taken in consideration, such as the departure time of the picker, once the batches are ready to be collected. In a first approach, the sooner the better, however, awaiting might result in a final improvement, since there are more chances for the arrival of new orders to the system, that might be included in the next batch. Also, we found that the use of several neighborhoods is a key strategy, since it is not always easy to perform moves within the available space in the batches. We have detected that sorting the orders in a descending weight might help to obtain batches with less wasted space. Empirical results indicate that the fewer number of batches, the better.

It is important to highlight that when considering the total time needed to collect all the orders as objective function, the time of collecting a particular batch does not change, no matters the moment in the time when the route to collect this batch starts. However, when considering the turnover time as objective function, not only the batch configuration is important, but also the moment in the time when the picking route starts for each batch. This is an additional difficulty of this variant of the problem. In this sense, it is necessary to set a moment in the time as the hypothetical starting point, taken as a reference, to compare two different solutions.

As a final observation, the online version of the OBP makes possible many situations where a short number of orders are handled at the same time. Therefore, it could be an interesting opportunity for designing matheuristic algorithms in the future.

Appendix A: Results per instance (instance set Albareda [1])

Table 13 Results per instance for the dataset introduced in [1] over the two considered objective functions

Instance	Objective function		Instance	Objective function	
	T (s)	Turnover (s)		T (s)	Turnover (s)
W1_100_000	22, 308	10, 864	W3_100_000	39, 902	30, 959
W1_100_030	18, 371	6394	W3_100_030	33, 212	25, 539
W1_100_060	21, 676	9911	W3_100_060	41, 067	33, 276
W1_100_090	18, 313	6157	W3_100_090	33, 974	24, 111
W1_150_000	28, 612	18, 449	W3_150_000	54, 189	45, 495
W1_150_030	24, 386	13, 556	W3_150_030	46, 013	38, 616
W1_150_060	30, 770	18, 812	W3_150_060	58, 797	50, 833
W1_150_090	24, 101	12, 129	W3_150_090	46, 210	38, 277
W1_200_000	42, 091	29, 194	W3_200_000	72, 452	63, 494
W1_200_030	30, 810	18, 461	W3_200_030	60, 713	54, 808
W1_200_060	38, 485	27, 209	W3_200_060	79, 633	72, 394
W1_200_090	31, 405	20, 774	W3_200_090	57, 516	52, 072
W1_250_000	53, 101	40, 683	W3_250_000	93, 604	84, 912
W1_250_030	38, 490	26, 414	W3_250_030	74, 203	68, 954
W1_250_060	51, 255	40, 891	W3_250_060	99, 265	91, 559
W1_250_090	40, 606	30, 471	W3_250_090	76, 361	71, 208
W2_100_000	17, 203	6153	W4_100_000	110, 218	96, 620
W2_100_030	15, 535	2702	W4_100_030	92, 936	79, 951
W2_100_060	17, 131	4849	W4_100_060	94, 370	82, 276
W2_100_090	15, 069	2849	W4_100_090	77, 919	64, 560
W2_150_000	24, 227	13, 214	W4_150_000	155, 919	14, 1927
W2_150_030	21, 052	9595	W4_150_030	118, 893	105, 478
W2_150_060	24, 375	13, 561	W4_150_060	155, 998	141, 955
W2_150_090	21, 183	10, 110	W4_150_090	119, 539	106, 666
W2_200_000	33, 170	22, 498	W4_200_000	198, 530	186, 439
W2_200_030	26, 341	15, 274	W4_200_030	150, 094	136, 789
W2_200_060	31, 020	21, 247	W4_200_060	202, 348	188, 713
W2_200_090	26, 429	15, 433	W4_200_090	169, 074	154, 963
W2_250_000	38, 100	28, 017	W4_250_000	249, 690	236, 033
W2_250_030	33, 341	21, 956	W4_250_030	181, 508	168, 829
W2_250_060	41, 148	30, 372	W4_250_060	249, 863	235, 831
W2_250_090	34, 352	23, 382	W4_250_090	199, 738	185, 948

Appendix B: Results per instance (instance set Henn [18])

Table 14 Results per instance for the dataset introduced in [18] over the two considered objective functions

Instance	Objective function		Instance	Objective function	
	T (s)	Turnover (s)		T (s)	Turnover (s)
W5_abc1_40_29	21, 109	9848	W5_ran1_40_29	24, 689	12, 838
W5_abc1_40_30	17, 541	5133	W5_ran1_40_30	18, 023	8715
W5_abc1_40_31	17, 831	4788	W5_ran1_40_31	18, 815	6364
W5_abc1_40_32	16, 226	4488	W5_ran1_40_32	16, 393	5982
W5_abc1_60_37	31, 794	19, 729	W5_ran1_60_37	36, 681	25, 558
W5_abc1_60_38	23, 366	10, 978	W5_ran1_60_38	26, 418	14, 984
W5_abc1_60_39	21, 061	10, 985	W5_ran1_60_39	23, 674	12, 124
W5_abc1_60_40	18, 196	8538	W5_ran1_60_40	20, 759	9480
W5_abc1_80_61	40, 745	27, 372	W5_ran1_80_61	47, 254	34, 077
W5_abc1_80_62	32, 385	20, 458	W5_ran1_80_62	37, 688	24, 958
W5_abc1_80_63	26, 083	14, 713	W5_ran1_80_63	29, 133	17, 544
W5_abc1_80_64	24, 189	12, 530	W5_ran1_80_64	27, 565	15, 419
W5_abc1_100_69	45, 613	32, 404	W5_ran1_100_69	53, 976	41, 064
W5_abc1_100_70	33, 888	21, 849	W5_ran1_100_70	39, 367	27, 710
W5_abc1_100_71	30, 081	20, 595	W5_ran1_100_71	34, 027	23, 168
W5_abc1_100_72	28, 543	17, 174	W5_ran1_100_72	31, 446	19, 298
W5_abc2_40_9	21, 729	9279	W5_ran2_40_9	24, 660	14, 619
W5_abc2_40_10	18, 170	9377	W5_ran2_40_10	21, 073	8990
W5_abc2_40_11	15, 708	4829	W5_ran2_40_11	16, 051	5874
W5_abc2_40_12	15, 129	4177	W5_ran2_40_12	16, 623	5166
W5_abc2_60_17	29, 225	17, 812	W5_ran2_60_17	34, 631	22, 110
W5_abc2_60_18	25, 303	12, 760	W5_ran2_60_18	27, 725	17, 760
W5_abc2_60_19	20, 510	8120	W5_ran2_60_19	23, 565	13, 142
W5_abc2_60_20	18, 304	8467	W5_ran2_60_20	19, 750	8963
W5_abc2_80_45	38, 013	24, 949	W5_ran2_80_45	44, 605	31, 155
W5_abc2_80_46	28, 691	16, 423	W5_ran2_80_46	32, 758	21, 503
W5_abc2_80_47	25, 956	14, 817	W5_ran2_80_47	30, 911	18, 730
W5_abc2_80_48	23, 524	13, 018	W5_ran2_80_48	26, 983	16, 964
W5_abc2_100_53	49, 363	36, 092	W5_ran2_100_53	57, 097	43, 018
W5_abc2_100_54	35, 046	24, 269	W5_ran2_100_54	40, 093	27, 849
W5_abc2_100_55	34, 670	23, 567	W5_ran2_100_55	39, 278	28, 689
W5_abc2_100_56	28, 941	18, 828	W5_ran2_100_56	32, 025	20, 591

References

1. Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., De Blas, C.S.: Variable neighborhood search for order batching in a warehouse. Asia-Pac. J. Oper. Res. **26**(5), 655–683 (2009)
2. Chen, F., Wei, Y., Wang, H.: A heuristic based batching and assigning method for online customer orders. Flex. Serv. Manuf. J. **30**(4), 640–685 (2018)
3. Coyle, J.J., Bardi, E.J., Langley, C.J., et al.: The Management of Business Logistics, vol. 6. West Publishing Company Minneapolis, St Paul (1996)
4. De Koster, M.B.M., Van der Poort, E.S., Wolters, M.: Efficient orderbatching methods in warehouses. Int. J. Prod. Res. **37**(7), 1479–1504 (1999)
5. De Koster, R., Le-Duc, T., Roodbergen, K.J.: Design and control of warehouse order picking: a literature review. Eur. J. Oper. Res. **182**, 481–501 (2007)
6. De Koster, R., Jan Roodbergen, K., van Voorden, R.: Reduction of walking time in the distribution center of de bijenkorf. In: New trends in distribution logistics, pp. 215–234. Springer, Berlin (1999)
7. Drury, J., Warehouse Operations Special Interest Group, Order Picking Working Party, Turnbull, B., Institute of Logistics (Great Britain): Towards More Efficient Order Picking. IMM monograph. Institute of Materials Management. ISBN 9781870214063 (1988). https://books.google.es/books?id=LbTKAAAACAAJ
8. Duarte, A., Pantrigo, J.J., Pardo, E.G., Mladenovic, N.: Multi-objective variable neighborhood search: an application to combinatorial optimization problems. J. Global Optim. **63**(3), 515–536 (2015)
9. Duarte, A., Pantrigo, J.J., Pardo, E.G., Sánchez-Oro, J.: Parallel variable neighbourhood search strategies for the cutwidth minimization problem. IMA J. Manag. Math. **27**(1), 55–73 (2013)
10. Feo, T.A., Resende, M.: Greedy randomized adaptive search procedures. J. Global Optim. **6**, 109–133 (1995)
11. Gademann, A.J.R.M., Van Den Berg, J.P., Van Der Hoff, H.H.: An order batching algorithm for wave picking in a parallel-aisle warehouse. IIE Trans. **33**(5), 385–398 (2001)
12. Gademann, N., Velde, S.: Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Trans. **37**(1), 63–75 (2005)
13. Gibson, D.R., Sharp, G.P.: Order batching procedures. Eur. J. Oper. Res. **58**(1), 57–67 (1992)
14. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse design and performance evaluation: a comprehensive review. Eur. J. Oper. Res. **203**(3), 539–549 (2010)
15. Hall, R.W.: Distance approximations for routing manual pickers in a warehouse. IIE Trans. **25**(4), 76–87 (1993)
16. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
17. Hansen, P., Mladenović, N., Moreno-Pérez, J.A.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010)
18. Henn, S.: Algorithms for on-line order batching in an order picking warehouse. Comput. Oper. Res. **39**(11), 2549–2563 (2012)
19. Henn, S., Koch, S., Doerner, K., Strauss, C., Wäscher, G.: Metaheuristics for the order batching problem in manual order picking systems. BuR Bus. Res. J. **3**(1), 82–105 (2010)
20. Henn, S., Schmid, V.: Metaheuristics for order batching and sequencing in manual order picking systems. Comput. Ind. Eng. **66**(2), 338–351 (2013)
21. Henn, S., Wäscher, G.: Tabu search heuristics for the order batching problem in manual order picking systems. Eur. J. Oper. Res. **222**(3), 484–494 (2012)
22. Ho, Y.-C., Tseng, Y.-Y.: A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. Int. J. Prod. Res. **44**(17), 3391–3417 (2006)
23. Hsu, C.-M., Chen, K.-Y., Chen, M.-C.: Batching orders in warehouses by minimizing travel distance with genetic algorithms. Comput. Ind. **56**(2), 169–178 (2005)
24. Koch, S., Wäscher, G.: A grouping genetic algorithm for the Order Batching Problem in distribution warehouses. J. Bus. Econ. **86**(1–2), 131–153 (2016)
25. Menéndez, B., Bustillo, M., Pardo, E.G., Duarte, A.: General variable neighborhood search for the order batching and sequencing problem. Eur. J. Oper. Res. **263**(1), 82–93 (2017)
26. Menéndez, B., Pardo, E.G., Alonso-Ayuso, A., Molina, E., Duarte, A.: Variable neighborhood search strategies for the order batching problem. Comput. Oper. Res. **78**, 500–512 (2017)
27. Menéndez, B., Pardo, E.G., Duarte, A., Alonso-Ayuso, A., Molina, E.: General variable neighborhood search applied to the picking process in a warehouse. Electron. Notes Discrete Math. **47**, 77–84 (2015)
28. Menéndez, B., Pardo, E.G., Sánchez-Oro, J., Duarte, A.: Parallel variable neighborhood search for the min–max order batching problem. Int. Trans. Oper. Res, **24**(3), 635–662 (2017)
29. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)

30. Öncan, T.: MILP formulations and an iterated local search algorithm with Tabu thresholding for the order batching problem. J. Oper. Res. **243**, 142–155 (2015)
31. Öncan, T., Cağırıcı, M.: MILP formulations for the order batching problem in low-level picker-to-part warehouse systems. IFAC Proc. Vol. **46**(9), 471–476 (2013)
32. Pan, C.-H., Liu, S.-Y.: A comparative study of order batching algorithms. Omega **23**(6), 691–700 (1995)
33. Pardo, E.G., Mladenović, N., Pantrigo, J.J., Duarte, A.: Variable formulation search for the cutwidth minimization problem. Appl. Soft Comput. **13**(5), 2242–2252 (2013)
34. Pérez-Rodríguez, R., Hernández-Aguirre, A., Jöns, S.: A continuous estimation of distribution algorithm for the online order-batching problem. Int. J. Adv. Manuf. Technol. **79**(1), 569–588 (2015)
35. Petersen, C.G.: An evaluation of order picking routeing policies. Int. J. Oper. Prod. Manag. **17**(11), 1098–1111 (1997)
36. Petersen, C.G.: Routeing and storage policy interaction in order picking operations. Decis. Sci. Inst. Proc. **31**(3), 1614–1616 (1995)
37. Ratliff, H.D., Rosenthal, A.S.: Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper. Res. **31**(3), 507–521 (1983)
38. Roodbergen, K.J., Koster, R.D.: Routing methods for warehouses with multiple cross aisles. Int. J. Prod. Res. **39**(9), 1865–1883 (2001)
39. Roodbergen, K.J., Petersen, C.G.: How to improve order picking efficiency with routing and storage policies. In: G.R. Forger et al. (eds.), Progress in Material Handling Practice. Material Handling Institute, Charlotte, North Carolina, pp. 107–124 (1999)
40. Rosenwein, M.B.: A comparison of heuristics for the problem of batching orders for warehouse selection. Int. J. Prod. Res. **34**(3), 657–664 (1996)
41. Rubrico, J.I.U., Higashi, T., Tamura, H., Ota, J.: Online rescheduling of multiple picking agents for warehouse management. Robot. Comput. Integr. Manuf. **27**(1), 62–71 (2011)
42. Scholz, A., Schubert, D., Wäscher, G.: Order picking with multiple pickers and due dates—simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. Eur. J. Oper. Res. **263**(2), 461–478 (2017)
43. Scholz, A., Wäscher, G.: Order Batching and Picker Routing in manual order picking systems: the benefits of integrated routing. CEJOR **25**(2), 491–520 (2017)
44. Tang, L.C., Chew, E.P.: Order picking systems: batching and storage assignment strategies. Comput. Ind. Eng. **33**(3), 817–820 (1997). Selected Papers from the Proceedings of 1996 ICC&IC
45. Van Nieuwenhuyse, I., Colpaert, J.: Order batching in multi-server pick-and-sort warehouses. DTEW-KBI **0731**, 1–29 (2007)
46. Zhang, J., Wang, X., Chan, F.T.S., Ruan, J.: On-line order batching and sequencing problem with multiple pickers: a hybrid rule-based algorithm. Appl. Math. Model. **45**, 271–284 (2017)
47. Zhang, J., Wang, X., Huang, K.: Integrated on-line scheduling of order batching and delivery under b2c e-commerce. Comput. Ind. Eng. **94**, 280–289 (2016)
48. Zhang, J., Wang, X., Huang, K.: On-line scheduling of order picking and delivery with multiple zones and limited vehicle capacity. Omega **79**, 104–115 (2018)
49. Žulj, I., Kramer, S., Schneider, M.: A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. Eur. J. Oper. Res. **264**(2), 653–664 (2018)

Affiliations

Sergio Gil-Borrás[1] · Eduardo G. Pardo[2] · Antonio Alonso-Ayuso[2] · Abraham Duarte[2]

Sergio Gil-Borrás
sergio.gil@upm.es

Antonio Alonso-Ayuso
antonio.alonso@urjc.es

Abraham Duarte
abraham.duarte@urjc.es

[1] Universidad Politécnica de Madrid, C/Alan Turing s/n (Ctra. de Valencia, Km. 7), 28031 Madrid, Spain

[2] Universidad Rey Juan Carlos, C/Tulipán s/n, 28933 Móstoles, Madrid, Spain

Chapter 5

Online Order Batching Problem with Multiple Pickers

The Online Order Batching Problem with Multiple Pickers is a variant of the Order Batching Problem, where the arrival of products to the warehouse is dynamic and there is more than one picker in the warehouse. As the result of the research performed in this Doctoral Thesis, two articles have been published for this variant of the problem, which are presented next. Then, for each publication, we compile the bibliographic details of the publication (complete reference, journal, ranking index, category, and ranking score) and, finally, a copy of the published article is attached.

The article titled "*Basic VNS for a Variant of the Online Batching Problem*" [90] compiles the third paper within this Doctoral Thesis and includes the preliminary work performed to solve the Online Order Batching Problem with Multiple Pickers. The objective function studied in this paper was the minimization of the total time needed to collect all orders in an environment with two pickers. In this case, the arrival of orders occurs in a period of 4 hours. The problem was solved for a rectangular warehouse with a single block, 10 parallel aisles and a total of 900 stored products. The set of instances used for the problem was a set of 64 instances widely used in the literature [125]. As we have already seen in the previous sections, to solve any variant of the OBP, several associated subtasks have to be solved. Among them, when there are more than one picker in the warehouse, we need to assign each newly created batch to a picker to collect it. Additionally, in this case, we compared three algorithms for the routing task (S-Shape, Largest-Gap, and Combined). For the batching task we introduced a new algorithm, based on the Basic Variable Neighborhood Search (BVNS) methodology [197]. The proposal was compared to the seed algorithm introduced in the literature in [304]. The results obtained showed an improvement between 0.60% and 5% over the previous proposal in the literature, depending on the routing algorithm used. This article is attached in Section 5.1.

The article "*A heuristic approach for the online order batching problem with multiple pickers*" [93] was the fourth publication obtained as the result of the research carried out in this Doctoral Thesis. It can be considered as an evolution of the previous work, to solve the Online Order Batching Problem with multiple pickers. In this work, we studied three different objective functions for the problem: minimizing the completion time, minimizing the picking time, and minimizing differences in the workload among the pickers. The arrival of orders occurred in two different time periods: 2 and 4 hours, and the problem was solved for a rectangular warehouse of a single block and multiple pickers. We used two different sets of instances widely used in the literature [4, 125]. These sets of instances include five different warehouse configurations with a variant number of parallel aisles. In addition, we presented a mathematical model to define the problem addressed in its offline version. To tackle the batching task we proposed two versions of a multistart procedure hybridized with a Variable Neighborhood Descent (VND)

metaheuristic [197]. One version was devoted to reduce the picking time, and the other one to balance the workload of the pickers. The VND algorithm was designed using the same three neighborhoods used in the online version of the problem with only one picker [92]. The routing task was handled with an S-shape algorithm. Finally, the assignment task followed a simple rule, consisting of assigning the next batch to collect the picker with a lower workload in that moment. We compared the proposals to two previous approaches in the state of the art: an Iterated Local Search [5] and a Hybrid Rule-Based Algorithm hybridized with a seed algorithm [304]. We studied different scenarios varying the number of pickers, the congestion of the system, and the objective function. The results obtained indicated that the method using the workload balance as guiding function during the search is the best method to minimize either the completion time and workload balance in most scenarios presented. This article is attached in Section 5.2.

5.1 Basic VNS for a Variant of the Online Order Batching Problem

Gil-Borrás S., Pardo E.G., Alonso-Ayuso A., Duarte A. (2020) Basic VNS for a Variant of the Online Order Batching Problem. In: Benmansour R., Sifaleras A., Mladenović N. (eds) Variable Neighborhood Search. ICVNS 2019. Lecture Notes in Computer Science, vol 12010. pp 17-36 Springer, Cham.

Published in "**Lecture Notes in Computer Science**"

ISSN: 0302-9743 / 1611-3349

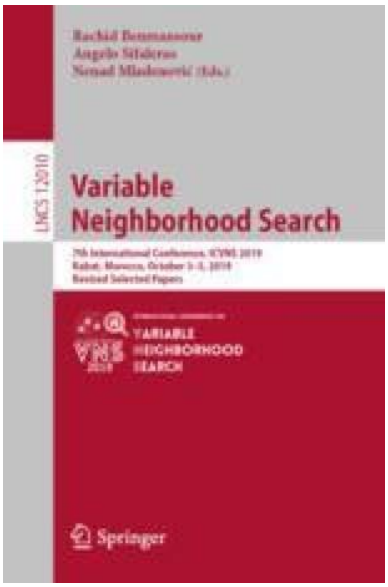
https://doi.org/10.1007/978-3-030-44932-2_2

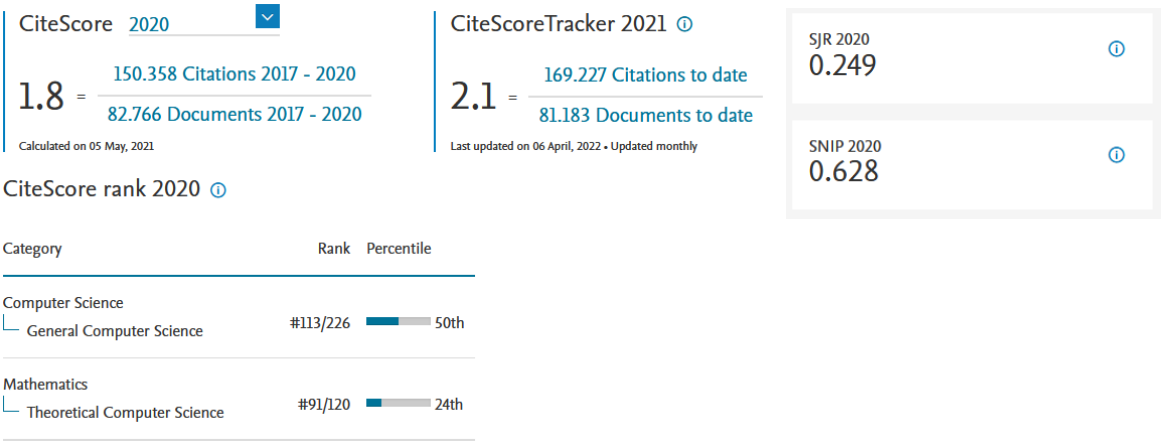
CiteScore 2020: 1.8

SJR 2020: 0.249

CiteScore rank 2020

- Computer Science, General Computer Science. Ranking 113/226 (Q3).
- Mathematics, Theoretical Computer Science. Ranking 91/120 (Q4).





Basic VNS for a Variant of the Online Order Batching Problem

Sergio Gil-Borrás[1(✉)], Eduardo G. Pardo[2(✉)], Antonio Alonso-Ayuso[2], and Abraham Duarte[2]

[1] Dept. Sistemas Informáticos, Universidad Politécnica de Madrid, Madrid, Spain
sergio.gil@upm.es
[2] Department of Computer Science, Universidad Rey Juan Carlos, Móstoles, Spain
{eduardo.pardo,antonio.alonso,abraham.duarte}@urjc.es

Abstract. The Online Order Batching Problem is a combinatorial optimization problem related to the process of retrieving items within a warehouse. It appears in the context of warehousing, when the warehouse follows an order-batching picking policy, which means that orders are packed together into batches before been collected. Additionally, since this problem is online, orders are arriving to the warehouse continuously, which is usually due to the fact that orders come from an e-commerce platform. The variant of the problem tacked in this paper also considers an additional characteristic: there are multiple pickers available to collect the batches. In this paper we propose several strategies, based on the Variable Neighborhood Search methodology, to tackle the problem and we compare them with the algorithms in the state of the art, using previously referred data sets. Additionally, we test the influence of different routing strategies not used before in the context of this variant.

Keywords: Online Order Batching Problem · Batching · Variable Neighborhood Search · Multiple pickers

1 Introduction

The e-commerce has suffered an explosion in last few years, thousands of products are sold online everyday and this is just the beginning. The increase in the online sales has made companies to development new processes related to their supply chain management, and also to improve/modify the old ones. However, the evolution in the supply chain models is not new, since it has been happening for many years, as it is possible to trace back in the associated

This research was partially funded by the projects: MTM2015-63710-P, RTI2018-094269-B-I00, TIN2015-65460-C2-2-P and PGC2018-095322-B-C22 from Ministerio de Ciencia, Innovación y Universidades (Spain); by Comunidad de Madrid and European Regional Development Fund, grant ref. P2018/TCS-4566; and by Programa Propio de I+D+i de la Universidad Politécnica de Madrid (Programa 466A).

R. Benmansour et al. (Eds.): ICVNS 2019, LNCS 12010, pp. 17–36, 2020.
https://doi.org/10.1007/978-3-030-44932-2_2

literature from the early eighties up to today. Particularly, in the last ten years, the number of papers related to the supply chain management has suffered a significant increase. In this sense, the online commerce has appeared as one of the next steps. In fact, many classical optimization problems have been reformulated taking into consideration online restrictions.

As part of the supply chain, we focus our attention in the activities that happen within a warehouse. More exactly, in the picking process of items. The global objective of the picking activity is mainly related to satisfy the demand of the customers as soon as possible. However, there are also other important issues in which Warehouse Management Systems (WMS) must pay attention to, such as: balance the workload of the workers in the warehouse, satisfy a predefined due date, save energy, or simply reduce the travel time of the pickers when collecting the items.

In this paper, we tackle the Online Order Batching Problem with Multiple Pickers (OOBPMP). In this optimization problem, orders are arriving to the warehouse 24 h a day/7 days a week, so it means that instances of the problem are changing dynamically. The objective of this optimization problem is either minimizing the time used to collect all the items in the orders received, or minimizing the maximum turnover time of any order (i.e., the time that an order remains in the system). In this problem, the picking strategy is based the concept of batch, which stands for a group of orders that are packed together, before start collecting them. Then, all the items in the same batch are collected in a single route. Notice that the orders can not be split into more than one batch. Also, the batches can not exceed a predefined maximum capacity (weight and/or volume restriction). Every batch can be assigned only to one picker, and every picker can not simultaneously collect items from more than one batch. In this sense, the picking strategy falls into the picker-to-part category. Additionally, the OOBPMP takes into consideration the existence of multiple pickers in the warehouse. In this paper, we propose several strategies to construct the batches, to set the priority in which the batches are assigned to the pickers, and to determine a route to collect the items in the same batch. We do not study here the impact of the storage policy, nor the influence of the different distributions of arrival time moments of the orders.

There are different and well-known routing policies for the picker in the literature related to warehousing, which are suitable for this problem. These strategies range from exact to heuristic methods. The performance of each method partially depends on the shape and structure of the warehouse. The exact methods, further than the longer times needed to calculate a route for the problem, are frequently excluded from real scenarios, because many times they create routes difficult to understand and follow by the operators. This difficulty increases as the complexity of the warehouse grows. On the other hand, simple routing heuristics are usually fast to calculate and they produce reasonable good results with routes easy to understand for the pickers.

The rest of the paper is organized as follows: in Sect. 2, we present the state of the art of the Order Batching family of problems and we focus our attention in the OOBPMP. We also review here the most outstanding heuristic routing procedures in the literature. In Sect. 3, we present a new algorithm for tackling the batching task of the considered problem. Section 4 compiles the computational results obtained with the proposed algorithms over some referenced data sets. Finally, our conclusions and future research lines are exposed in Sect. 5.

2 State of the Art

The Order Batching Problem, further than a single optimization problem, can be considered as a family of optimization problems which groups together those problems related to the retrieval of goods from a warehouse, using a picking policy based on the order batching strategy.

However, within this family of problems, the simplest and most classical version is also referred to in the literature as Order Batching Problem (OBP) [81]. The OBP consists in minimizing the total time needed to collect a group of orders received in a warehouse in a context with a single picker, and having all the orders considered at hand, before starting the batching process. This version of the problem can be considered as static and it has raised a relevant interest in the scientific community. Theoretical studies about the OBP indicated that the problem is $\mathcal{NP}$-hard for general instances [23], but solvable in polynomial time if each batch does not contain more than two orders [23]. However, most of the real instances does not usually fulfill the previous requirement. Due to its hardness, but also to the necessity of finding solutions to the problem in short amounts of time, heuristics and metaheuristics have been applied to tackle the problem. The First-Come First-Served (FCFS) strategy was one of the first heuristic strategies proposed and used in practice to assign orders to batches in a warehouse. This strategy has been widely used due to its simplicity. Other important heuristic methods are the *seed methods* [25,38,62] and the *saving methods* [76]. In [13] it is possible to find a survey of those methods where the authors proposed a classification. The first metaheuristic algorithm applied to the simple OBP was based in a Genetic Algorithm and it was proposed in [42]. Later, a method based on the Variable Neighborhood Search methodology was presented in [1]; an Iterated Local Search in [36] and a Tabu Search in [34]. In [59], the authors proposed an new Iterated Local Search algorithm with a Tabu Thresholding. The current state of the art for the problem, as far as we know, was a multi-start Variable Neighborhood Search method proposed in [53].

Despite of the fact that the simplest OBP has received the largest attention, other static variants have also been studied in the literature: the Order Batching and Sequencing Problem (OBSP) is a variant of the OBP which introduce due dates in the orders [33,52]; and the Min-Max Order Batching Problem (Min-Max OBP) looks for a work balance among several operators in a warehouse [22,55].

As far as the online variants are concerned (i.e., those which receive orders continuously in the system) the first approach found was presented in [81], where the authors proposed a simple FCFS algorithm and considered a variant of the OOBP with multiple pickers. Later, in [86] the OOBP was studied for multiple-block warehouses. In [77] the Online Rescheduling Problem with multiple pickers was tackled by using a Steepest Descent Insertion strategy, and a Multistage Rescheduling strategy. In [27,32,66] a single-block warehouse with a single picker version was tackled in the online context. In the first case the authors proposed an Iterated Local Search and, in the second case, they proposed an Estimation of Distribution Algorithm (EDA). In [94,95] the authors added a new constraint to the problem, related to the scheduling of the delivery. The first work considered only one picker, meanwhile the second one considered multiple pickers. The most recent approach within this context was presented in [10], where the OOBP was studied for multiple blocks and multiple pickers.

We have summarized all the aforementioned methods in Table 1, where the papers are classified depending on the variant of the problem considered. Particularly, we have divided the works into two columns: offline (static) and online (dynamic). For each column we have separated those works which consider only one picker from those which consider multiple pickers. Also we have classified the papers depending on the inclusion or not of due dates in the orders.

Table 1. Publications related with the Order Batching Problem, classified according to the variant of the problem tackled.

		Online	Offline
One picker	With due date	[19]	[4,8,36,43,44,52,82,99]
	Without due date	[11,26,32,45,48,66,69,72,81,86,87,91,95,96]	[1,2,5,6,9,13,20,23–25,34,38–40,42,46,49–51,53,54,56,57,59–63,65,67,68,73,76,78,83–85,88–90,92,93,97,98]
Multiple pickers	With due date	–	[37,41,79,80]
	Without due date	[10,21,77,94]	[3,7,22,35,55]

In this paper we focus our attention in the online version of the OBP which considers multiple pickers and do not include due dates, previously referred to as OOBPMP. Next, in Sect. 2.1 we review the latest batching strategy published for the problem in [94]. Finally, in Sect. 2.2, we review some the most outstanding routing strategies in the context of the OBP.

2.1 Batching State-of-the-Art Algorithm for the OOBPMP

As far as we know, the latest batching algorithm proposed for the OOBPMP was introduced in [94]. The authors used the well-known "seed" strategy [38] for clustering, in order to perform the batching task of the problem. This clustering strategy consists in selecting a "seed" (in this case the seed is represented by an order) as a centroid of a cluster (in this case the cluster is represented by a batch). Then, the seed is assigned to an empty cluster (i.e., the order is assigned to an empty batch) and other available orders might be added to the same batch, depending on the similarity with respect the selected seed order, while the capacity of the batch is not exceeded.

Therefore, for each "seed method" it is necessary to decide how to choose the seed order, and how to determine the similarity of the orders with respect to the seed. In this case, the strategy used to select an order as a "seed" is based on the Smallest Arrival Time rule (i.e., the order which arrived first to the system and has not been assigned yet to any batch is selected as a seed). Once the seed order has been chosen, it is assigned to an empty batch. Then, the strategy used to aggregate other orders to the same batch follows an Aisle-Time-Based strategy. This strategy takes into consideration two dimensions: the percentage of orders that the seed order has in common with the candidate order; and also, since we are in an online context, it includes a measure related to the arrival time of the considered order. This similarity measure is calculated for every order whose device-capacity demand do not exceed the remaining capacity of the picking cart, and then the next order be added to the current batch is selected in a greedy way. Once the batch is full (i.e., no other order among the available ones can be added) the method selects a new seed and so on, until all the orders have been assigned to a batch. We invite the reader to carefully review this method in [94].

2.2 Routing Algorithms

The routing algorithm is responsible for generating a path to collect every item in the orders of the batch, following a single route. As it was aforementioned, the order picking operations are one of the most important and costly processes in a warehouse [12,16]. In this case, when the picking policy is based on batches, considering the batching and picking tasks together might suppose a reduction up to 35% in the total travel time [14].

The problem of finding a route within the warehouse, where the picker must visit a group of positions, is a simplified version of the Travelling Salesman Problem (TSP) [15] and therefore, there are many different proposals available in the literature to solve it. Particularly, in this case, specific algorithms have been developed considering the rectangular structure of the warehouse used in this paper, which defines a special metric space, whose properties can be used in the design of the route. In fact, there are an exact method [74], based on dynamic programming, which generates the optimal path within this context. However, further than the extra time needed to compute the route, the exact

method generates complicated paths for the pickers, which make them to have difficulties to remember and interpret the generated routes [23]. In this sense, other simpler methods, like heuristic ones, are commonly used in practice to solve the problem. The heuristic procedures are usually very fast to compute and they generate simple paths, which are easy to follow for the pickers. Many routing heuristics have been proposed in the literature (see [28,70,71,75] for several proposals and comparisons). In this paper we review the most important and used heuristic procedures in the literature: S-Shape, Largest Gap and Combined.

S-Shape. The S-shape algorithm is one of the most used routing algorithms in the literature mainly due to its simplicity. It is not only easy to implement but also it generates simple routes for the operators. The method constructs a route, starting from the depot, which begins traversing the leftmost aisle which contains at least one item to collect. Then it goes through all the aisles that have items to pick up from any order in the batch. Therefore the picker is performing changes from the front-cross aisle to the back-cross aisle and the other way round. If the number of aisles to be traversed is even, the last parallel aisle will be completely covered (i.e., the picker will finish the route in the front-cross aisle). However, if the number of aisles to traverse is odd, the last corridor is only covered until the last element to be collected and then, the picker performs a U-turn, in order to come back to the front-cross aisle (which contains the depot). An example of a route following this strategy is depicted in Fig. 1. Notice, that in this example there are 5 aisles which contain items to collect.

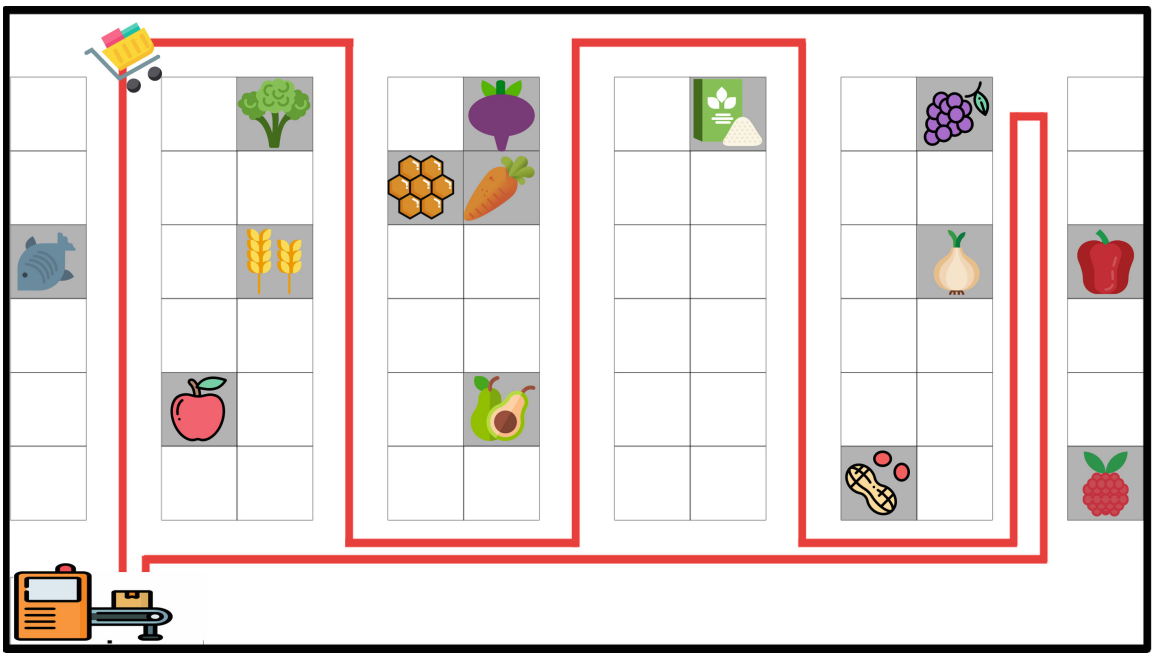


Fig. 1. Path created with S-Shape method.

Largest Gap. The Largest gap algorithm, along with the S-shape, is one of the routing algorithms widely used in the literature. To understand the largest gap, we first define the concept of gap as the space in an aisle between every two positions which contain an item to collect. Additionally, the space between the front cross-aisle and the first position which contains an item to collect and, the space between the last position which contains an item to collect and the back-cross aisle are also considered gaps. For each aisle that has items to pick up, the largest gap in an aisle is the longest distance among all the possible gaps in the aisle. Then, the Largest gap strategy avoids traversing the largest gap of each aisle by performing an U-turn each time a picker arises the position where the largest gap starts/ends. This algorithm also starts exploring the first aisle to the left which contains items to collect. This aisle will be fully traversed, in order to start collecting from the back-cross aisle. Similarly, the last parallel aisle with items to collect will be also fully traversed in order to come back to the front-cross aisle. An example of a route following this strategy is depicted in Fig. 2.

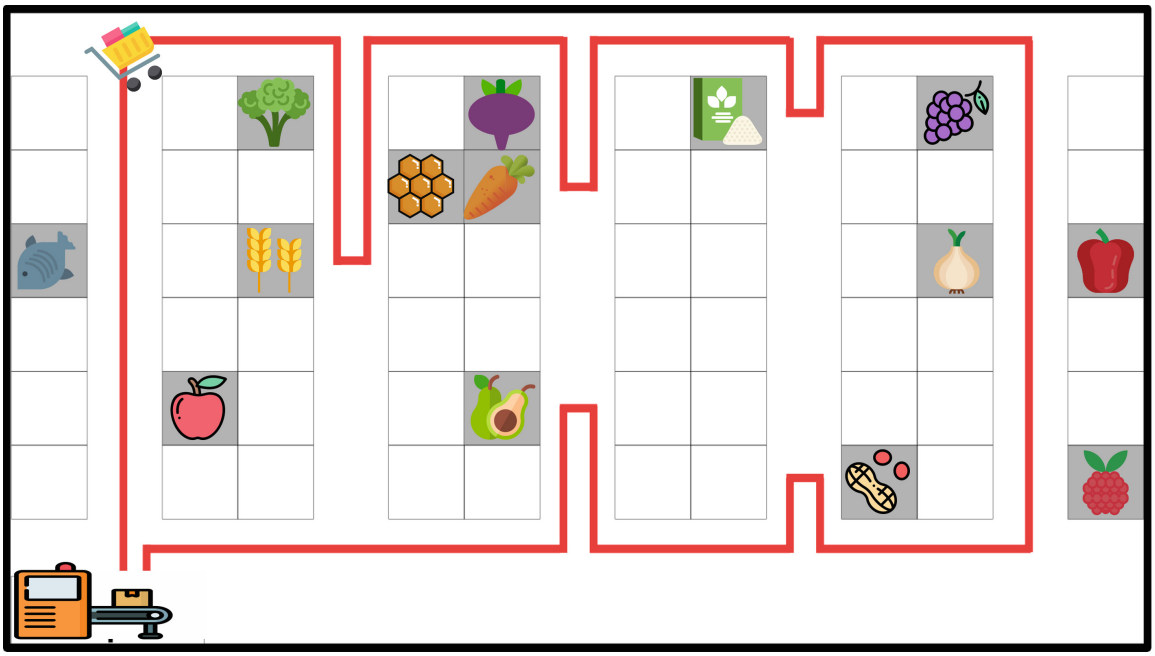


Fig. 2. Path created with Largest Gap method.

Combined. The Combined algorithm was first proposed in [47]. The idea was to combine the two previously introduced methods (S-shape and Largest gap) in order to make a more efficient method. In this case, the algorithm decides, for each parallel aisle, if it is shorter to collect the items in that aisle using an S-shape strategy or a Largest gap one. Then, the algorithm selects the most convenient way. The method has to consider that the number of parallel aisles traversed with S-shape must be even. In some occasions, the circumstances may

force the algorithm to change the previously selected strategy for a particular aisle, in order to end the route in the front-cross aisle. An example of a route following this strategy is depicted in Fig. 3.

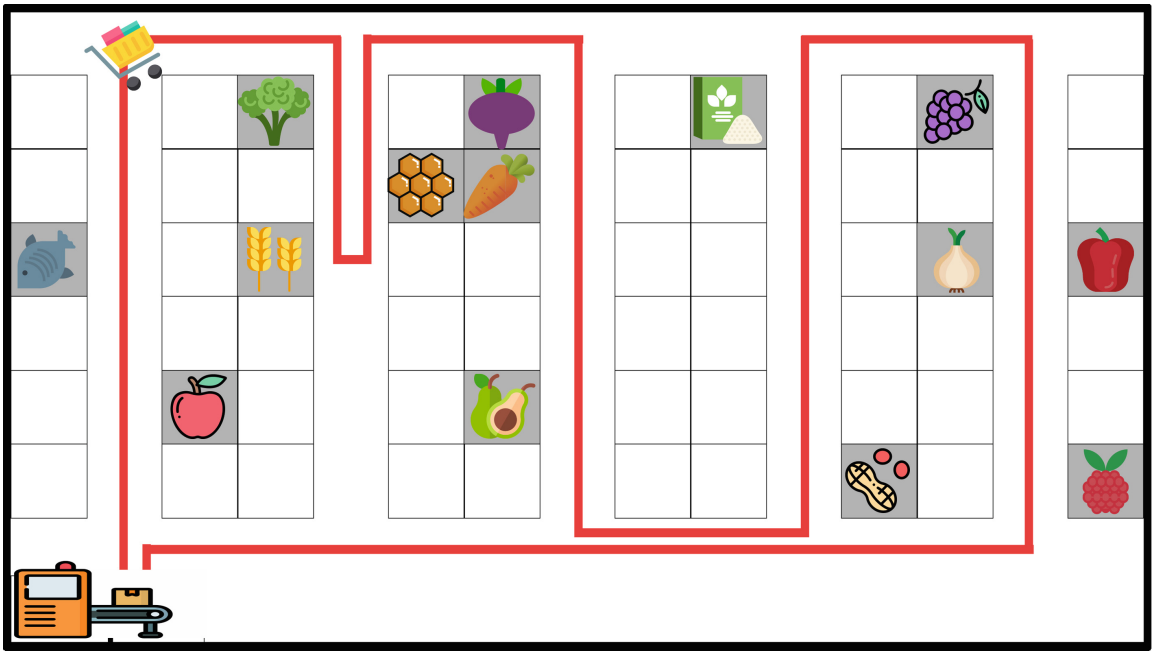


Fig. 3. Path created with Combined method.

3 Algorithmic Proposal

In this section we present our algorithmic proposal to tackle the OOBPMP. In particular, we propose the use of the Basic Variable Neighborhood Search (BVNS) schema [31,58]. BVNS is a variant of the VNS methodology which was proposed in [58] as a general method to solve hard optimization problems. It is based on the concept of change of the neighborhood structure in order to escape from local minima. There are many different variants of VNS, however the best known ones are: Reduced VNS (RVNS), Basic VNS (BVNS), Variable Neighborhood Descent (VND), General VNS (GVNS), Skewed VNS (SVNS), and Variable Neighborhood Decomposition Search (VNDS). Those variants differs in the use of stochastic/deterministic explorations or a mix of both (as it is the case of BVNS) of the neighborhoods considered. We refer the reader to [29,30,58] for a deep understanding. Some other interesting variants of the VNS methodology have been recently proposed. Among others, we can find: Variable Formulation Search (VFS) [64], Multi-Objective Variable Neighborhood Search [17], and Parallel Variable Neighborhood Search [18,55].

In Algorithm 1 we present a pseudocode of the BVNS method proposed in this paper. The method receives three input parameters: (i) an initial solution S; (ii) a value k_{max}; and (iii) the maximum time (t_{max}). The initial solution will be calculated using an external method that will be described in Sect. 3.1.

On the other hand k_{max} determines the maximum number of neighborhoods that will be explored. Particularly, the method explores the current neighborhood of the solution trying to obtain a better solution. BVNS includes three different stages to explore the current neighborhood and it determines if there has been an improvement. First, the method performs a perturbation to the current solution, in order to escape from the current local optimum. Second, the method runs an improvement procedure based on a local search, which is able to find a local optimum in the current neighborhood. Third, the procedure `Neighborhoodchange`, determines if there has been any improvement in the solution. If so, the next neighborhood to explore will be the first one. Otherwise, the value of the variable k is increased and, therefore, in the next iteration the number of perturbations performed to the current solution in the `Shake` procedure is increased. The method stops when the value of k equals k_{max}, and the maximum allowed time is reached.

Algorithm 1. `BVNS`$(S, k_{\max}, t_{\max})$

```
repeat
    k ← 1
    while k ≤ k_max do
        S' ← Shake(S, k)
        S'' ← LocalSearch(S')
        k ← NeighborhoodChange(S, S'', k)
    end while
until t < t_max
return S
```

The description of the `Shake` and `LocalSearch` procedures are presented, in Sects. 3.2 and 3.3 respectively. The `NeighborhoodChange` procedure follows an standard implementation which can be reviewed in [58].

3.1 Constructive Procedure

We have used a random algorithm as a constructive method in order to provide an initial solution to the BVNS algorithm. The constructive algorithm receives a list of orders L_{orders} as an input parameter. In each iteration, an order is randomly selected from the list and it is placed in the next available batch. When the selected order does not fit in the current batch, a new batch is created with this order. Then, the next order will be placed in this new batch and the process is repeated until the order list is fully scanned and all the orders have a batch assigned. Once the process is finished, the procedure returns a list of batches S as a solution. In Algorithm 2 we present a pseudocode of this procedure.

Algorithm 2. `Constructive`(L_{orders})

```
1: S ← NewBatchList()
2: B ← NewBatch()
3: repeat
4:     o ← ChooseRandomOrder(L_orders)
5:     L_orders ← L_orders \ o
6:     if Fits(B, o) then
7:         Add(B, o)
8:     else
9:         Add(S, B)
10:        B ← NewBatch()
11:        Add(B, o)
12:    end if
13: until L_orders = ∅
14: return S
```

3.2 Shake Procedure

The shake procedure is in charge of performing a perturbation to the current solution. The method starts by selecting two random batches. Then, it selects two random orders (one from each batch) and finally, the method tries to perform an exchange move. The move is not done if the size of any of the selected batches is exceeded. This process is repeated as many times as indicates k.

The `Shake` procedure receives two input parameters: an initial solution S and the size of the perturbation k. In each iteration, k indicates the number of perturbations to perform. At the end of this procedure, a solution in a different neighborhood is returned. We present a pseudocode of this procedure in Algorithm 3.

Algorithm 3. `Shake`(S, k)

```
1: repeat
2:     repeat
3:         B_i ← ChooseRandomBatch(S)
4:         B_j ← ChooseRandomBatch(S)
5:     until B_i ≠ B_j
6:     o_i ← ChooseRandomOrder(B_i)
7:     o_j ← ChooseRandomOrder(B_j)
8:     if Fits(B_i \ o_i, o_j) and Fits(B_j \ o_j, o_i) then
9:         B_i ← B_i \ o_i
10:        Add(B_i, o_j)
11:        B_j ← B_j \ o_j
12:        Add(B_j, o_i)
13:        k ← k − 1
14:    end if
15: until k = 0
16: return S
```

3.3 Local Search Procedure

The BVNS uses a local search in order to deterministically find a local optimum in the current neighborhood. The local search proposed here is based in the one-to-one exchange move. The only input parameter to the local search is a solution S. The method will return another solution which is locally optimum with respect to the initial solution and the neighborhood defined by the exchange move. The local search ends when all candidate interchanges have been explored and no one produces an improve in the current solution. We present the pseudocode of the local search procedure in Algorithm 4.

Algorithm 4. `LocalSearch`(S)

```
repeat
    improved ← false
    for ∀ o_i ∈ S do
        for ∀ o_j ∈ S do
            S' ← Exchange(S, o_i, o_j)
            if f(S') < f(S) then
                S ← S'
                improved ← true
                break
            end if
        end for
    end for
until improved = false
return S
```

4 Results

In order to test our proposals, we compare our BVNS with the Seed method introduced in [94] and described in Sect. 2.1. The objective function used to compare the algorithms is the total time needed to collect all the orders in a context where the number of pickers is two. The experiments were run 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, and all the codes were developed in Java 8.

The 64 instances used in our experiments were derived from those reported in [32]. Those instances represent a real warehouse with one block, rectangular shape and 900 storage positions. Particularly, there are 10 aisles with shelves at both sides of the aisle and 45 picking positions in each side. The warehouse layout and the distribution of the items are key elements in the design of the batches. We consider both: random and ABC sorting strategies of the items. The depot (i.e., the place where the items are handed once they have been collected) is placed in the front cross-aisle, either in the left corner or in center of the aisle.

As far as the orders are concerned, the instances contain different number of orders (40, 60, 80, 100). Also there are several picking-cart sizes (30, 45, 60, 75). It is important to notice that, not all the orders represented in each instance are available at the beginning of the process, but they arrive through the working time observed. Particularly, in this paper we consider a time horizon of 4 h. This means that we will observe the behaviour of our algorithms in four hours (remember that in an online problem, the system is actually working 24 h). Then, then number of orders of each instance must arrive to the system in four hours. We use an exponencial distribution for determining the instant in the time when each order arrives to the system, as it is customary in this kind of scenarios.

The BVNS algorithm was parameterized with $k_{max} = 5$ and $t_{max} = 30$ s, and then it has been successfully compared with a variant of the Seed algorithm described in the Sect. 2 using different routing methods. In Table 2 we report the results obtained when using the S-Shape routing method. Similarly, in Table 3 and Table 4 we report the results obtained when using the Largest Gap and Combined routing methods, respectively. For each table, we report the average time used to collect all the orders (Avg. (s)), the deviation with respect to the best value found in the experiment (Dev. (%)) and the number of best solutions found (#Best). In these three experiments we have compared both methods (BVNS and Seed) using the same routing algorithm. Therefore, the results obtained are merit just from the batching strategy. As it is possible to see observing these three tables, BVNS is consistently better than Seed considering both: deviation and number of best solutions found. However, the differences in deviation are very small for the three routing methods. Additionally, when using the Largest Gap routing method, the number of best solutions found by the BVNS and Seed method are almost the same.

However, despite of the fact that we have paired either BVNS and Seed methods with three routing strategies, the original proposal of the Seed method introduced in [94] was based only on the S-Shape routing algorithm. Next, we compare the results obtained by our BVNS paired with Largest Gap and Combined routing methods with respect to the Seed method paired with S-Shape (as it is described by the authors). The results are reported in Tables 5 and 6 respectively. In both cases, the deviation obtained has been considerably improved with respect to the method in the state of the art. Similarly, the number of best-known values has also been increased.

In order to facilitate future comparisons, in the Appendix A we report the best values found per each of the instances considered in this paper.

Table 2. Comparison with the state of the art using the S-Shape routing method.

Batching	BVNS	Seed [94]
Routing	S-Shape	S-Shape
Avg. (s)	32886	33046
Dev. (%)	0,29%	0,91%
#Best	48	28

Table 3. Comparison with the state of the art using the Largest Gap routing method.

Batching	BVNS	Seed [94]
Routing	Largest Gap	Largest Gap
Avg. (s)	32315	32309
Dev. (%)	0,44%	0,51%
#Best	37	36

Table 4. Comparison with the state of the art using the Combined routing method.

Batching	BVNS	Seed [94]
Routing	Combined	Combined
Avg. (s)	31420	31534
Dev. (%)	0,21%	0,66%
#Best	46	24

Table 5. Comparison between the BVNS paired with Largest Gap with respect to the Seed method paired with S-Shape.

Batching	BVNS	Seed [94]
Routing	Largest Gap	S-Shape
Avg. (s)	32315	33046
Dev. (%)	0,91%	3,18%
#Best	42	22

Table 6. Comparison between the BVNS paired with Combined with respect to the Seed method paired with S-Shape.

Batching	BVNS	Seed [94]
Routing	Combined	S-Shape
Avg. (s)	31420	33046
Dev. (%)	0,04%	5,01%
#Best	60	4

5 Conclusions

In this paper we have dealt with a variant of the Online Order Batching Problem. Particularly, the variant which considers multiple pickers to collect the items in the batches. We have reviewed the state of the art of the problem and highlighted the latest approach to tackle it. We have also designed an algorithm, based on the Basic Variable Neighborhood Search methodology, in order to provide good

quality solutions for the OOBPMP. The obtained results have been compared with the state of the art using different routing algorithms. In all the considered cases, the BVNS proposed improved the previous method in the state of the art. We also noticed that the use of the Combined routing method was the most effective among the considered ones for this problem.

A Best-Known Values per Instance

See Table 7.

Table 7. Best-known values of the objective function Time (s) per instance.

Instance	Time (s)	Instance	Time (s)	Instance	Time (s)
abc1_40_29	23992	**abc1_40_30**	22398	**abc1_40_31**	22526
abc1_40_32	22528	**abc1_60_37**	31029	**abc1_60_38**	28469
abc1_60_39	25501	**abc1_60_40**	24796	**abc1_80_61**	41391
abc1_80_62	33856	**abc1_80_63**	29225	**abc1_80_64**	29963
abc1_100_69	44026	**abc1_100_70**	35298	**abc1_100_71**	32068
abc1_100_72	31089	**abc2_40_9**	23942	**abc2_40_10**	21370
abc2_40_11	21098	**abc2_40_12**	20909	**abc2_60_17**	29979
abc2_60_18	29193	**abc2_60_19**	27954	**abc2_60_20**	22775
abc2_80_45	38246	**abc2_80_46**	31943	**abc2_80_47**	30175
abc2_80_48	28280	**abc2_100_53**	49381	**abc2_100_54**	35574
abc2_100_55	36281	**abc2_100_56**	30630	**ran1_40_29**	26011
ran1_40_30	23601	**ran1_40_31**	24904	**ran1_40_32**	21948
ran1_60_37	35830	**ran1_60_38**	30448	**ran1_60_39**	26893
ran1_60_40	27907	**ran1_80_61**	47704	**ran1_80_62**	39016
ran1_80_63	31755	**ran1_80_64**	30074	**ran1_100_69**	52084
ran1_100_70	39676	**ran1_100_71**	35887	**ran1_100_72**	33779
ran2_40_9	25476	**ran2_40_10**	25794	**ran2_40_11**	22076
ran2_40_12	21629	**ran2_60_17**	33661	**ran2_60_18**	30619
ran2_60_19	26974	**ran2_60_20**	26496	**ran2_80_45**	44121
ran2_80_46	36148	**ran2_80_47**	32117	**ran2_80_48**	30286
ran2_100_53	56735	**ran2_100_54**	40345	**ran2_100_55**	40688
ran2_100_56	33380				

References

1. Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., De Blas, C.S.: Variable neighborhood search for order batching in a warehouse. Asia Pac. J. Oper. Res. **26**(5), 655–683 (2009)

2. Ardjmand, E., Bajgiran, O.S., Youssef, E.: Using list-based simulated annealing and genetic algorithm for order batching and picker routing in put wall based picking systems. Appl. Soft Comput. **75**, 106–119 (2019)
3. Ardjmand, E., Shakeri, H., Singh, M., Bajgiran, O.S.: Minimizing order picking makespan with multiple pickers in a wave picking warehouse. Int. J. Prod. Econ. **206**, 169–183 (2018)
4. Azadnia, A.H., Taheri, S., Ghadimi, P., Mat Saman, M.Z., Wong, K.Y.: Order batching in warehouses by minimizing total tardiness: a hybrid approach of weighted association rule mining and genetic algorithms. Sci. World J. **2013** (2013)
5. Bozer, Y.A., Kile, J.W.: Order batching in walk-and-pick order picking systems. Int. J. Prod. Res. **46**(7), 1887–1909 (2008)
6. Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A.L., Ogier, M.: A column generation based approach for the joint order batching and picker routing problem. In: ROADEF 2018 (2018)
7. Bué, M., Cattaruzza, D., Ogier, M., Semet, F.: An integrated order batching and picker routing problem. HAL (hal-01849980) (2018)
8. Bustillo, M., Menéndez, B., Pardo, E.G., Duarte, A.: An algorithm for batching, sequencing and picking operations in a warehouse. In: 2015 International Conference on Industrial Engineering and Systems Management (IESM), pp. 842–849, October 2015
9. Cano, J.A., Correa-Espinal, A.A., Gómez-Montoya, R.A.: Solución del problema de conformación de lotes en almacenes utilizando algoritmos genéticos. Información tecnológica **29**(6), 235–244 (2018)
10. Chen, F., Wei, Y., Wang, H.: A heuristic based batching and assigning method for online customer orders. Flex. Serv. Manuf. J. **30**(4), 640–685 (2018). https://doi.org/10.1007/s10696-017-9277-7
11. Chew, E.P., Tang, L.C.: Travel time analysis for general item location assignment in a rectangular warehouse. Eur. J. Oper. Res. **112**(3), 582–597 (1999)
12. Coyle, J.J., Bardi, E.J., Langley, C.J., et al.: The Management of Business Logistics, vol. 6. West Publishing Company Minneapolis, St Paul (1996)
13. De Koster, M.B.M., Van der Poort, E.S., Wolters, M.: Efficient order batching methods in warehouses. Int. J. Prod. Res. **37**(7), 1479–1504 (1999)
14. de Koster, R., Roodbergen, K.J., van Voorden, R.: Reduction of walking time in the distribution center of De Bijenkorf. In: Speranza, M.G., Stähly, P. (eds.) New Trends in Distribution Logistics. Lecture Notes in Economics and Mathematical Systems, vol. 480, pp. 215–234. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-642-58568-5_11
15. De Koster, R., Le-Duc, T., Roodbergen, K.J.: Design and control of warehouse order picking: a literature review. Eur. J. Oper. Res. **182**(2), 481–501 (2007)
16. Drury, J.: Towards more efficient order picking. IMM Monograph, No. 1 (1988)
17. Duarte, A., Pantrigo, J.J., Pardo, E.G., Mladenovic, N.: Multi-objective variable neighborhood search: an application to combinatorial optimization problems. J. Glob. Optim. **63**(3), 515–536 (2015). https://doi.org/10.1007/s10898-014-0213-z
18. Duarte, A., Pantrigo, J.J., Pardo, E.G., Sánchez-Oro, J.: Parallel variable neighbourhood search strategies for the cutwidth minimization problem. IMA J. Manag. Math. **27**(1), 55–73 (2016)
19. Elsayed, E., Lee, M.K.: Order processing in automated storage/retrieval systems with due dates. IIE Trans. **28**(7), 567–577 (1996)
20. Elsayed, E.A.: Algorithms for optimal material handling in automatic warehousing systems. Int. J. Prod. Res. **19**(5), 525–535 (1981)

21. van der Gaast, J.P., Jargalsaikhan, B., Roodbergen, K.J.: Dynamic batching for order picking in warehouses. In: 15th IMHRC Proceedings, Savannah, Georgia, USA (2018)
22. Gademann, A.J.R.M., Van Den Berg, J.P., Van Der Hoff, H.H.: An order batching algorithm for wave picking in a parallel-aisle warehouse. IIE Trans. **33**(5), 385–398 (2001)
23. Gademann, N., Velde, S.: Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Trans. **37**(1), 63–75 (2005)
24. Galka, S., Ulbrich, A., Günthner, W.: Performance calculation for order picking systems by analytical methods and simulation. Technical report, Technische Universität München, München, Germany (2008)
25. Gibson, D.R., Sharp, G.P.: Order batching procedures. Eur. J. Oper. Res. **58**(1), 57–67 (1992)
26. Gil-Borrás, S., Duarte, A., Alonso-Ayuso, A., Pardo, E.G.: Búsqueda de vecindad variable para el problema de la agrupación y recogida de pedidos online en almacenes logísticos. In: XVIII Conferencia de la Asociación Española para la Inteligencia Artificial, Granada, España, pp. 551–556, October 2018
27. Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte, A.: New VNS variants for the online order batching problem. In: Sifaleras, A., Salhi, S., Brimberg, J. (eds.) ICVNS 2018. LNCS, vol. 11328, pp. 89–100. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15843-9_8
28. Hall, R.W.: Distance approximations for routing manual pickers in a warehouse. IIE Trans. **25**(4), 76–87 (1993)
29. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
30. Hansen, P., Mladenović, N., Moreno-Pérez, J.A.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010). https://doi.org/10.1007/s10479-009-0657-6
31. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2017). https://doi.org/10.1007/s13675-016-0075-x
32. Henn, S.: Algorithms for on-line order batching in an order picking warehouse. Comput. Oper. Res. **39**(11), 2549–2563 (2012)
33. Henn, S., Schmid, V.: Metaheuristics for order batching and sequencing in manual order picking systems. Comput. Ind. Eng. **66**(2), 338–351 (2013)
34. Henn, S., Wäscher, G.: Tabu search heuristics for the order batching problem in manual order picking systems. Eur. J. Oper. Res. **222**(3), 484–494 (2012)
35. Henn, S.: Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. Flex. Serv. Manuf. J. **27**(1), 86–114 (2015). https://doi.org/10.1007/s10696-012-9164-1
36. Henn, S., Koch, S., Doerner, K.F., Strauss, C., Wäscher, G.: Metaheuristics for the order batching problem in manual order picking systems. Bus. Res. **3**(1), 82–105 (2010)
37. Henn, S., et al.: Variable neighborhood search for the order batching and sequencing problem with multiple pickers. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management (2012)
38. Ho, Y.C., Tseng, Y.Y.: A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. Int. J. Prod. Res. **44**(17), 3391–3417 (2006)
39. Ho, Y.C., Su, T.S., Shi, Z.B.: Order-batching methods for an order-picking warehouse with two cross aisles. Comput. Ind. Eng. **55**(2), 321–347 (2008)

40. Hong, S., Johnson, A.L., Peters, B.A.: Analysis of picker blocking in narrow-aisle batch picking. Technical report, Texas A&M University (2010)
41. Hong, S., Johnson, A.L., Peters, B.A.: Batch picking in narrow-aisle order picking systems with consideration for picker blocking. Eur. J. Oper. Res. **221**(3), 557–570 (2012)
42. Hsu, C.M., Chen, K.Y., Chen, M.C.: Batching orders in warehouses by minimizing travel distance with genetic algorithms. Comput. Ind. **56**(2), 169–178 (2005)
43. Huang, M., Guo, Q., Liu, J., Huang, X.: Mixed model assembly line scheduling approach to order picking problem in online supermarkets. Sustainability **10**(11), 3931 (2018)
44. Jiang, X., Zhou, Y., Zhang, Y., Sun, L., Hu, X.: Order batching and sequencing problem under the pick-and-sort strategy in online supermarkets. Procedia Comput. Sci. **126**, 1985–1993 (2018)
45. Kamin, N.: On-line optimization of order picking in an automated warehouse. Ph.D. thesis, Technische Universität Belin, Belin, Germany (1998)
46. Koch, S., Wäscher, G.: A grouping genetic algorithm for the Order Batching Problem in distribution warehouses. J. Bus. Econ. **86**(1–2), 131–153 (2016). https://doi.org/10.1007/s11573-015-0789-x
47. Koster, R.D., Poort, E.V.D.: Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. IIE Trans. **30**(5), 469–480 (1998)
48. Le-Duc, T.: Design and control of efficient order picking processes. Ph.D. thesis, Erasmus University Rotterdam. Erasmus Research Institute of Management, Rotterdam, Holland, September 2005
49. Lenoble, N., Frein, Y., Hammami, R.: Optimization of order batching in a picking system with a vertical lift module. In: Temponi, C., Vandaele, N. (eds.) ILS 2016. LNBIP, vol. 262, pp. 153–167. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73758-4_11
50. Lenoble, N., Frein, Y., Hammami, R.: Optimization of order batching in a picking system with carousels. In: 20th World Congress of the International Federation of Automatic Control, IFAC 2017 (2017)
51. Lin, C.C., Kang, J.R., Hou, C.C., Cheng, C.Y.: Joint order batching and picker manhattan routing problem. Comput. Ind. Eng. **95**, 164–174 (2016)
52. Menéndez, B., Bustillo, M., Pardo, E.G., Duarte, A.: General variable neighborhood search for the order batching and sequencing problem. Eur. J. Oper. Res. **263**(1), 82–93 (2017)
53. Menéndez, B., Pardo, E.G., Alonso-Ayuso, A., Molina, E., Duarte, A.: Variable neighborhood search strategies for the order batching problem. Comput. Oper. Res. **78**, 500–512 (2017)
54. Menéndez, B., Pardo, E.G., Duarte, A., Alonso-Ayuso, A., Molina, E.: General variable neighborhood search applied to the picking process in a warehouse. Electron. Notes Discrete Math. **47**, 77–84 (2015)
55. Menéndez, B., Pardo, E.G., Sánchez-Oro, J., Duarte, A.: Parallel variable neighborhood search for the min-max order batching problem. Int. Trans. Oper. Res. **24**(3), 635–662 (2017)
56. Menéndez, B., Pardo, E.G., Duarte, A.: Búsqueda de vecindad variable general aplicada al proceso de recogida de productos en almacenes. In: XVI Conferencia de la Asociación Española para la Inteligencia Artificial, Albacete, España, Noviembre 2015
57. Miguel, F., Frutos, M., Tohmé, F., Rossit, D.: A memetic algorithm for the integral obp/opp problem in a logistics distribution center. Uncertain Supply Chain. Manag. **7**(2), 203–214 (2019)

58. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
59. Öncan, T.: MILP formulations and an iterated local search algorithm with Tabu thresholding for the order batching problem. Eur. J. Oper. Res. **243**(1), 142–155 (2015)
60. Öncan, T., Cağırıcı, M.: MILP formulations for the order batching problem in low-level picker-to-part warehouse systems. IFAC Proc. Vol. **46**(9), 471–476 (2013)
61. Oncan, T.: A genetic algorithm for the order batching problem in low-level picker-to-part warehouse systems. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1 (2013)
62. Pan, C.H., Liu, S.Y.: A comparative study of order batching algorithms. Omega **23**(6), 691–700 (1995)
63. Pan, J.C.H., Shih, P.H., Wu, M.H.: Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega **57**, 238–248 (2015)
64. Pardo, E.G., Mladenović, N., Pantrigo, J.J., Duarte, A.: Variable formulation search for the cutwidth minimization problem. Appl. Soft Comput. **13**(5), 2242–2252 (2013)
65. Parikh, P.J.: Designing order picking systems for distribution centers. Ph.D. thesis, Virginia Tech. Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA (2006)
66. Pérez-Rodríguez, R., Hernández-Aguirre, A., Jöns, S.: A continuous estimation of distribution algorithm for the online order-batching problem. Int. J. Adv. Manuf. Technol. **79**(1), 569–588 (2015). https://doi.org/10.1007/s00170-015-6835-6
67. Pérez-Rodríguez, R., Hernández-Aguirre, A.: An estimation of distribution algorithm-based approach for the order batching problem. Res. Comput. Sci. **93**, 141–150 (2015)
68. Pérez-Rodríguez, R., Hernández-Aguirre, A.: An estimation of distribution algorithm-based approach for the order batching problem: an experimental study. In: Handbook of Research on Military, Aeronautical, and Maritime Logistics and Operations, pp. 509–518. IGI Global (2016)
69. Pérez-Rodríguez, R., Hernández-Aguirre, A.: Finding interactions or relationships between customer orders for building better batches by means of an estimation of distribution algorithm-based approach for the online order batching problem. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp. 989–996. ACM (2016)
70. Petersen, C.G.: An evaluation of order picking routeing policies. Int. J. Oper. Prod. Manag. **17**(11), 1098–1111 (1997)
71. Petersen, C.: Routeing and storage policy interaction in order picking operations. Decis. Sci. Inst. Proc. **31**(3), 1614–1616 (1995)
72. Postema, J.T.: Metaheuristics for order batching in ecommerce warehouses. In: 27th Twente Student Conference on IT. University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, Enschede, The Netherlands, July 2017
73. Raj, L.S., Girubha, R.J.: Aggregation of order picking system using order batching. IOSR J. Mech. Civ. Eng. **11**(2), 01–04 (2014)
74. Ratliff, H.D., Rosenthal, A.S.: Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper. Res. **31**(3), 507–521 (1983)
75. Roodbergen, K.J., Petersen, C.G.: How to improve order picking efficiency with routing and storage policies. In: Progress in Material Handling Practice, pp. 107–124 (1999)

76. Rosenwein, M.B.: A comparison of heuristics for the problem of batching orders for warehouse selection. Int. J. Prod. Res. **34**(3), 657–664 (1996)
77. Rubrico, J., Higashi, T., Tamura, H., Ota, J.: Online rescheduling of multiple picking agents for warehouse management. Robot. Comput. Integr. Manuf. **27**(1), 62–71 (2011)
78. Scholz, A., Wäscher, G.: Order Batching and Picker Routing in manual order picking systems: the benefits of integrated routing. Cent. Eur. J. Oper. Res. **25**(2), 491–520 (2017). https://doi.org/10.1007/s10100-017-0467-x
79. Scholz, A., Schubert, D., Wäscher, G.: Order picking with multiple pickers and due dates-simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. Eur. J. Oper. Res. **263**(2), 461–478 (2017)
80. Schubert, D., Scholz, A., Wäscher, G.: Integrated order picking and vehicle routing with due dates. OR Spectr. **40**(4), 1109–1139 (2018). https://doi.org/10.1007/s00291-018-0517-3
81. Tang, L.C., Chew, E.P.: Order picking systems: batching and storage assignment strategies. Comput. Ind. Eng. **33**(3), 817–820 (1997). Selected Papers from the Proceedings of 1996 ICC&IC
82. Tsai, C.Y., Liou, J.J., Huang, T.M.: Using a multiple-ga method to solve the batch picking problem: considering travel distance and order due time. Int. J. Prod. Res. **46**(22), 6533–6555 (2008)
83. Valle, C.A., Beasley, J.E., da Cunha, A.S.: Optimally solving the joint order batching and picker routing problem. Eur. J. Oper. Res. **262**(3), 817–834 (2017)
84. Valle, C.A., Beasley, J.E.: Order batching for picker routing using a distance approximation. arXiv preprint arXiv:1808.00499 (2018)
85. Van Gils, T., Braekers, K., Ramaekers, K., Depaire, B., Caris, A.: Improving order picking efficiency by analyzing the combination of storage, batching, zoning and routing policies in a 2-block warehouse. Technical report, Hasselt University, Martelarenlaan 42, 3500 Hasselt, Belgium (2016)
86. Van Nieuwenhuyse, I., de Koster, R., Colpaert, J.: Order batching in multi-server pick-and-sort warehouses. Katholieke Universiteit Leuven, Department of Decision Sciences and Information Management, vol. 180, no. 140, pp. 367–8869 (2007)
87. Van Nieuwenhuyse, I., de Koster, R.B.: Evaluating order throughput time in 2-block warehouses with time window batching. Int. J. Prod. Econ. **121**(2), 654–664 (2009)
88. Verschure, A.: Improving picking efficiency in a warehouse with multiple floors at Docdata NV. Ph.D. thesis, Technische Universities Eindhoven, Eindhoven, Netherlands (2014)
89. Wäscher, G., Scholz, A., et al.: A solution approach for the joint order batching and picker routing problem in a two-block layout. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management (2015)
90. Wasusri, T., Theerawongsathon, P.: An application of discrete event simulation on order picking strategies: A case study of footwear warehouses. In: Claus, T., Frank Herrmann, M.M.O.R. (eds.) Proceedings 30th European Conference on Modelling and Simulation - ECMS, pp. 121–127 (2016)
91. Won, J., Olafsson, S.: Joint order batching and order picking in warehouse operations. Int. J. Prod. Res. **43**(7), 1427–1442 (2005)
92. Won, J.: Order batching and picking optimization in terms of supply chain management. Ph.D. thesis, Iowa State University, Iowa, USA (2004)
93. Yu, M.M.: Enhancing warehouse performance by efficient order picking. Ph.D. thesis, Erasmus University Rotterdam. Erasmus Research Institute of Management, Rotterdam, Holland (2008)

94. Zhang, J., Wang, X., Chan, F.T.S., Ruan, J.: On-line order batching and sequencing problem with multiple pickers: a hybrid rule-based algorithm. Appl. Math. Model. **45**, 271–284 (2017)
95. Zhang, J., Wang, X., Huang, K.: Integrated on-line scheduling of order batching and delivery under B2C e-commerce. Comput. Ind. Eng. **94**, 280–289 (2016)
96. Zhang, J., Wang, X., Huang, K.: On-line scheduling of order picking and delivery with multiple zones and limited vehicle capacity. Omega **79**, 104–115 (2018)
97. Zhu, J., Zhang, H., Zhou, L., Guo, J.: Order batching optimization in dual zone type warehouse based on genetic algorithms. Sci. J. Bus. Manag. **3**(3), 77–81 (2015)
98. Žulj, I., Kramer, S., Schneider, M.: A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. Eur. J. Oper. Res. **264**(2), 653–664 (2018)
99. Zuniga, C., Olivares-Benitez, E., Tenahua, A., Mujica, M.: A methodology to solve the order batching problem. IFAC-PapersOnLine **48**(3), 1380–1386 (2015)

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

Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte A. (2021). A heuristic approach for the online order batching problem with multiple pickers. Computers & Industrial Engineering, 160 (p. 107517).

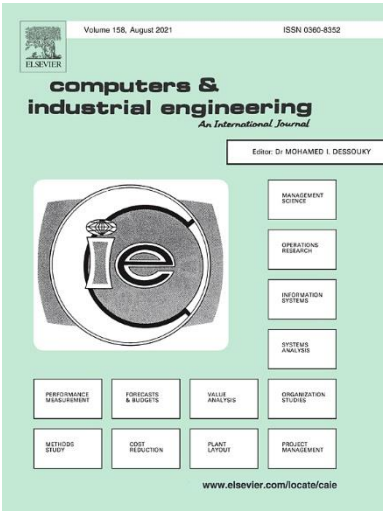
Published in "**Computers & Industrial Engineering**"

ISSN: 0360-8352 / 1879-0550

https://doi.org/10.1016/j.cie.2021.107517

Impact factor (JCR 2020): 5.431

Journal Citation Indicator (JCI): 1.18



Rank by Journal Impact Factor

- Computer science, Interdisciplinary applications.
  Ranking 21/111 (Q1).
- Engineering, Industrial.
  Ranking 14/49 (Q2).

Rank by Journal Citation Indicator (JCI)

- Computer science, Interdisciplinary applications.
  Ranking 40/142 (Q2).
- Engineering, Industrial.
  Ranking 13/62 (Q1).

Rank by Journal Impact Factor

Journals within a category are sorted in descending order by Journal Impact Factor (JIF) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

EDITION
Science Citation Index Expanded (SCIE)
CATEGORY
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS
21/111

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	21/111	Q1	81.53
2019	18/109	Q1	83.94
2018	24/106	Q1	77.83
2017	22/105	Q1	79.52
2016	28/105	Q2	73.81

EDITION
Science Citation Index Expanded (SCIE)
CATEGORY
ENGINEERING, INDUSTRIAL
14/49

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	14/49	Q2	72.45
2019	10/48	Q1	80.21
2018	11/46	Q1	77.17
2017	9/47	Q1	81.91
2016	9/44	Q1	80.68

Rank by Journal Citation Indicator (JCI)

Journals within a category are sorted in descending order by Journal Citation Indicator (JCI) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

CATEGORY
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS
40/142

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	40/142	Q2	72.18
2019	41/140	Q2	71.07
2018	39/138	Q2	72.10
2017	38/136	Q2	72.43

CATEGORY
ENGINEERING, INDUSTRIAL
13/62

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	13/62	Q1	79.84
2019	12/62	Q1	81.45
2018	13/60	Q1	79.17
2017	14/57	Q1	76.32

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

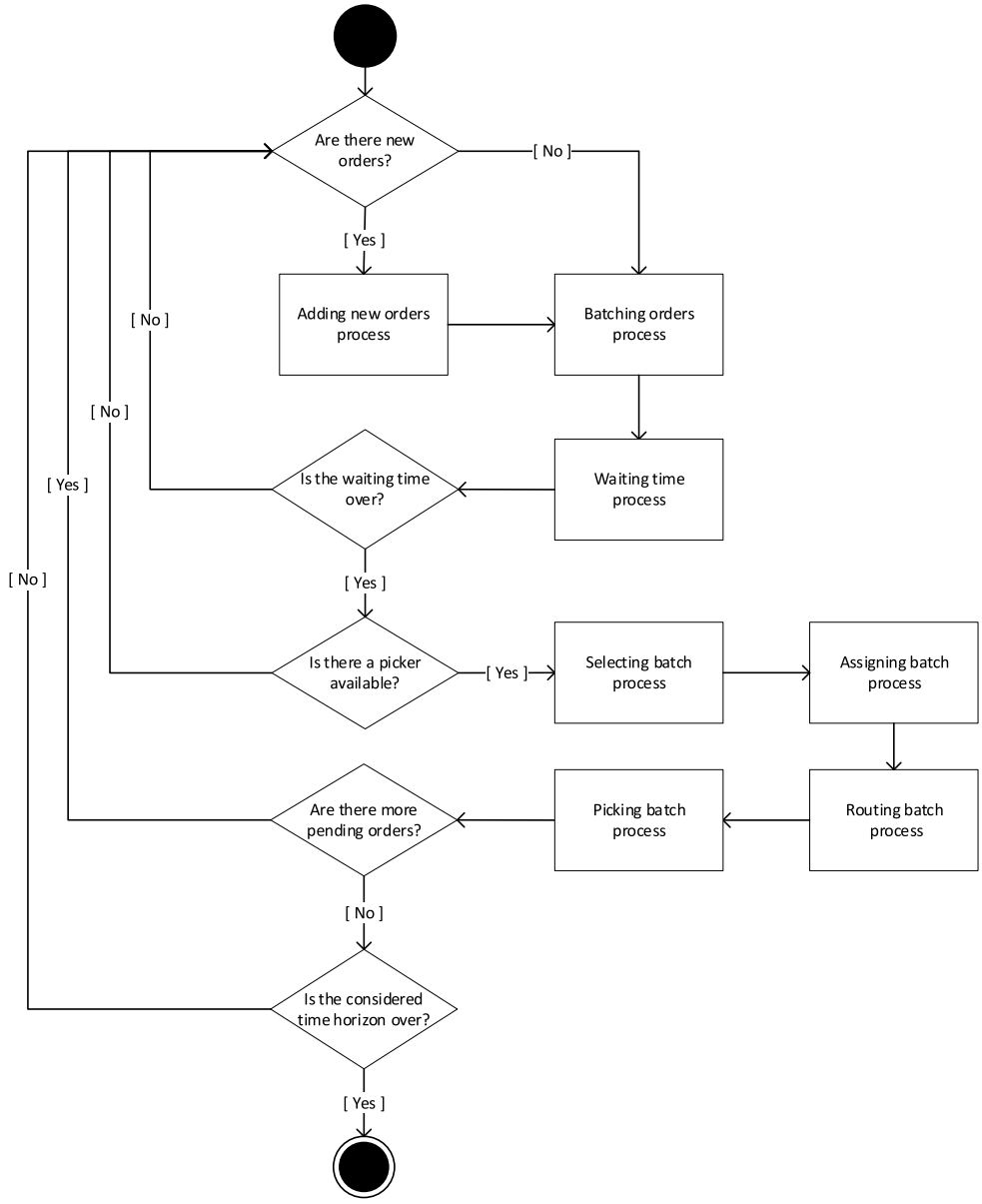


Fig. 1. Activity diagram of the processes involved in the OOBPMP.

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

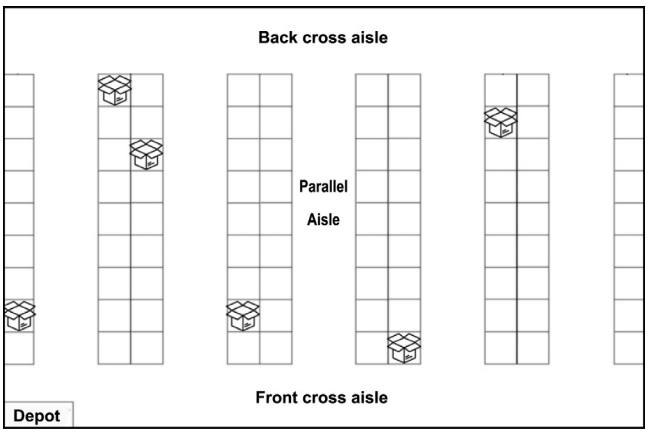


Fig. 2. Layout of the warehouse studied in this paper, with two cross aisles and a variable number of parallel aisles.

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 (`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 ($\arg\max$ `f`(CL)) and minimum ($\arg\min$ `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: $th \leftarrow \arg\max$ `f`$(CL) - \alpha(\arg\max$ `f`$(CL) - \arg\min$ `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

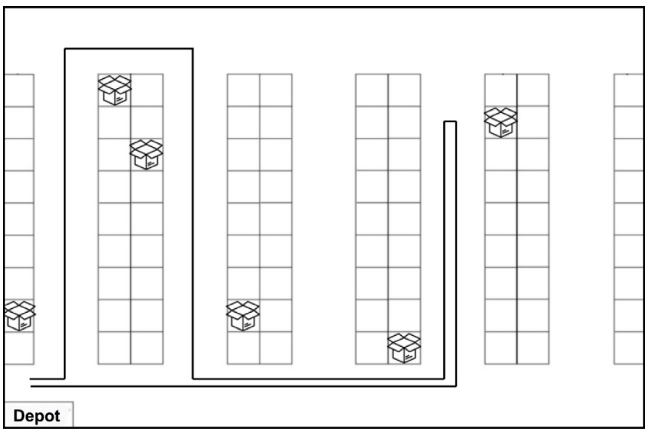


Fig. 3. Example of the route obtained with the S-Shape routing algorithm to collect the boxes in the figure.

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.

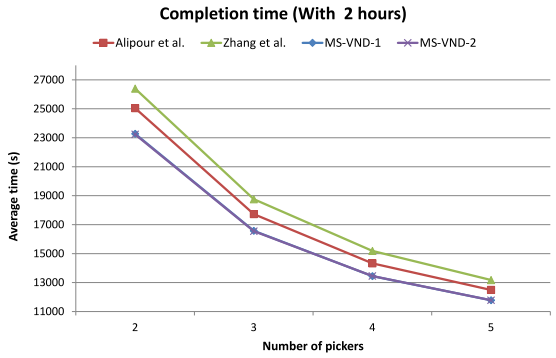
As a general conclusion, the congestion rate depends on the arrival of orders in a particular time horizon, and on the pickers available at the warehouse. Also, we can conclude that there exists a relationship between this rate and the studied objective functions. Studying the congestion rate could be 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

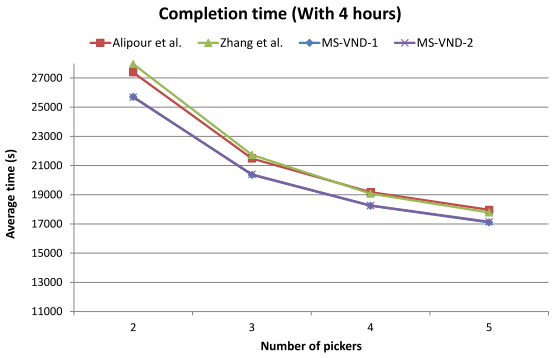
In this paper we have studied the Online Order Batching Problem with Multiple Pickers. This problem is one of the most realistic

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%

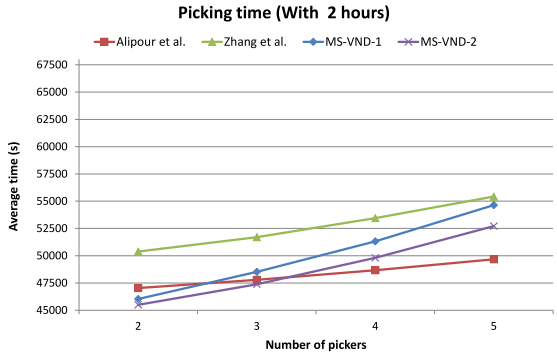


(a) Averaged completion time for 2 hours scenario.

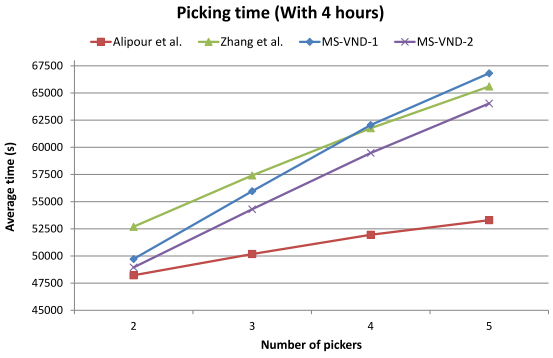


(b) Averaged completion time for 4 hourss cenario.

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



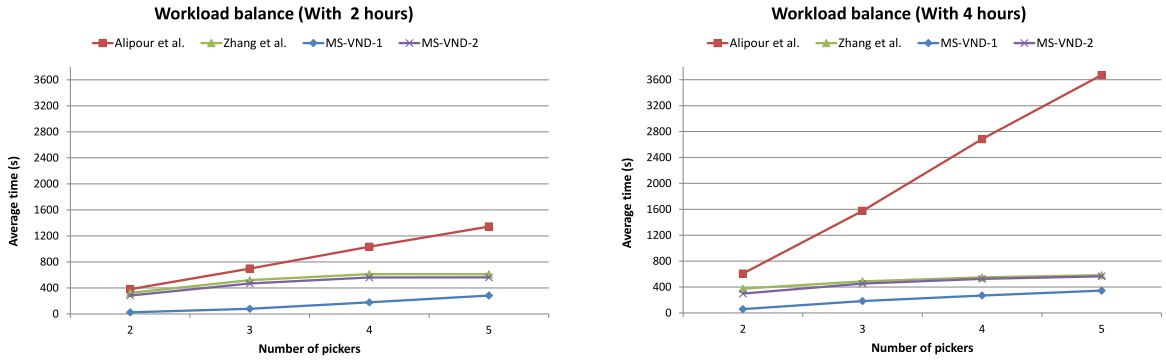
(a) Averaged picking time for 2 hours scenario.



(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.

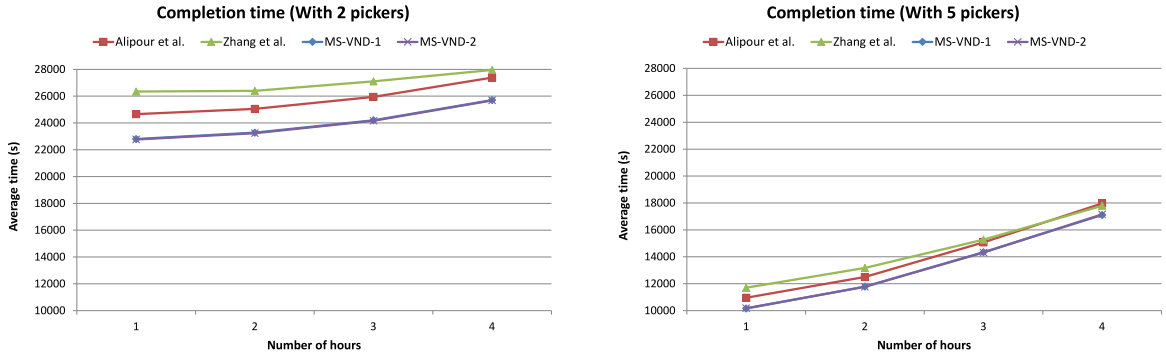
variants of the Order Batching family of problems, since it considers the existence of several pickers in the warehouse at the same time, and the online arrival of orders to the system. We have classified the studied variant and identified all previous methods in the state of the art. We noticed that 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.

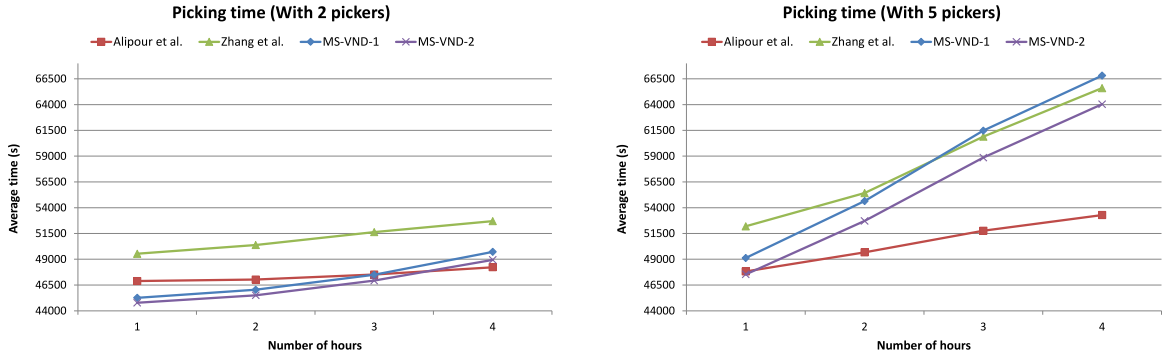
Fig. 6. Performance of the algorithms, in terms of workload balance, when increasing the number of pickers for 2 and 4 h scenarios.



(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.

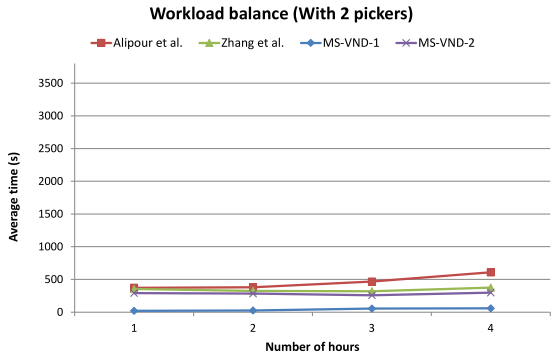
(b) Averaged picking time for 5 pickers scenario.

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.

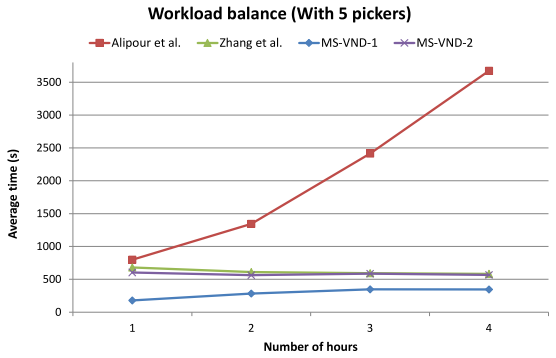
system and the influence of the congestion rate on the arrival of orders in several scenarios. Next, we expose our conclusions derived from the analysis of the obtained results.

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.

Chapter 6

Online Order Batching Problem with Time window

The resolution of any variant of the Online Order Batching Problem includes to solve different tasks. Among them, the waiting time consists of determining the time that a picker should wait for the arrival of new orders, before starting a new route. This task has been little studied in the literature despite the high impact that it has in the overall performance of order batching methods. In this Doctoral Thesis, we have performed a study about the influence of the waiting task and, as the result of the research performed, two articles have been elaborated (one of them still under review) which are presented next. Then, for each publication, we compile the bibliographic details of the publication (complete reference, journal, ranking index, category, and ranking score) and, finally, a copy of the article is attached.

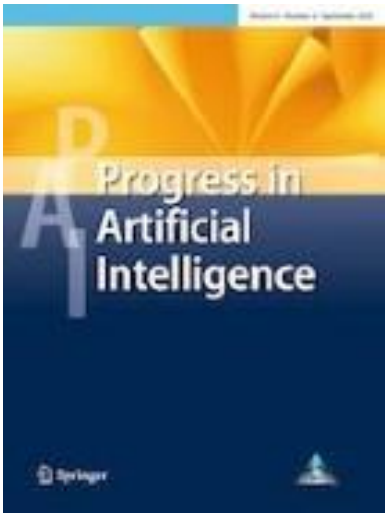
The article "*Fixed versus variable time window warehousing strategies in real time*" [91] was the fifth work carried out for this Doctoral Thesis and presents the preliminary study about solving the Online Order Batching Problem when considering a time window strategy. The objective of this article is to study and compare different waiting strategies. Specifically, we compared four Fixed Time Window (FTW) strategies based on waiting a prefixed amount of time, and three Variable Time Window (VTW) strategies based on the number of orders arrived to the system. The objective functions used to compare the algorithms are: the minimization of the completion time and the minimization of the picking time, in an scenario with only one picker. The arrival of orders occurs in a period of 4 hours. The problem is solved for a rectangular warehouse with a single block, with 10 parallel aisles and a total of 900 picking positions/stored products. The set of instances used for the problem is a reduced set of 16 instances widely used in the literature [125]. We evaluated the influence of time window strategies considering two batching algorithms (First Come First Served and a greedy algorithm based on weight). The routing algorithm used was S-Shape, and the selecting algorithm for choosing the next batch to be collected was a greedy algorithm based on the weight/occupancy of the batch. We identified that a FTW strategy of no waiting was the best technique to reduce the completion time. However, a VTW consisting in waiting until 16 orders have arrived to the system was the best proposal to reduce the picking time in the evaluated scenarios. This article is attached in Section 6.1.

The technical report "*A comparative study of the influence of the time-window strategy in the Online Order Batching Problem*" [95] is currently under review (second round) in a JCR journal. This technical report is the sixth article, chronologically speaking, obtained as the result of the research performed in this Doctoral Thesis, and can be considered as an evolution of the one presented in [91]. In this paper, our objective is to determine the real influence of using a waiting strategy on the overall performance of methods which handle the OBP in online contexts and to determine if the waiting time is dependent on other tasks such as routing or

batching. To that aim we explored a wide range of working scenarios. Particularly, in this work, we studied three different objective functions for the problem: minimizing the completion time, minimizing the picking time, and minimizing the maximum turnover time. Also, we studied the problem when varying the congestion of the system and the number of pickers (from 1 to 5), the batching algorithm (First Come First Served, ILS, and GVNS), and the routing algorithm (S-Shape, Largest Gap, and Combined). Finally, we reviewed the waiting strategies in the literature and experimentally evaluated a total of eight different waiting strategies (two of them proposed in this work). We used two different sets of instances widely used in the literature [4, 125]. These sets represented a total of five different rectangular warehouse layouts with a single block and different number of parallel aisles. The arrival of orders occurred in a period of 4 hours. As a conclusion of this research we were able to associate different behaviors of the algorithms for different Time-Window strategies. Furthermore, the two new time-window methods proposed are able to improve in various scenarios the methods previously proposed in the state of the art. This article is currently under review and its latest version is attached in Section 6.2.

6.1 Fixed versus variable time window warehousing strategies in real time

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



Published in "**Progress in Artificial Intelligence**"

ISSN: 2192-6352 / 2192-6360

https://doi.org/10.1007/s13748-020-00215-1

Journal Citation Indicator (JCI): 0.28

CiteScore 2020: 3.4

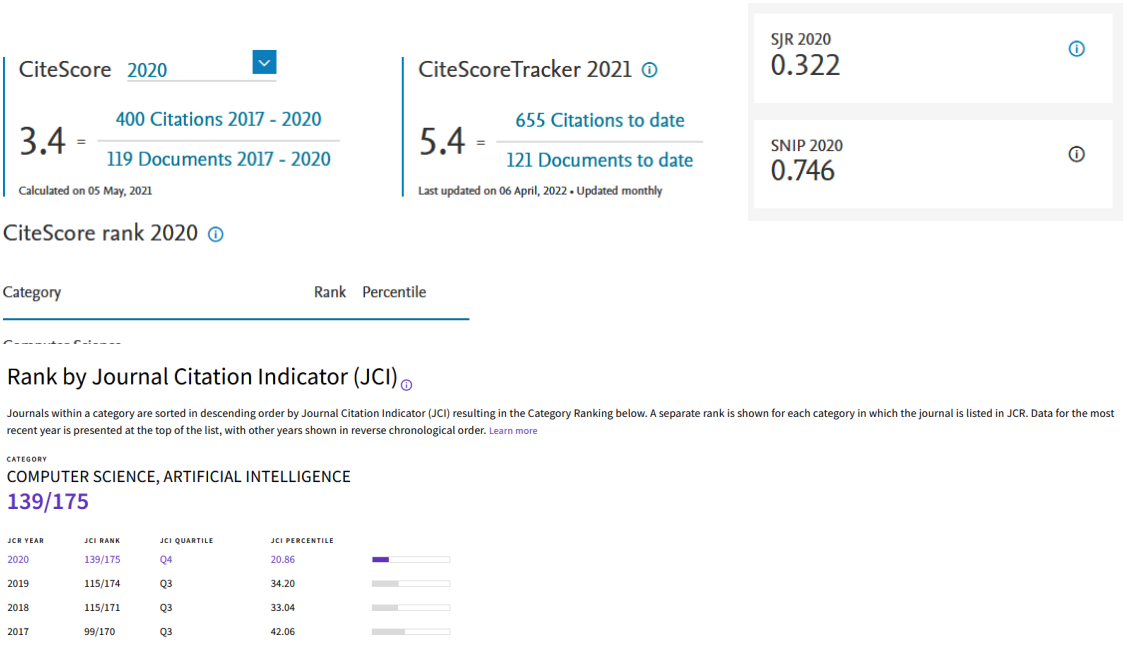
SJR 2020: 0.322

CiteScore rank 2020

- Computer Science, Artificial Intelligence. Ranking 113/227 (Q2).

Rank by Journal Citation Indicator (JCI)

- Computer Science, Artificial Intelligence. Ranking 139/175 (Q4).



Progress in Artificial Intelligence
https://doi.org/10.1007/s13748-020-00215-1

REGULAR PAPER

Fixed versus variable time window warehousing strategies in real time

Sergio Gil-Borrás[1] · Eduardo G. Pardo[2] · Antonio Alonso-Ayuso[2] · Abraham Duarte[2]

Received: 8 May 2020 / Accepted: 31 July 2020

Abstract
Warehousing includes many different regular activities such as receiving, batching, picking, packaging, and shipping goods. Several authors indicate that the picking operation might consume up to 55% of the total operational costs. In this paper, we deal with a subtask arising within the picking task in a warehouse, when the picking policy follows the order batching strategy (i.e., orders are grouped into batches before being collected) and orders are received online. Particularly, once the batches have been compiled it is necessary to determine the moment in the time when the picker starts collecting each batch. The waiting time of the picker before starting to collect the next available batch is usually known as time window. In this paper, we compare the performance of two different time window strategies: Fixed Time Window and Variable Time Window. Since those strategies cannot be tested in isolation, we have considered: two different batching algorithms (First Come First Served and a Greedy algorithm based on weight); one routing algorithm (S-Shape); and a greedy selection algorithm for choosing the next batch to collect based on the weight.

Keywords Time window · Fixed time window · Variable time window · Online order batching · Warehousing

1 Introduction

The picking operation in a warehouse consist of collecting all the items, demanded by the customers, which are stored in the warehouse. This operation might consume up to 55% of the total operational costs of the warehouse [34]. It is well documented that the picking is highly influenced by the layout of the warehouse, the storage policy, and the routing strategy followed by the pickers [27]. The picking operation can be divided into two main groups: strict order picking (each order is collected individually) and order batching (orders are grouped into batches before being collected).

In this paper, we focus our attention on the picking policy that follows an order batching strategy. When using this strategy, a family of related optimization problems (order batching problems) emerges. Order batching problems usually look for an efficient organization of the picking operation. The simplest version of this family of problems, usually known as Order Batching Problem (OBP) consist of minimizing the total time needed to collect a group of orders that have been received in a warehouse. This version of the OBP has been tackled by many authors in the literature: Albareda et al. in [1] compared different classical constructive heuristics (First Come First Served, Clarke and Wright [5], and Seed methods [6]) with a Variable Neighborhood Descent [23] metaheuristic over different sets of instances derived from real warehouses. Later, Henn et al. in [14] proposed the use of Iterated Local Search [32] and Ant Colony Optimization [18] to tackle the problem. Again, Henn et al. in [16] introduced the use of other two metaheuristics for the problem: Tabu Search [10] and Attribute-Based Hill Climbing [38]. An exact method based on a mixed-

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 and European Regional Development Fund, Grant Ref. P2018/TCS-4566; and by Programa Propio de I+D+i de la Universidad Politécnica de Madrid (Programa 466A).

Eduardo G. Pardo
eduardo.pardo@urjc.es

Sergio Gil-Borrás
sergio.gil@upm.es

Antonio Alonso-Ayuso
antonio.alonso@urjc.es

Abraham Duarte
abraham.duarte@urjc.es

[1] Departamento de Sistemas Informáticos, Universidad Politécnica de Madrid, Madrid, Spain

[2] Department of Computer Sciences, Universidad Rey Juan Carlos, Móstoles, Spain

Published online: 13 August 2020

integer linear programming model, and a new Iterated Local Search were introduced in [24,25]. More recently, Menéndez et al. in [20,21] proposed a new multi-start method combined with Variable Neighborhood Search which outperformed previous approaches in the state of the art.

Despite of the fact that the simplest version of the OBP is the one with a more extensive literature, other well-known variants of this family of problems have also been tackled in the state of the art. Each variant is characterized by either the optimization of a different objective function or the consideration of different constraints. Among the most outstanding ones, we can find the Min–Max Order Batching Problem, consisting of minimizing the maximum differences among the working-time of a group pickers [12,22,31,39]; or the Order Batching and Sequencing Problem which consists of minimizing the total tardiness of a group of orders with a due date associated to each order [3,13,15,19].

All the previous variants of the order batching family can be considered as static variants, since all the orders are available at the beginning of the process and, therefore, all the information is known beforehand. However, in this paper, we focus our attention on a subtask that occurs within the dynamic version of the OBP, usually known in the literature as Online Order Batching Problem (OOBP). In this variant, only a small part of the orders is available at the beginning of the picking process and, the rest of the orders are received online (i.e., they arrive to the warehouse at any time during a considered time horizon, once the picking process has already begun). In the version of the OOBP considered in this paper, each batch is collected by a single picker, who retrieves all the items belonging to the orders sorted in that batch. Additionally, an order cannot be splitted into different batches. Within this context it is possible to study different objective functions such as: the total time needed to collect all the orders (completion time); the time that pickers spend collecting the orders (picking/service time); the maximum/average time that an order remains in the system (turnover time); the balance of time among the pickers (workload balance); the delays in handling the orders (tardiness); etc.

As in the case of the OBP, the OOBP has also been tackled in the past using heuristic and metaheuristic methods. Particularly, S. Henn in [11] proposed an Iterated Local Search for the OOBP. Later, Pérez et al. in [26] used a method based on the Estimation Distribution Algorithm [17]. Then, Zhang et al. in [41] also introduced several heuristics for the OOBP, but this proposal was not compared with the previous ones. Finally, the most recent approaches have been proposed by Gil et al. in [7,9], where the authors proposed different methods based on the Variable Neighborhood Search metaheuristic, outperforming previous methods in the state of the art. Closely related, Rubrico et al. in [30] studied a multipicker variant of the OOBP by proposing a Steepest Descent Insertion and a Multi-Stage Rescheduling method. More recently, Gil et al. [8] also handled a multipicker version of the problem by proposing a Basic Variable Neighborhood Search for this variant.

In this paper, we study a subtask which occurs within the context of the OOBP, consisting of determining the time that a picker should wait before start collecting the next available batch. This time is usually known as time window in the literature of the OOBP. The rest of the article is organized as follows: in Sect. 2 we introduce the time window in the context of the OOBP; in Sect. 3 we describe the routing, batching and time window strategies used in the experimental comparison; in Sect. 4 we present the experiments performed; and finally, the conclusions and future works are compiled in Sect. 5.

2 Problem description

In the context of the OOBP it is necessary to study several static and dynamic factors in order to design efficient algorithms to handle the problem. Among the static ones we can find: the warehouse design (number of aisles, width of each aisle, number of blocks, number and position of the depots, etc.), or the distribution of the products in the warehouse (random distribution, ABC distribution, single/multiple locations for the products, etc.), among others. As far as the dynamic factors are concerned, we can find: the number of pickers, the number and size of orders arrived to the warehouse, the due date of some orders, among others.

In this paper, we focus our attention on the OOBP for rectangular-shaped warehouses composed by several parallel aisles and two cross aisles. In Fig. 1 we show an example of the structure of the warehouse considered. In this case, the warehouse is composed by 5 parallel aisles and 2 cross aisles (one at the front and another one at the back of the figure). Each parallel aisle is formed by 9 picking positions at each side of the aisle. The depot (i.e., the point where the routes start and finish) can be placed either at the left most corner or at the centre point of the front cross aisle. In the example warehouse of the Fig. 1, the depot is placed at the centre of the front cross aisle. The details of the set of instances used in our experiments (number of parallel aisles, position of the depot, distribution of the items, etc.) are described in Sect. 4.1.

Additionally, when tackling the OOBP it is necessary to face several subproblems that have been previously handled in the literature: determining how the orders are sorted into batches [7,20]; assigning each batch to a picker [8,11,22]; setting the sequence in which the batches are retrieved [13,19]; and designing the route that the pickers will follow within the warehouse [29,35]. However, not much attention has been paid to determining the time window strategy (i.e., the time that a picker waits before starting to collect the next available

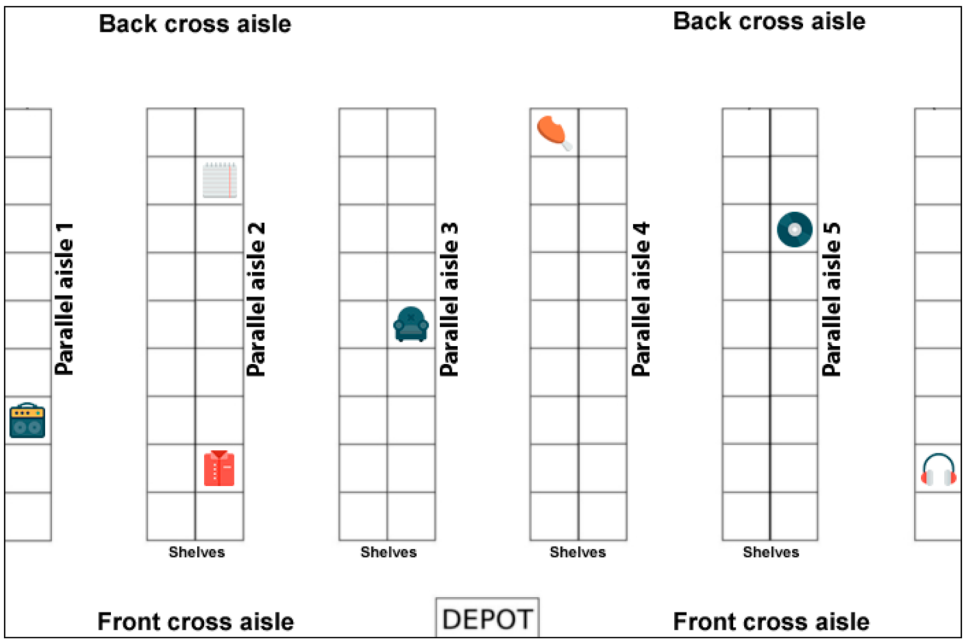


Fig. 1 Warehouse layout (adapted from [8])

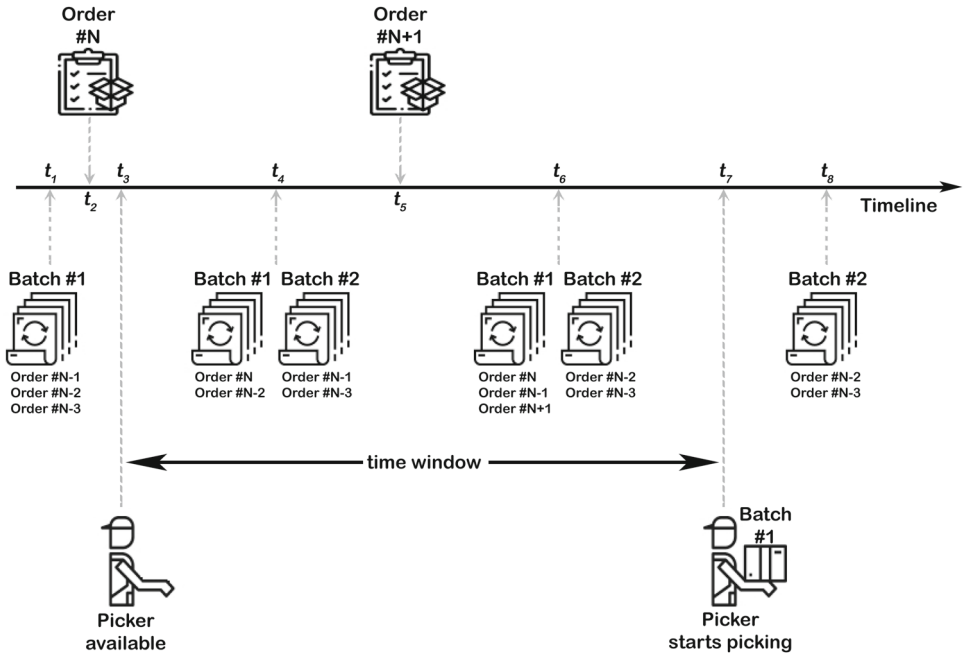


Fig. 2 Events in the timeline of a warehouse

batch). Since we focus our attention on this strategy, in the Fig. 2, we graphically introduce the concept of time window. In this figure we can observe the occurrence of several events at different moments t_i of the timeline considered. Particularly t_2 and t_5 represent moments in the time when a new order arrives to the system. Notice, that orders can arrive to the system at any time with independence of the rest of the tasks being performed in the warehouse (i.e., online arrivals). On the other hand, t_1, t_4, t_6 and t_8 represent moments in the time when a new solution has been found by the batching algorithm. Notice that the batching process is continuously been performed, trying to fit all the orders arrived to the system, not previously collected, into new batches. Finally, t_3 represents the moment in the time when the picker becomes available (i.e., it has already handled all the orders from the previous batch into the depot). Similarly, t_7 represents the moment in the time when a picker starts collecting the next available batch (i.e., Batch #1 in the figure). Given this context, the time window is defined as the difference in the time between the moments t_7 and t_3.

As we will show in the experimental section of this paper, we can affirm that the strategy used to determine the time

window has also a deep impact in the picking task. Generally speaking, there are two main groups of strategies to handle the time window subtask: Fixed Time Window (FTW) strategy and Variable Time Window (VTW) strategy. The FTW strategy establishes a fixed amount of time that a picker waits before starting to collect the next available batch. On the other hand, in the VTW the time that the picker waits varies through the time. In this sense there are many different VTW strategies. However, the most common one is based on the rate of orders that arrive to the system.

In the context of the OOBP, previous works in the state of the art mention the use of a Time Window strategy to increase the performance of their picking algorithms [4,33,40]. Also, there are references in the literature which use the time window as a part of their batching strategy [28,36]. More recent proposals include new algorithms to determine the time window within a particular context [2,11,36,41]. However, as far as we know, there are no specific comparisons among the time window strategies present in the literature of the OOBP.

This paper is focused on a preliminary comparison between the two main families of time window waiting strategies (FTW and VTW) in the context of the OOBP. With that aim we have considered one well-known routing strategy (S-Shape) and two different batching algorithms. Also, we study the performance of the strategies depending on the objective function reported. In this sense, we consider the picking time (the time that the picker spends collecting items in the warehouse without taking into consideration the waiting time) and the completion time (the total time needed to collect all the orders, including the waiting time).

3 Algorithms

In this section we describe the algorithms used in our experimental comparison. The section is divided considering the subtask of the OOBP tackled. Particularly, the two compared time window strategies are described in Sect. 3.4. However, in order to tackle the OOBP it is also necessary to establish a batching strategy (see Sect. 3.1 for the description of the two batching strategies used), a selection strategy (see Sect. 3.2), and a routing strategy (see Sect. 3.3). The objective is to measure the impact of the time window proposals in combination with the other algorithms for the OOBP.

3.1 Batching algorithm

Here, we present the two batching strategies selected for our experimental comparison. The batching algorithms used can be considered as two basic batching strategies very well-known in the literature of Order Batching problems.

- **First Come First Served (FCFS)** is a very simple batching algorithm which assigns the orders to the batches based on the instant of the time when each order arrives to the system. This is, the first order arriving to the system is assigned to the first available batch, the second order arriving to the system is tried to be assigned to the same batch. If it does not fit in it a new batch is created and the order is assigned to this new batch. The process is repeated by trying to assign orders to the last available batch or creating a new batch if it does not fit in the last available one.
- **Greedy algorithm based on weight** sorts all the orders available in the system (i.e., not collected yet) based on its weight, in the way that the heaviest order come first and the slightest is the last one. Then, it assigns the orders one by one to the batches, by trying first to include the order considered into an already existing batch (with the aim of completing the batch as much as possible) or creating a new batch if the order does not fit into any of the previous ones.

3.2 Selection algorithm

Once the batching algorithm has sorted the orders into batches it is necessary to determine which, among the available batches, will be the next one to be picked. This task might be very important depending on the objective function tackled. For instance, some variants of the OOBP consider a due date for the orders. In these cases, the algorithm which selects the next batch to be collected is a key component of the process. However, in this paper, we do not focus our attention on this task. Therefore, we have used a very simple selection algorithm which is based on the total weight of the orders assigned to each batch. In particular, we calculate, per batch, the sum of the weights of the orders assigned to that batch. Then, we select the heaviest batch as the next batch to collect. If there are more than one batch with the same maximum weight, to break the tie we select the batch which has assigned the shorter path to collect all the orders in that batch. Notice that the length of the path depends on the routing algorithm that is presented in Sect. 3.3.

3.3 Routing algorithm

In the context of the OOBP it is necessary to establish a route for collecting all the items within the orders assigned to the same batch. Notice that all items must be collected in the same route by a single picker. In this paper, we propose the use of a very well-known routing algorithm in the literature (**S-Shape**), in order to determine the path that the picker must follow. All routes start and end at the depot. The S-Shape algorithm establishes that the picker fully traverses each parallel aisle with an item to collect, using the front and

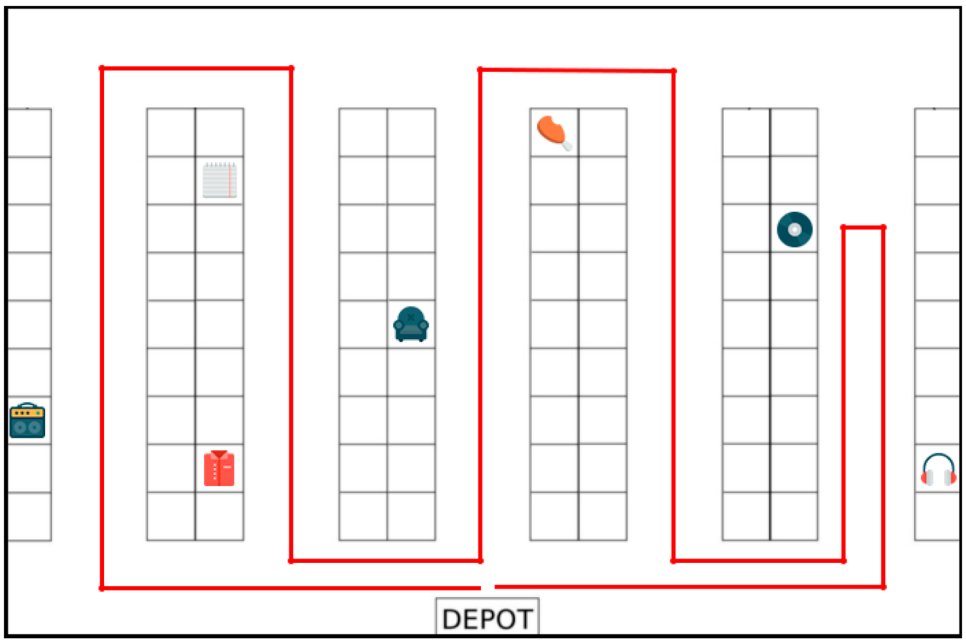


Fig. 3 S-Shape routing strategy (adapted from [8])

back cross aisles to change from one parallel aisle to another one. The first parallel aisle to be traversed will be the leftmost one with items to be collected. If the number of parallel aisles to be traversed is odd, then the picker performs a U-turn in the last parallel aisle in order to return to the front cross aisle, where the depot is placed. In Fig. 3 we show an example of a route designed using S-Shape for the warehouse introduced in Fig. 1.

3.4 Time window algorithm

The time window in the context of the OOBP is the time that a picker is waiting at the depot before starting to collect the next available batch. A naive time window strategy would be a "no-waiting policy". This policy indicates that the picker starts a new route as soon as he/she finishes the previous one and there are orders awaiting to be collected. However, a more elaborated strategy might result in a better performance. The main contribution of this paper is the comparison of two families of time window strategies together with two objective functions, two batching algorithms, and one routing algorithm. Next, we describe the time window strategies compared:

- **Fixed Time Window (FTW)** strategy consists of defining a fixed number of minutes that the picker must wait before starting a new route. In this paper, we have considered three different time horizons: 3, 6, and 12 min. Notice that if after the considered time horizon there is not at least one order available, the picker waits until a new order comes into the system.
- **Variable Time Window (VTW)** strategy varies the waiting time depending on the environmental conditions of the warehouse. In this case, the VTW strategy used is based on the arrival rate of orders. In particular, the picker waits until there are a minimum number of orders awaiting in the system before starting the picking. We have considered the cases of 4, 8, and 16 orders.

It is important to highlight that the most suitable time window strategy might depend on many different factors such as the routing and batching strategies used, the objective function considered, or the static conditions in the warehouse.

4 Experimental comparison

In this section we present the data used in our experiments and the computational results obtained with the algorithms described in Sect. 3. Since our objective is devoted to compare different time window strategies, for each experiment we report the performance of the considered time window algorithms: FTW, and VTW with several configurations. Particularly, FTW has been configured with 3, 6, and 12 waiting minutes. This means, that the picker waits that number of minutes before starting to collect the next available batch. Similarly, the VTW has been configured with 4, 8 and 16 orders, which means that the picker waits until there are at least that number of orders awaiting to be collected. Additionally, in order to set a reference baseline, we have also reported the performance of the no-waiting strategy (the picking of the next batch starts as soon as the picker becomes available).

Table 1 Comparison of Time Window strategies for the OOBP with picking time objective function, using the FCFS batching algorithm and the S-Shape routing algorithm

Picking time			
Method	Dev. (%)	#Best	Avg. (s)
No-waiting	4.91	1	27,934
FTW 3 Min.	4.91	1	27,934
FTW 6 Min.	4.91	1	27,934
FTW 12 Min.	4.04	1	27,764
VTW 4 orders	0.96	7	27,243
VTW 8 orders	0.33	10	27,104
VTW 16 orders	0.04	14	27,057

Table 2 Comparison of Time Window strategies for the OOBP with completion time objective function, using the FCFS batching algorithm and the S-Shape routing algorithm

Completion time			
Method	Dev. (%)	#Best	Avg. (s)
No-waiting	0.53	6	28,194
FTW 3 Min.	0.53	5	28,194
FTW 6 Min.	0.53	4	28,194
FTW 12 Min.	0.23	8	28,127
VTW 4 orders	1.38	3	28,430
VTW 8 orders	4.69	0	29,189
VTW 16 orders	10.93	0	30,499

The time horizon considered for the arrival of orders has been set to 4 h and the distribution of the arrival of the orders follows a Poisson point process [37]. The Poisson point process uses the 4 h as an input parameter to distribute the instants in the time for the arrival of each considered order in the instance, which results in an approximate total time of 4 h. Notice, that the picker will only collect the orders scheduled by this distribution. However, the picking process might take longer than the 4 h considered as the time horizon.

Experiments are organized in two different sections (depending on the batching strategy followed) and, per section, we include two tables depending on the objective function considered. For all tables we report the deviation to the best solution found in the experiment (Dev. (%)), the number of best solutions found in the experiment (#Best), and the average of the objective function (Avg.(s)) measured in seconds. Additionally, we have included, in each section, a figure which graphically compares the quality of the solutions obtained with respect to each objective function. Finally, we have performed a set of statistical tests in order to determine if the differences found in the results are statistically significant.

All the experiments were run in an Intel (R) Core (TM) 2 Quad CPU Q6600 2.4 GHz computer, with 4 GB DDR2 RAM memory. The operating system used was Ubuntu 18.04.1 64 bit LTS, and all the codes were developed in Java 8.

4.1 Instances

The data set used in our experiments is composed by 16 diverse instances derived from a larger dataset originally proposed in [11]. The size of the selected instances ranges from 40 to 100 orders. The one-block, rectangular-shaped warehouse considered is similar to the one presented in Sect. 2. In this case, all the instances are referred to a warehouse with 10 parallel aisles, and with the depot placed at the centre of the front cross aisle. The products are organized following an ABC distribution.

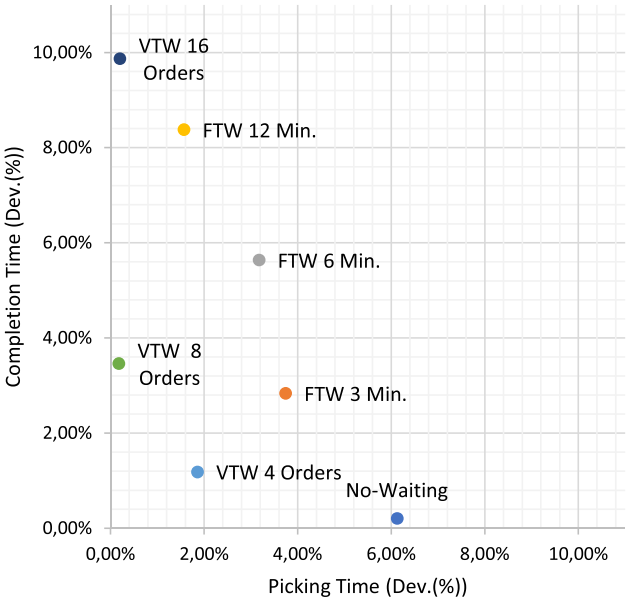


Fig. 4 Comparison of the performance in Dev. (%) of the different variants of the algorithms configured with the FCFS batching strategy for the two objective functions

Table 3 Friedman rank test for each objective function when using the FCFS batching method

FCFS			
Picking time		Completion time	
Method	Rank val	Method	Rank val.
VTW 16 orders	1.91	FTW 12 Min.	2.72
VTW 8 orders	2.28	FTW 6 Min.	2.88
VTW 4 orders	2.63	No-waiting	3.03
FTW 12 Min.	4.69	FTW 3 Min.	3.13
FTW 6 Min.	5.50	VTW 4 orders	3.63
FTW 3 Min.	5.50	VTW 8 orders	5.72
No-waiting	5.50	VTW 16 orders	6.91

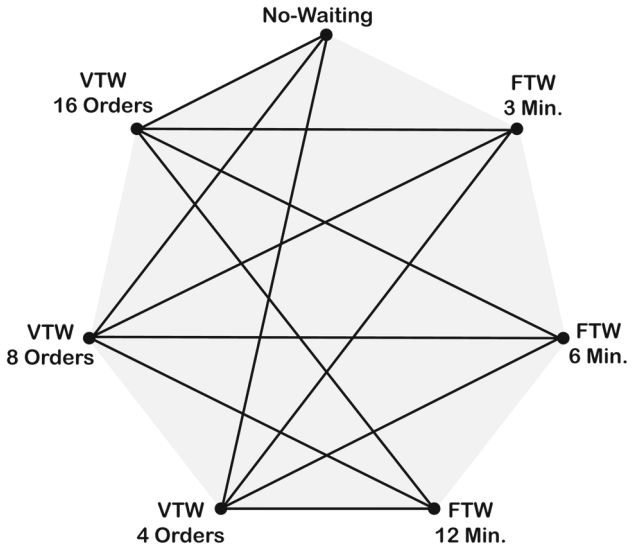


Fig. 5 Representation of the Wilcoxon test comparison by pairs of methods, for the picking time objective function and FCFS batching strategy

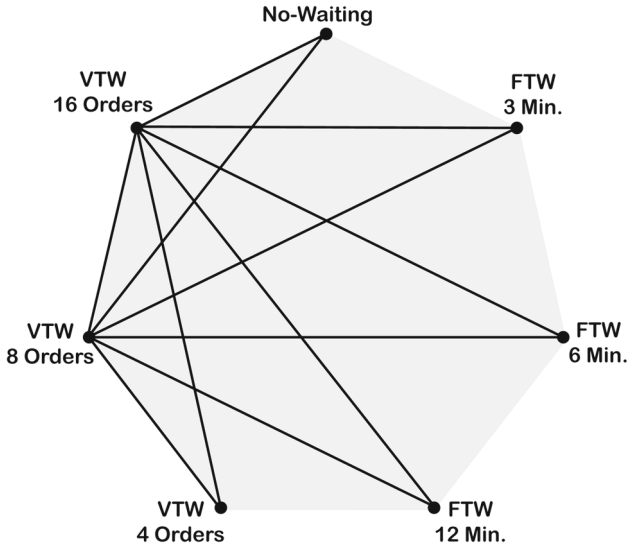


Fig. 6 Representation of the Wilcoxon test comparison by pairs of methods, for the completion time objective function and FCFS batching strategy

4.2 Comparison of time window strategies using FCFS batching algorithm

We compare the performance of the algorithms designed for the OOBP configured with FCFS as the batching method, S-Shape as the routing method and the two Time Window strategies configured with 3 different parameters each. In Tables 1 and 2 we report, respectively, the results obtained for the "picking time" and the "completion time" objective functions. For the instances considered in these experiments, we observe, in Table 1, that all the strategies based on VTW outperform the results by the FTW ones. Particularly, the best strategy for the picking time objective function is the VTW with 16 orders in terms of deviation to best solution found and in the number of best solutions. On the other hand, in Table 2 we observe the results for the completion time objective function. In this case, all the strategies based on the FTW approach outperform the VTW ones. In particular, the FTW with 12 min is the best strategy closely followed by the no-waiting strategy and the other FTW configurations.

Since the best performance for each objective function has been reached by a different Time Window algorithm, in Fig. 4 we compare the performance of the different configurations of the algorithms with respect to both objective functions. Observing the figure, we find that the performance of some configurations is very good in one of the objective functions and very poor in the other. However, the algorithm which includes VTW with 4 orders finds a balance between the two objective functions compared. Notice that, as it is shown in the figure, there are no differences among the no-waiting strategy, the FTW with 3 min strategy and the FTW with 6 min, when considering both objective functions.

In order to confirm if the differences found among the methods in Tables 1 and 2 are statistically significant, we have conducted a set of statistical tests. First, we have compared all the methods using the Friedman Rank test per table. In both cases, the obtained p values of 0.000 indicate that the methods differ among them, when considering all of them together. Additionally, in Table 3 we report the rank values obtained for each method in both: the picking time and the completion time. As we can observe some of the methods differ greatly, but others achieve the same or almost the same rank value. To complement this test, we have compared each pair of methods individually, with the Wilcoxon signed rank test, in order to identify differences between the objective function values of the best solutions found by each method with respect to the rest of them. In Figs. 5 and 6 we report a graphical representation of the differences by pairs, among the 7 methods compared, for the picking time and the completion time objective functions, respectively. In both figures, a black line between each pair of methods indicates that there are statistically significant differences between them, while the absence of a link between a pair of methods indicates that there are no differences between them.

4.3 Comparison of time window strategies using Greedy batching algorithm

Similarly, in this section, we compare the performance of the algorithms designed for the OOBP configured with the Greedy as the batching method, S-Shape as the routing method and different configurations of the Time Window strategies. In Tables 4 and 5 we respectively report the results obtained for the "picking time" and "completion time"

Table 4 Comparison of Time Window strategies for the OOBP with picking time objective function, using the Greedy batching algorithm and the S-Shape routing algorithm

Picking time			
Method	Dev. (%)	#Best	Avg. (s)
No-waiting	6.13	0	26,407
FTW 3 Min.	3.74	0	26,042
FTW 6 Min.	3.17	0	25,955
FTW 12 Min.	1.57	2	25,675
VTW 4 orders	1.86	2	25,668
VTW 8 orders	0.18	4	25,375
VTW 16 orders	0.20	9	25,383

Table 5 Comparison of Time Window strategies for the OOBP with completion time objective function, using the Greedy batching algorithm and the S-Shape routing algorithm

Completion time			
Method	Dev. (%)	#Best	Avg. (s)
No-waiting	0.21	9	26,667
FTW 3 Min.	2.84	0	27,248
FTW 6 Min.	5.64	0	27,832
FTW 12 Min.	8.38	0	28,380
VTW 4 orders	1.19	6	26,892
VTW 8 orders	3.46	1	27,450
VTW 16 orders	9.87	0	28,826

objective functions. For the instances considered in these experiments, when the objective function is the picking time, we observe that the best strategy is the VTW with 8 orders closely followed by the VTW with 16 orders (see Table 4). On the other hand, in Table 5 we observe that, when the objective function is the completion time, the best strategy is the no-waiting one.

Again, in Fig. 7 we compare the performance of the methods over the two objective functions simultaneously. In this case, the VTW with 4 orders and the VTW with 8 orders are the methods which find a better balance between both objective functions.

As in the case of the FCFS batching method, in order to confirm if the differences found among the methods in Tables 4 and 5 are statistically significant, we have conducted a Friedman Rank test per table. In both cases, the obtained p values of 0.000 indicate that the methods differ among them, when considering all of them together. However, in Table 6 we report the rank values obtained for each method in both: the picking time and the completion time for the Greedy batching method. As it was the case of the FCFS batching method, we can observe some of the methods differ greatly, but others achieve the same or almost the same rank

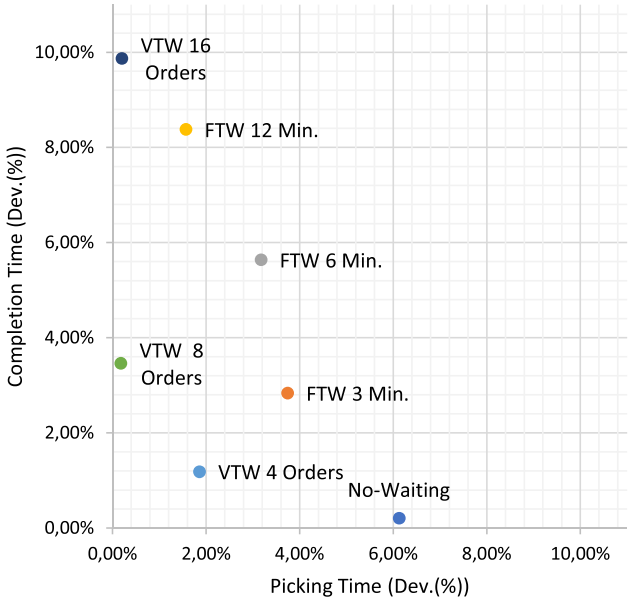


Fig. 7 Comparison of the performance in Dev.(%) of the different variants of the algorithms configured with the Greedy batching strategy for the two objective functions

Table 6 Friedman rank test for each objective function when using the Greedy batching method

Greedy			
Picking time		Completion time	
Method	Rank val.	Method	Rank val.
VTW 16 orders	2.06	No-waiting	1.50
VTW 8 orders	2.09	VTW 4 orders	2.19
VTW 4 orders	3.38	FTW 3 Min.	3.23
FTW 12 Min.	3.38	VTW 8 orders	3.97
FTW 6 Min.	5.13	FTW 6 Min.	4.75
FTW 3 Min.	5.31	FTW 12 Min.	5.69
No-waiting	6.66	VTW 16 orders	6.66

value. To complement this test, we have also compared each pair of methods individually, with the Wilcoxon signed rank test, in order to identify differences between the objective function values of the best solutions found by each method with respect to the rest of them. In Figs. 8 and 9 we report a graphical representation of the differences by pairs among the 7 methods compared, for the picking time and the completion time objective functions, respectively. In both figures, a black line between each pair of methods indicates that there are statistically significant differences between them, while the absence of a link between a pair of methods indicates that there are no differences between them.

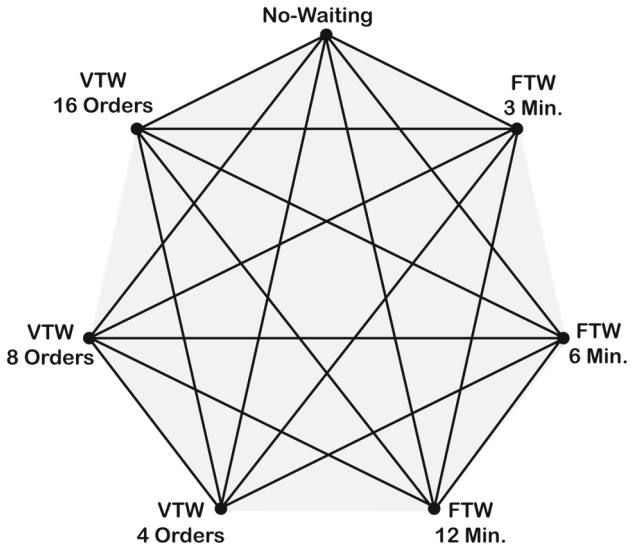


Fig. 8 Representation of the Wilcoxon test comparison by pairs of methods, for the picking time objective function and and the Greedy batching strategy

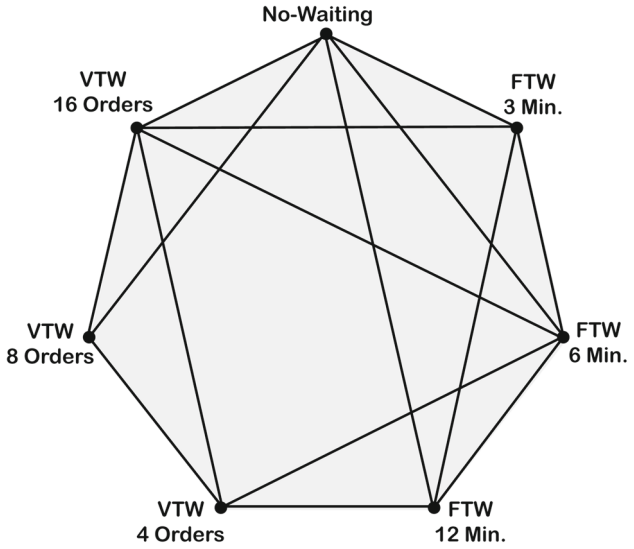


Fig. 9 Representation of the Wilcoxon test comparison by pairs of methods, for the completion time objective function and the Greedy batching strategy

5 Conclusions and future work

In this paper, we have compared two families of time window strategies in the context of the Online Order Batching Problem. The compared strategies have been evaluated together with two different batching algorithms and with two different objective functions. However, we have only considered one routing algorithm (S-Shape) for all the methods compared.

Our main conclusion is that the time window strategy highly impacts in the quality of the solutions obtained in the context of the OOBP. Also, we observed that the most suitable time window strategy might depend on the objective function evaluated, and on the routing and batching algorithms considered. However, the influence of the time window when using the FCFS batching algorithm seems to be smaller than in the case of using a more elaborated one. Furthermore, when more than one objective function is optimized at the same time, some time window strategies find a balance in the quality of the solution.

Given the previous findings, we suggest to enlarge this study by performing a more elaborated comparison. In particular, the most outstanding time window algorithms in the state of the art should be considered and paired with different batching and routing strategies. Also, several warehouse configurations could be evaluated by enlarging the data set used in our experiments.

References

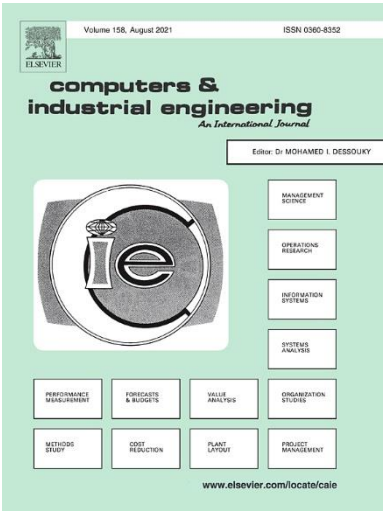
1. Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., De Blas, C.S.: Variable neighborhood search for order batching in a warehouse. Asia Pac. J. Oper. Res. **26**(5), 655–683 (2009)
2. Bukchin, Y., Khmelnitsky, E., Yakuel, P.: Optimizing a dynamic order-picking process. Eur. J. Oper. Res. **219**(2), 335–346 (2012)
3. Cano, J.A., Correa-Espinal, A.A., Gómez-Montoya, R.A.: Mathematical programming modeling for joint order batching, sequencing and picker routing problems in manual order picking systems. J. King Saud Univ. Eng. Sci. **32**(3), 219–228 (2020)
4. Chew, E.P., Tang, L.C.: Travel time analysis for general item location assignment in a rectangular warehouse. Eur. J. Oper. Res. **112**(3), 582–597 (1999)
5. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. **12**(4), 568–581 (1964)
6. Gibson, D.R., Sharp, G.P.: Order batching procedures. Eur. J. Oper. Res. **58**(1), 57–67 (1992)
7. Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte, A.: New VNS variants for the online order batching problem. In: Sifaleras, A., Salhi, S., Brimberg, J. (eds.) Variable Neighborhood Search. ICVNS 2018. Lecture Notes in Computer Science, vol. 11328, pp. 89–100. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15843-9_8
8. Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte, A.: Basic VNS for a variant of the online order batching problem. In: Benmansour, R., Sifaleras, A., Mladenović, N. (eds.) Variable Neighborhood Search. ICVNS 2019. Lecture Notes in Computer Science, vol. 12010, pp. 17–36. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44932-2_2
9. Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte, A.: Grasp with variable neighborhood descent for the online order batching problem. J. Global Optim. (2020). https://doi.org/10.1007/s10898-020-00910-2
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
11. Henn, S.: Algorithms for on-line order batching in an order picking warehouse. Comput. Oper. Res. **39**(11), 2549–2563 (2012)
12. Henn, S.: Variable neighborhood search for the order batching and sequencing problem with multiple pickers. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management (2012)

13. Henn, S.: Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. Flex. Serv. Manuf. J. **27**(1), 86–114 (2015)
14. Henn, S., Koch, S., Doerner, K.F., Strauss, C., Wäscher, G.: Metaheuristics for the order batching problem in manual order picking systems. Bus. Res. **3**(1), 82–105 (2010)
15. Henn, S., Schmid, V.: Metaheuristics for order batching and sequencing in manual order picking systems. Comput. Ind. Eng. **66**(2), 338–351 (2013)
16. Henn, S., Wäscher, G.: Tabu search heuristics for the order batching problem in manual order picking systems. Eur. J. Oper. Res. **222**(3), 484–494 (2012)
17. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, vol. 2. Springer, Berlin (2001)
18. López-Ibáñez, M., Stützle, T., Dorigo, M.: Ant colony optimization: a component-wise overview. In: Martí, R., Panos, P., Resende, M. (eds.) Handbook of Heuristics, pp. 371–407. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-07153-4_21-1
19. Menéndez, B., Bustillo, M., Pardo, E.G., Duarte, A.: General variable neighborhood search for the order batching and sequencing problem. Eur. J. Oper. Res. **263**(1), 82–93 (2017)
20. Menéndez, B., Pardo, E.G., Alonso-Ayuso, A., Molina, E., Duarte, A.: Variable neighborhood search strategies for the order batching problem. Comput. Oper. Res. **78**, 500–512 (2017)
21. Menéndez, B., Pardo, E.G., Duarte, A., Alonso-Ayuso, A., Molina, E.: General variable neighborhood search applied to the picking process in a warehouse. Electron. Notes Discrete Math. **47**, 77–84 (2015)
22. Menéndez, B., Pardo, E.G., Sánchez-Oro, J., Duarte, A.: Parallel variable neighborhood search for the min–max order batching problem. Int. Trans. Oper. Res. **24**(3), 635–662 (2017)
23. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
24. Öncan, T.: MILP formulations and an iterated local search algorithm with Tabu thresholding for the order batching problem. Eur. J. Oper. Res. **243**(1), 142–155 (2015)
25. Öncan, T., Cağırıcı, M.: MILP formulations for the order batching problem in low-level picker-to-part warehouse systems. IFAC Proc. Vol. **46**(9), 471–476 (2013)
26. Pérez-Rodríguez, R., Hernández-Aguirre, A., Jöns, S.: A continuous estimation of distribution algorithm for the online order-batching problem. Int. J. Adv. Manuf. Technol. **79**(1), 569–588 (2015)
27. Petersen, C.G.: An evaluation of order picking routeing policies. Int. J. Oper. Prod. Manag. **17**(11), 1098–1111 (1997)
28. Quinn, E.B.: Simulation of order processing waves. Mater. Handling Focus **83**, 5 (1983)
29. Roodbergen, K.J., Koster, R.D.: Routing methods for warehouses with multiple cross aisles. Int. J. Prod. Res. **39**(9), 1865–1883 (2001)
30. Rubrico, J.I.U., Higashi, T., Tamura, H., Ota, J.: Online rescheduling of multiple picking agents for warehouse management. Robot. Comput. Integr. Manuf. **27**(1), 62–71 (2011)
31. Scholz, A., Schubert, D., Wäscher, G.: Order picking with multiple pickers and due dates-simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. Eur. J. Oper. Res. **263**(2), 461–478 (2017)
32. Stützle, T., Ruiz, R.: Iterated local search. In: Martí, R., Panos, P., Resende, M. (eds.) Handbook of Heuristics, pp. 579–605. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-07153-4_8-1
33. Tang, L.C., Chew, E.P.: Order picking systems: batching and storage assignment strategies. Comput. Ind. Eng. **33**(3), 817–820 (1997). Selected Papers from the Proceedings of 1996 ICC&IC
34. Tompkins, J.A., White, J.A., Bozer, Y.A., Tanchoco, J.M.A.: Facilities Planning. Wiley, Chichester (2010)
35. Valle, C.A., Beasley, J.E., da Cunha, A.S.: Optimally solving the joint order batching and picker routing problem. Eur. J. Oper. Res. **262**(3), 817–834 (2017)
36. Van Nieuwenhuyse, I., de Koster, R.B.M.: Evaluating order throughput time in 2-block warehouses with time window batching. Int. J. Prod. Econ. **121**(2), 654–664 (2009)
37. von Bortkiewicz, L.: Das Gesetz der kleinen Zahlen. B.G. Teubner, Berlin (1898)
38. Whittley, I.M., Smith, G.D.: The attribute based hill climber. J. Math. Model. Algorithms **3**(2), 167–178 (2004)
39. Yokota, T.: Min-max-strategy-based optimum co-operative picking with AGVS in warehouse. In: 2019 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 236–242. IEEE (2019)
40. Yu, M., De Koster, R.B.M.: The impact of order batching and picking area zoning on order picking system performance. Eur. J. Oper. Res. **198**(2), 480–490 (2009)
41. Zhang, J., Wang, X., Chan, F.T.S., Ruan, J.: On-line order batching and sequencing problem with multiple pickers: a hybrid rule-based algorithm. Appl. Math. Model. **45**(Supplement C), 271–284 (2017)

6.2 A comparative study of the influence of the time-window strategy in online order batching problems

Gil-Borrás, S., Pardo, E. G., Jiménez, E. Kenneth, S. (2022). A comparative study of the influence of the time-window strategy in online order batching problems. Computers & Industrial Engineering, Article in 2nd review

Published in "**Computers & Industrial Engineering**"



ISSN: 0360-8352 / 1879-0550

Article under review (2nd Round)

Impact factor (JCR 2020): 5.431

Journal Citation Indicator (JCI): 1.18

Rank by Journal Impact Factor

- Computer science, Interdisciplinary applications.
  Ranking 21/111 (Q1).
- Engineering, Industrial.
  Ranking 14/49 (Q2).

Rank by Journal Citation Indicator (JCI)

- Computer science, Interdisciplinary applications.
  Ranking 40/142 (Q2).
- Engineering, Industrial.
  Ranking 13/62 (Q1).

Rank by Journal Impact Factor

Journals within a category are sorted in descending order by Journal Impact Factor (JIF) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

EDITION
Science Citation Index Expanded (SCIE)

CATEGORY
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS
21/111

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	21/111	Q1	81.53
2019	18/109	Q1	83.94
2018	24/106	Q1	77.83
2017	22/105	Q1	79.52
2016	28/105	Q2	73.81

EDITION
Science Citation Index Expanded (SCIE)

CATEGORY
ENGINEERING, INDUSTRIAL
14/49

JCR YEAR	JIF RANK	JIF QUARTILE	JIF PERCENTILE
2020	14/49	Q2	72.45
2019	10/48	Q1	80.21
2018	11/46	Q1	77.17
2017	9/47	Q1	81.91
2016	9/44	Q1	80.68

Rank by Journal Citation Indicator (JCI)

Journals within a category are sorted in descending order by Journal Citation Indicator (JCI) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

CATEGORY
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS
40/142

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	40/142	Q2	72.18
2019	41/140	Q2	71.07
2018	39/138	Q2	72.10
2017	38/136	Q2	72.43

CATEGORY
ENGINEERING, INDUSTRIAL
13/62

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2020	13/62	Q1	79.84
2019	12/62	Q1	81.45
2018	13/60	Q1	79.17
2017	14/57	Q1	76.32

Computers & Industrial Engineering

A comparative study of the influence of the time-window strategy in the Online Order Batching Problem

--Manuscript Draft--

Manuscript Number:	CAIE-D-22-00906R1
Article Type:	Research Paper
Keywords:	Online Order Batching Problem; Fixed time-window; Variable time-window; Warehousing; Supply chain
Corresponding Author:	Eduardo G. Pardo, Ph.D. Universidad Rey Juan Carlos Móstoles, SPAIN
First Author:	Sergio Gil-Borrás
Order of Authors:	Sergio Gil-Borrás
	Kenneth Sörensen
	Ernesto Jiménez
	Eduardo G. Pardo, Ph.D.
Abstract:	The Online Order Batching Problem (OOBP) consists of determining an efficient picking operation of the orders arriving at a warehouse, during a specific time horizon. It includes handling several subproblems, such as batching, waiting, assigning, or routing, among others. In this paper, we tackle the waiting task which consists of making a decision on whether to start picking (reducing idle time of the pickers) or to wait for more orders (allowing for a more efficient batch configuration). The waiting problem is also known as the determination of the Time Window, and it has received far less attention than other decision problems in the order picking context, such as batching and routing. We demonstrate that this lack of attention is unwarranted, and that the method used to determine the time window can have a significant impact on the overall performance of the warehouse. To that aim, we first show that the time-window strategy is independent of the methods used to solve the other subproblems of the OOBP such as batching and routing. Furthermore, we propose two new time-window methods for determining the time window and then, we show the influence of the proposed methods and the methods from the literature on the objective functions of the OOBP, when varying the number of pickers and the number of orders to be collected. Furthermore, two new time-window methods are proposed in

A comparative study of the influence of the time-window strategy in the Online Order Batching Problem

Abstract

The Online Order Batching Problem (OOBP) consists of determining an efficient picking operation of the orders arriving at a warehouse, during a specific time horizon. It includes handling several subproblems, such as batching, waiting, assigning, or routing, among others. In this paper, we tackle the waiting task which consists of making a decision on whether to start picking (reducing idle time of the pickers) or to wait for more orders (allowing for a more efficient batch configuration). The waiting problem is also known as the determination of the Time Window, and it has received far less attention than other decision problems in the order picking context, such as batching and routing. We demonstrate that this lack of attention is unwarranted, and that the method used to determine the time window can have a significant impact on the overall performance of the warehouse. To that aim, we first show that the time-window strategy is independent of the methods used to solve the other subproblems of the OOBP such as batching and routing. Furthermore, we propose two new time-window methods for determining the time window and then, we show the influence of the proposed methods and the methods from the literature on the objective functions of the OOBP , when varying the number of pickers and the number of orders to be collected. Furthermore, two new time-window methods are proposed in this work that improve in various scenarios the methods previously presented in the state of the art. All conclusions have been experimentally validated on reference instances previously published in the context of the OOBP.

Keywords: Online Order Batching Problem, Fixed time-window, Variable time-window, Warehousing, Supply chain

1. Introduction

Logistics in a warehouse encompasses a large number of activities. Among them, the collection of orders is one of the most important, due to the high cost associated with this operation compared to the rest of the processes. The operational costs of order picking has been the target for may researchers during the years. Back to the eighties / nineties, it was estimated that the operational costs could represent up to 60% of total costs within a warehouse (Drury, 1988; Coyle et al., 1996). More recent approaches indicated that labor costs related to the picking process consume about 50-60% of all labor activities in the warehouse (Gademann and Velde, 2005). Although the introduction of technology in the warehouses has partially reduced the costs associated with picking, it is still identified as the most important operation activity in achieving an efficient warehouse management Shah et al. (2017); Rushton et al. (2022).

This paper focuses on warehouse order picking systems, which follow the batch order picking policy (i.e., orders are grouped in batches before being collected and orders in the same batch are collected in the same picking tour) . In this type of system, there are numerous factors that significantly influence the performance of the collection process. In Petersen (1997), the authors identified the following factors: The layout of the warehouse, which consists of the shape, number of blocks, number of aisles, number of picking positions, etc.; the routing policy, which consists of determining the route that pickers follow within the warehouse to collect the orders; the sorting policy, which consists of determining when and how the picked products are separated in orders; the storage strategy, which determines where to store each type of product; the batching method, which determines how the products are grouped in batches prior to being collected.

Our article focuses on another additional factor, named Time Window (TW), not considered by Petersen, which also has a significant influence on the picking time (i.e., the time that pickers need to perform the picking task) and the completion time (i.e., the picking time together with the waiting time) (Gil-Borrás et al., 2020a) . The TW, also known as waiting time in this context, is defined as the time that a picker waits idle in the depot, before starting a new picking route. This factor is studied as an additional task that needs to be determined in the context of problems using a batch collection policy. Determining the waiting time is especially relevant when the arrival of orders is dynamic (i.e., online), although it could make sense also in multi-

picker contexts, when all orders are known before starting the pickup (i.e., offline) but there are potential deadlock situations, such as several pickers trying to access simultaneously to a picking position, to a corridor, or just to the depot.

This work is motivated by the lack of previous works related to determining the time window in the context of the Online Order Batching Problem (OOBP). Despite the fact that there are works in the literature where the time window is taken into consideration, it is only studied by applying a specific algorithm / function to determine it. However, there is no previous paper focusing on the influence of this task on objective functions related to the OOBP, nor comparing the most common existent time-window strategies for different scenarios.

This work can be considered as an extension of a previous work presented in Gil-Borrás et al. (2020a). In that work, the authors performed a preliminary study which compared the two main families of time-window strategies: Fixed Time Window and Variable Time Window. However, the authors only considered some basic heuristic strategies for the task. In this article, we focus on studying, in a detailed way, the influence of the time window on the value of the objective function , in the context of OOBP, depending on different factors such as: Batching, routing, objective function, or congestion in the arrival of orders. Also, we compile and compare the most outstanding strategies for determining the time window in each of the compared scenarios.

The main contributions of this work are: First, a chronological review of the evolution of the concept of time window related to the OOBP. Second, a compilation and classification of the most relevant time-window methods in the literature. Third, a detailed empirical study on the influence of the time window in several objective functions defined for the OOBP. Fourth, the proposal of two new Variable Time Window methods for determining the time window. And fifth, an extensive experimental analysis about the performance of each method in different scenarios.

The rest of the article is organized as follows. In Section 2 we describe the OOBP. In Section 3, we present the state of the art related to the time window. Next, in Section 4, we describe the methods to determine the time window found in the state of the art, and the two new methods proposed in this article. In Section 5, we describe the comparative study carried out. In this section, we also review the algorithms for the rest of the tasks that need to be handled in the context of OOBP. In Section 6, we present the experi-

ments of this paper. Finally, in Section 7, some conclusions and pointers for future research are presented.

2. Online Order Batching Problem

The Online Order Batching Problem is a dynamic optimization problem in which orders continually arrive at a warehouse during the processing time. In this way, not all orders to be processed nor their arrival times are known beforehand. In this context, orders are grouped into batches prior to be picked. The batch collection process can be carried out by a single picker or by several pickers. To solve the OOBP, it is necessary to tackle different tasks / subproblems. First, the orders that reach the system are grouped into batches for later collection. This process of grouping orders into batches is known as *batching*. Once the batches have been generated, a decision must be made on whether it is more convenient to start collecting any of the batches or to wait for new orders to arrive. This process is known as *determining the time window*. Although at first glance it may seem unnatural (even inefficient) to wait, new orders may arrive during the time window, which potentially could improve the distribution of orders in the batches already created. The time window can be a fixed amount of time or a variable one (i.e., a different waiting time for each picker on each new route). Furthermore, it is necessary to decide which batch, among all generated ones, is going to be collected next. This process is known as *selecting*. Notice that the term *selecting* is used when only the next batch to collect is chosen, as it is customary in dynamic / online environments, since the conditions might change in the future. On the other hand, the term *sequencing* is used for the same task when all the conformed batches are sorted to determine the order in which they will be picked, as it is customary in static / offline environments. Finally, once the batch and the time for starting the route are decided, it is necessary to determine the route that the picker should follow through the warehouse to perform the picking. This process is known as *routing*. Sometimes, there is an additional process consisting in *sorting* the products collected on the same picking route. This is due to the fact that products belonging to different orders are collected together and they might be placed in the same basket during picking. Therefore, a sorting process must be performed afterward to separate the products into different orders. However, we do not study the influence of this task in our work.

Next, we mathematically define the OOBP. To that aim, in Table 1, we

Parameters		
n	$\rightarrow$	Number of customer orders received in the system
m	$\rightarrow$	Upper bound of the number of batches (a straightforward value is $m = n$).
l	$\rightarrow$	Number of order pickers.
$v_{routing}$	$\rightarrow$	Routing velocity: number of length units that the picker can traverse in the warehouse per unit of time.
$v_{extraction}$	$\rightarrow$	Number of items that the picker can search and pick per time unit.
t_{setup}	$\rightarrow$	Time that the picker needs to initiate a new route with a new order list and end the route let the collected orders in the depot.
w_i	$\rightarrow$	Number of items of order o_i for $1 \leq i \leq n$.
W	$\rightarrow$	Maximum number of articles that can be included in a batch (device capacity).
ar_i	$\rightarrow$	Arrival time of order i for $1 \leq i \leq n$.

Variables		
st_j	$\rightarrow$	Start time of batch j for $1 \leq j \leq m$.
x_{ji}	$\rightarrow$	$\begin{cases} 1, & \text{if order } o_i \text{ is assigned to batch } b_j, \\ & for\ 1 \leq i \leq n,\ and\ 1 \leq j \leq m. \\ 0, & \text{otherwise.} \end{cases}$
y_{jk}	$\rightarrow$	$\begin{cases} 1, & \text{if picker } p_k \text{ is assigned to batch } b_j, \\ & for\ 1 \leq k \leq l,\ and\ 1 \leq j \leq m. \\ 0, & \text{otherwise.} \end{cases}$

Table 1: Parameters and variables for the General OOBP.

first present the parameters and variables used in the model. Then, we define the OOBP considering the minimization of the picking time (i.e., the sum of picking time of all pickers) as an objective function as follows:

$$\min \sum_{j=1}^{m} T_{service}(b_j). \tag{1}$$

subject to,

$$\sum_{j=1}^{m} x_{ji} = 1, \quad \forall\, i \in \{1, \ldots, n\}. \tag{2}$$

$$\sum_{k=1}^{l} y_{jk} = 1, \quad \forall\, j \in \{1, \ldots, m\}. \tag{3}$$

$$\sum_{i=1}^{n} w_i * x_{ji} \leq W, \quad \forall\, j \in \{1, \ldots, m\}. \tag{4}$$

$$st_j \geq \min_{k \in \{1,\ldots,l\}} \max_{s \in \{1,\ldots,j-1\}} y_{sk} * \big(st_s + T_{service}(b_s)\big), \quad \forall\, j \in \{2, \ldots, m\}. \tag{5}$$

$$st_j \geq st_{j-1}, \quad \forall\, j \in \{2, \ldots, m\}. \tag{6}$$

$$st_j \geq ar_i * x_{ji}, \quad \forall\, i \in \{1, \ldots, n\}, \text{ and } \forall\, j \in \{1, \ldots, m\}. \tag{7}$$

$$st_j \geq \texttt{tw}(), \quad \forall\, j \in \{1, \ldots, m\}. \tag{8}$$

$$st_j \geq 0, \quad \forall j \in \{1, \ldots, m\}. \tag{9}$$

$$x_{ji} \in \{0, 1\}, \quad \forall\, j \in \{1, \ldots, m\} \text{ and } \forall\, i \in \{1, \ldots, n\}. \tag{10}$$

$$y_{jk} \in \{0, 1\}, \quad \forall\, j \in \{1, \ldots, m\} \text{ and } \forall\, k \in \{1, \ldots, l\}. \tag{11}$$

where `dis` represents the distance function (i.e., the routing algorithm) used and `tw` represents the time-window function that is being evaluated. The constraints in (2) guarantee that each order is assigned to a single batch. The constraints in (3) guarantee that each batch is assigned to a single picker. The constraints in (4) guarantee that the maximum capacity of each batch is not exceeded. The constraints in (5) guarantee that the collection of batch b_j begins once a picker is available. The constraints in (6) guarantee that the collection of batch b_j starts once the collection of batch b_{j-1} has already started. The constraints in (7) guarantee that the route to collect a batch

b_j cannot start before the timestamps (moments in time) when the orders o_i assigned to that batch have reached the system. The constraints in (8) guarantee that the collection of batch j does not start before the time indicated by the time-window function. The constraints in (9) guarantee that st_j is positive. The constraints in (10) guarantee that the variables x_{ji} are binary. Finally, the constraints in (11) guarantee that the variables y_{jk} are binary. Additionally,

$$T_{service}(b_j) = \frac{\texttt{dis}(b_j)}{v_{routing}} + \sum_{i=1}^{n} \frac{w_i x_{ji}}{v_{extraction}} + t_{setup}, \quad \forall j \in \{1, \ldots, m\}. \tag{12}$$

Similarly, the same problem can be studied by minimizing the completion time of the received orders or the maximum turnover time. In both cases, the only difference with respect to the previous model would be the replacement of the objective function (Eq. 1) with Eq. 13 (in the case of the completion time) or Eq. 14 (in the case of the maximum turnover time).

$$\min \max_{j \in \{1,\ldots,m\}} \big(st_j + T_{service}(b_j)\big). \tag{13}$$

$$\min \max_{i \in \{1,\ldots,n\}} \sum_{j=1}^{m} \big((st_j + T_{service}(b_j)) * x_{ji}\big) - ar_i. \tag{14}$$

It is worth mentioning that the completion time is determined by the moment in which the picker delivers the last batch, while the maximum turnover time objective function is determined by the turnover time of the order that remains longer in the system.

3. State of the art

In this section we review the chronological evolution of the concept of time window in the literature, which has changed during the years. In addition, we highlight the contribution of each reviewed paper to this concept. Next, in Section 4 we classify and detail the most relevant time-window methods proposed in the state of the art.

The time-window concept emerges in the context of the development of batching algorithms as an additional batching strategy, and it has evolved until being studied as an independent part of the picking process. The first study dates from 1983 (Quinn, 1983) where the term time-window batching

is used as a strategy to reduce picking time. Il-Choe and Sharp (1991) presented several results for its general application in improving the efficiency of a picking system. Tang and Chew (1997) and Chew and Tang (1999) performed various mathematical analyses related to the use of time-window batching as a batch generation method. Specifically, the upper and lower limits are calculated for the problem, as well as the estimation of the variance and the mean of several objective functions such as travel time, service time, and turnover time. Yu and De Koster (2009) expanded previous works by presenting a model based on queueing theory and introducing the calculation of the concept of "Expected waiting time to form a new batch" ($E[Wj]$) applied to this type of problem. The same year, Van Nieuwenhuyse and De Koster (2009) published an article that estimates the average processing time of an order, using Variable Time-Window Batching (VTWB) or Fixed Time-Window Batching (FTWB) as a batching algorithm. In the same study, the impact of two picking policies (pick-and-sort and sort-while-pick) is also compared for the same Time-Window Batching scenario. This article also develops the concept of $E[Wj]$. Later, in 2012, a new method was introduced in Bukchin et al. (2012) for the first time to accurately calculate the departure time of each picker, assuming that all the arrival times of the orders are known. Based on this information, an approximate model is proposed to determine the waiting strategy for future arrivals of orders. The previous article uses Markov Decision Processes and they are compared with a couple of naive heuristics. This work studies the minimization of delays in the delivery of orders, as well as the costs associated with the overtime of the pickers. The same year, Henn (2012) presented a metaheuristic algorithm (Iterated Local Search) for batching, together with a heuristic to calculate the time window.

Xu et al. (2014) presented new results using $E[Wj]$ for the VTWB calculation, together with the First Come First Served (FCFS) algorithm for the batching process. In this paper, the authors calculate the optimal size of batches in this context, for the minimization of the average throughput time . Subsequently, Zhang et al. (2016) first and Zhang et al. (2017) later introduced a rule-based hybrid heuristic, which linked the computation of the time window together with seed-type algorithms for batching. The same year, Giannikas et al. (2017) proposed three variants of a so-called interventionist strategy, consisting in new Variable Time-Window policies based on the number of orders that arrive at the system. Duda and Stawowy (2019), used a Variable Neighborhood Search (VNS) to study the Joint Batch Sequenc-

ing and Picker Routing Problem with time windows. In this work, batches are formed beforehand and known in advance. Specifically, they divide the time window into fixed periods of 30 minutes and propose four new neighborhoods to solve the problem in a comprehensive way. Later, Gil-Borrás et al. (2020a) performed a simple comparison between Fixed Time-Window and Variable Time-Window methods. Finally, in the same year, Leung et al. (2020) presented the Intelligent B2B order handling system, solving the Integrated Online Pick-to-sort Order Batching using the Fixed and Variable Time-Window Batching strategies.

4. Time-window methods

According to the literature, a possible classification of time-window methods would divide them into Fixed Time Window and Variable Time Window. In this work, we use these categories to classify the descriptions of the most relevant time-window methods present in the state of the art. Furthermore, some of these methods are selected to be empirically compared in Section 6.

4.1. Fixed Time-Window methods in the literature

Fixed Time-Window (FTW) methods are characterized by determining a fixed waiting time that can depend on different factors of the system / instance, such as the available space in the batch, the distribution of the arrival of orders, or the average number of articles of the received orders . Specifically, the strategy that determines the time window could set a single fixed time for all instances of the problem. Alternatively, a FTW strategy could also set a different fixed time for each instance, but depending on a parameter of the instance. The most representative Fixed Time-Window methods in the literature are described below:

- **No-wait method (FTW_NW)**: This is the simplest FTW method. It consists of starting the picking of the next batch as soon as there is a picker available and a batch waiting to be picked. This strategy is one of the most used in the literature. In fact, if not stated otherwise, it is taken as the default strategy. Some examples of works that use this strategy are Gil-Borrás et al. (2020a,b, 2021).

- **Methods with the same fixed time for all instances (FTW_CT)**: It consists of establishing, following any criteria (e.g., a specific time

after each picking route, a fixed schedule for departures, etc.) , a fixed time that a picker must wait, each time the picker is available, regardless of the specific instance in which the picker is working. This strategy was explored in Gil-Borrás et al. (2020a), where different time-window methods are compared. Specifically, the authors in this paper demonstrate how, in general, this type of method returns worse solutions than methods which use a Variable Time-Window strategy.

- **Methods with a calculated fixed time for each instance (FTW_ZH)**: These methods can be considered as a particularization of the FTW_CT. They consist of setting a fixed time based on a calculation per instance. Specifically, the time window t to start collecting the batch b is calculated as follows: $t = (Q/q) \cdot \lambda \cdot \beta$, where Q is the maximum batch capacity, q is a uniform distribution indicating the number of items of an order, and λ defines the distribution that indicates the arrival rate. Therefore, $(Q/q) \cdot \lambda$ estimates the average time to fulfill a batch. Finally, β is a coefficient of the desired average of fulfillment of a batch before starting the picking process. Notice that β is a search parameter that is studied in Section 6.2.1 . An example of the use of this type of methods was described in Zhang et al. (2017) although it is based on the studies of Xu et al. (2014) and Van Nieuwenhuyse and De Koster (2009). In this work, the authors determined the expected arrival time of the next k orders, required to complete a batch, on the basis of the arrival information of previous orders.

We have selected FTW_NW and FTW_ZH methods to be experimentally evaluated in Section 6 since according to the previous comparison (Gil-Borrás et al., 2020a) FTW_CT methods performed worse.

4.2. Variable Time-Window methods in the literature

Variable Time-Window (VTW) methods are characterized by having a variable waiting time, which can be different at each picking moment for the same instance depending on different factors. The most representative methods in the literature are described below:

- **Methods based on a minimum number of orders in queue (VTW_QO)**: This strategy consists of starting the collection of orders in a batch when there is a minimum number of pending orders

to be collected in the system. This method was recently used in Gil-Borrás et al. (2020a), where the authors explored the performance of this strategy for different number of queued orders (4, 8, and 16).

- **Methods based on a minimum number of batches (VTW_BA)**: This method sets a different time window depending on the completion of a minimum number of batches previously defined, prior to starting picking. The simplest version of this strategy consists of waiting until the first batch is full. Particularly, when a new batch is configured, it is assumed that the previous batch does not have the capacity to accommodate new orders, so at that time the picker can start collecting the first batch, and so on.

- **Reactive methods according to the collection conditions (VTW_HE)**: This method determines the time window on the basis of the specific system conditions such as the arrival times of the orders and the service times to collect them (Henn, 2012). Specifically, if there is more than one batch available for being collected a new route is started as soon as a picker becomes available. Otherwise, the method calculates the moment to start a new route using the following formula: $\max(t, (1+\alpha) \cdot r_i + \alpha \cdot st_i - st_j)$, where t is the current instant time, st_j is the service time of the batch j, st_i is the longest service time of any order i in batch j when it is collected in isolation, and r_i is the arrival time of order i. Finally, α is a coefficient that weighs the arrival time of the order i and the service time of the current batch under construction. Notice that α can be considered as a search parameter and its influence is studied in Section 6.2.2.

- **Methods based on data-built models (VTW_MM)**: In addition to the previously introduced heuristic methods, there is a family of methods based on machine learning models. Specifically, in Bukchin et al. (2012) the authors propose a method that uses Markov decision processes to calculate a subsequent model that determines the optimal moment in which a picker should have started the picking of each batch. Based on the model built with previous data, it establishes a decision-making system for future orders. However, the construction of the exact model presents the difficulty of being a computationally costly task.

According to a previous comparison (Gil-Borrás et al., 2020a), VTW_BA and VTW_HE methods perform better than VTW_QO. Therefore, we have selected the former methods to be experimentally evaluated in Section 6. Additionally, VTW_MM is impractical for real-size instances as the ones used in this paper.

4.3. New time-window methods

In addition to the previous methods identified in the state of the art, in this work, we propose two new VTW methods to determine the time window, with the aim of finding balance between the effect of waiting in two different objective functions: picking time and completion time. Particularly, waiting might improve the picking time for a particular batch, but could also deteriorate the completion time of the whole system. The proposed strategies try to determine the expectancy of the next order that will arrive at the system to fit the current batch. These methods have been tested in different empirical scenarios and improve the methods in the state of the art in several of them. The experiments are reported in Section 6.

- **Method based on the available capacity (VTW_M1)**: This method considers that an available picker should wait while there is a batch under construction still incomplete, together with a high estimated probability that the next order will fit in the available capacity. Since the method is based on estimations, in the event that the next order that arrives in the system exceeds the available capacity (i.e., generating a new batch), the picker will start collecting the previous batch. On the other hand, while the capacity is not exceeded after a new arrival, the method constructs a probability distribution based on the size of the orders previously arrived at the system. Based on this distribution, it calculates the probability that the next order will have a smaller size than or equal to the available capacity in the current batch. Then, based on a threshold, the method decides whether to start picking or to wait. The probability threshold is a search parameter and should be adjusted experimentally.

- **Method based on the available capacity with rules (VTW_M2)**: This method could be considered an extension of VTW_M1, but adding two new criteria to start picking: 1) the time calculated for the picking task of the orders currently in the batch under construction is shorter

than the time needed until the estimated instant of arrival of the next order; 2) the average time required to pick the items assigned to the batch under construction is at least 10% shorter than the average picking time of items previously collected.

The VTW_M1 and VTW_M2 methods have been included in the experimental evaluation in Section 6.

5. Description of the comparative study

To evaluate the time-window algorithms exposed in Section 4, it is necessary to define the strategies that will be used for the rest of the activities in the context of the OOBP. Also, it is necessary to determine the objective functions to be optimized.

For variants with a single picker, it is necessary to define the batching, selecting, and routing strategies. Moreover, if there are multiple pickers, it is necessary to handle an additional task known as *assigning*, which consists in determining which picker is assigned to the collection of which batch. .

In this work, we study different batching and routing strategies, which are described in Section 5.1 and Section 5.2, respectively. On the other hand, the selecting method used consists of selecting the batch that contains the largest number of items. In the event that there is more than one batch with the maximum number of items, we select the batch with the shortest picking route. Additionally, the assigning method, which decides which batch is collected by which picker, assigns the next batch to be collected to the first available picker. Finally, we study the minimization of three different objective functions: Picking time, completion time, and maximum turnover time, which are described in Section 5.3.

5.1. Batching strategies

The batching task consists of grouping a set of orders in a batch, which will be collected together in a single picking route. We consider that orders cannot be split into different batches. There is a great variety of algorithms published in the literature for this task. In this work, we compare three different batching strategies: FCFS, Iterated Local Search (ILS), and General Variable Neighborhood Search (GVNS). The purpose of exploring different batching strategies is to determine if the time-window method selected depends on the batching algorithm used. Next, we review the batching strategies considered.

5.1.1. First Come First Served

FCFS is a simple heuristic algorithm that sorts orders according to the arrival time of each order to the system. Then, batches are generated starting from an empty one and adding orders to the batch one by one following the previous predefined sequence, starting with the earliest arrived order . Once an order does not fit into the current batch, that batch is considered complete, and a new batch is generated. The process is repeated until all orders have been assigned. This simple algorithm is widely used in the literature and is usually considered a baseline method for comparison.

5.1.2. Iterated Local Search

ILS metaheuristic was devised in Lourenço et al. (2003) as a method based on local search to perform an efficient exploration of the solution space. Specifically, it consists of alternating an improvement phase with a perturbation phase. The former is devoted to reach a local minimum, while the latter is used to diversify the search. ILS algorithm was used in the context of OBP in Henn et al. (2010). In this work, the improvement phase uses two neighborhoods: The first one is based on the exchange operation of two orders located in two different batches (swap); and the second one is based on the insertion of an order in a different batch from the current one (shift). On the other hand, the perturbation phase is based on the exchange of a random number of orders between batches selected at random. We have coded and used the strategy proposed in Henn et al. (2010) for our experiments. Particularly, the method received an initial solution obtained with the FCFS strategy. Then, it was executed following the recommendations of the authors, during at least 60 seconds and 100 iterations before considering a solution to select the next batch to collect.

5.1.3. General Variable Neighborhood Search

The VNS methodology was initially proposed in Mladenović and Hansen (1997) as a search strategy based on the concept of neighborhood change to escape from local optima. Among the multiple VNS variants, Variable Neighborhood Descent (VND) is characterized by systematically exploring several neighborhoods using a local search procedure. General Variable Neighborhood Search (GVNS) uses VND as an intensification strategy, but adds a perturbation strategy for diversification purposes. In this work, a GVNS algorithm has been used in combination with the VND procedure proposed in Gil-Borrás et al. (2020b) for the OOBP context. This algorithm uses three

neighborhoods based on the following operations: Exchange of two orders belonging to a different batch (1x1); exchange of two orders belonging to a batch with an order belonging to a different batch (2x1); and insertion of an order in a different batch than the current one (1x0). The neighborhoods and their order have been established as indicated by the authors. The proposed local search procedures use a first improvement strategy. Finally, the perturbation phase consists of performing a random exchange of orders between orders belonging to different batches. The initial solution provided to the GVNS was obtained at random. The method is then continuously executed for 15 seconds, restarting the search afterwards from a new random solution.

5.2. Routing strategies

The routing task consists of generating a route that enables the collection of the selected set of orders in a batch within the warehouse. This strategy is also used to determine aspects such as picking time or distance traveled. In general, there are three families of strategies to generate the route: Exact methods, heuristic methods, and metaheuristic methods. This research focuses on rectangular-shaped warehouses with two crossing aisles (one at the front and one at the back) and a variable number of parallel aisles where the products are stored. For this warehouse layout, heuristic methods are the most popular in the literature and in real scenarios, because they are easy to implement and easy to understand by pickers (Hall, 1993; Petersen and Schmenner, 1999; Fontin and Lin, 2020).

In this work, three routing heuristic algorithms have been compared: S-shape, Largest-gap, and Combined for demonstrating that the time-window strategy is independent of the routing algorithm used. Specifically, we have identified and reviewed 123 journal articles related to Order Batching. Among them, the S-Shape was used in 54% of the cases, the Largest Gap in 17% of the cases, and the Combined in 9% of the cases. The rest of the papers reviewed include a variety of methods such as: Mid-Point, Return, Combined+, or Ratliff and Rosenthal, among others. An empirical comparison of the three selected procedures can be found in Petersen (1997) and Petersen and Aase (2004).

5.2.1. S-Shape

The S-Shape method is a heuristic strategy in which pickers start their routes at the depot, which is located in the front cross aisle. Pickers move through the front cross aisle until they reach the leftmost parallel aisle with

items to collect in the warehouse. Then, the pickers fully traverse this parallel aisle and any other that contains items to collect. To change from one parallel aisle to another, the picker can use either the front cross aisle or the back cross aisle indistinctly, depending on the route. However, pickers must finish their route in the depot again, so in the event that the number of parallel aisles to traverse is odd, the last aisle will only be traversed until reaching the last item to be collected and then, the pickers turn around and come back to the front cross aisle to return to the depot and finish the picking route. We refer the reader to Hall (1993) and Petersen (1995) for a further detailed description of this strategy.

5.2.2. Largest Gap

The Largest-Gap method is a heuristic strategy based on the concept of gap. The gap is defined as the distance between two items to be picked consecutively and located in the same parallel aisle, or also the distance between a cross aisle and the first item to be picked in a parallel aisle. Again, the picking route begins and ends in the depot, which is located in the front cross aisle. Notice that the entrance to the parallel aisles can be made from any of the two crossing aisles (either the back or the front cross aisle) and pickers only enter the aisles that have items to collect. The Largest-Gap strategy consists of avoiding going through the part of the aisle with the largest gap. To that aim, when the pickers arrive at the position where the largest gap starts, they turn around in the parallel aisle and go back to the initial cross aisle. This procedure is performed except for the first and last parallel aisles with items to collect, which are fully traversed. Therefore, pickers might enter the same aisle twice: One from the front cross aisle and one from the back cross aisle. We refer the reader to Hall (1993) and Petersen (1995) for a further detailed description of this strategy.

5.2.3. Combined

Finally, the Combined heuristic strategy consists of the combination of the two previous heuristics: S-Shape and Largest Gap. Specifically, the Combined method uses the lowest cost technique (S-Shape or Largest Gap) in each aisle with items to collect. As in the previously studied methods, the pickers begin their route at the depot, which is located in the front cross aisle. Pickers traverse the front cross aisle until they reach the leftmost parallel aisle with items to collect, deciding the best strategy for performing picking in that and subsequent aisles. Notice that the restrictions associated

with either Largest Gap or S-Shape strategies must be satisfied. Once all items have been collected, the pickers return to the depot. We refer the reader to De Koster and Van Der Poort (1998) and Menéndez et al. (2017b) for a further detailed description of this strategy.

5.3. Objective functions

In this work, three widely used objective functions for different variants of the OBP are compared: The minimization of the picking time; the minimization of the completion time; and the minimization of the maximum turnover time. Specifically, we have identified and reviewed 123 journal articles related to Order Batching. Among them, 51% of the papers studied the picking time as the objective function, 22% of the papers considered the completion time, while 19% of them studied the turnover time (including those papers which consider the existence of due dates in the orders). Finally, the remaining 8% of papers considered other related objective functions such as cost or workload balance, among others.

The picking time is the time that pickers need to perform the picking task of all orders once the batches are conformed. The completion time is similar to the picking time, but also includes the waiting time for the arrival of new orders. Reducing the picking and completion times, frees some time for the pickers to perform other activities. Also, it helps to reduce energy consumption (in the case that the picking uses machinery) or to reduce the tiredness of the workers (in the case that they walk through the warehouse). Finally, the maximum turnover time is the maximum time that an order remains in the system since it arrives until it is served. Reducing the turnover time results in a direct benefit for the customer since it helps to deliver the products faster.

6. Experiments

This section presents different experiments to evaluate the time-window methods previously detailed in the context of the OOBP. Also, we study the relationship between time-window strategies and other strategies, such as routing or batching algorithms, as well as the influence of either the set instances or the optimized objective function on the performance of the methods.

All methods used in the experimentation, including those of the state of the art, were coded in Java 8 and run on an Intel (R) Core (TM) 2 Quad

CPU Q6600 2.4 Ghz computer, with 4 GB DDR2 RAM memory and Ubuntu 18.04.1 64 bit LTS operating system.

In Section 6.1, we present the sets of instances used in the comparison. In Section 6.2, we perform a set of preliminary experiments to adjust the values of the different compared time-window algorithms. Finally, in Section 6.3, we present the final experiments. In those experiments, we compare the behavior of routing and batching algorithms when combined with different time-window strategies.

6.1. Instances

Two sets of instances widely used in the state of the art of different variants of order batching problems have been selected for this article. It is important to note that, to the best of our knowledge, there are no previous specific datasets for "time-window algorithms" in the context of OOBP, since there are no previous studies just devoted to this concept in isolation. However, most of the methods included in our comparison, which deal with the concept of time window, are presented in articles using these data sets. The objective of using two sets of instances is to evaluate whether there is any dependence of the time-window strategies on the set of instances used.

The first data set (Dataset #1) is composed of 80 instances corresponding to 4 different warehouses (denoted as W1, W2, W3, and W4). It was originally proposed in Albareda-Sambola et al. (2009), and it has been used in many related works (Gil-Borrás et al., 2020b; Menéndez et al., 2017b, 2015).

The second data set (Dataset #2) is composed of 64 instances corresponding to a single warehouse (denoted as W5). It was originally proposed in Henn (2012), and it has also been used in many related papers (Aerts et al., 2021; Alipour et al., 2020; Gil-Borrás et al., 2020b; Koch and Wäscher, 2016; Menéndez et al., 2017a,b; Pérez-Rodríguez et al., 2015).

Instances in both data sets correspond to rectangular single-block warehouses with two cross aisles, one at the front and one at the back. The cross aisles are linked by several parallel aisles containing shelves with picking positions at both sides of the aisle. The depot is the place in the warehouse where pickers start and end the picking routes and where the products are deposited once they have been collected. We consider that there is a single depot in the warehouse that is located in the center or in the left corner of the front cross aisle, depending on the instance. These data sets can be down-

	Dataset #1 (Albareda-Sambola et al. (2009))				Dataset #2 (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
# parallel aisles	4	10	25	12	10
# of picking positions per aisle	2x30	2x20	2x25	2x16	2x45
# of total picking positions	240	400	1250	384	900
Parallel aisle length	50 m	10 m	50 m	80 m	45 m
Parallel aisle width	4.3 m	2.4 m	5 m	15 m	5 m
# of instances	20	20	20	20	64
Travel speed (m/min.)	48	48	48	48	48
Extraction speed (items/min)	6	6	6	6	6
Batch setup time	3 min	3 min	3 min	3 min	3 min

Table 2: Characteristics of the warehouses and the work parameters.

	#Customer orders							
	40	**50**	**60**	**80**	**100**	**150**	**200**	**250**
4 Hours	0.167	0.208	0.250	0.334	0.417	0.625	0.834	1.042

Table 3: λ values for each number of orders.

loaded at http://grafo.etsii.urjc.es/optsicom/oobp/. In Table 2, we compile the main characteristics of the instances used in our experiments.

Additionally, since these experiments occur in a dynamic / online context, where orders arrive dynamically (i.e., once the picking task has already started) it is also necessary to model the arrival of the orders to the system. In this case, we have studied a scenario where the arrival of orders follows an exponential distribution derived from a Poisson process (Poisson, 1837) characterized by a time horizon of 4 hours ($t = 4$). The number of events (the arrival of orders in this context) that occur in a time interval t is a random variable $X(t)$ with mean $E[X(t)] = \lambda * t$. The λ value is selected depending on the number of orders considered for each instance. In Table 3, we report the values of λ for each case considered.

6.2. Preliminary experiments

This section is devoted to tuning the parameters of the compared time-window algorithms: FTW_ZH (Zhang et al., 2017), VTW_HE (Henn, 2012),

VTW_M1, and VTW_M2 (proposed in this paper). To make these adjustments, a diverse subset of 24 instances has been selected from the original sets of instances previously described. It is worth mentioning that, in the case of VTW_BA, three different configurations (2, 3, and 4 generated batches) are used in the final experiments, denoted as: VTW_B1, VTW_B2, and VTW_B3, respectively.

6.2.1. Selection of parameter β in FTW_ZH

The time-window method proposed in Zhang et al. (2017) uses a β parameter in the formula that determines the fixed time interval of the time window. We refer the reader to Section 4.1 for a detailed description of this parameter.

In this experiment, different values of β (0.5, 1, and 1.5) and their influence are compared. In Table 4, we compare, for each value of β, the average value of the picking time measured in seconds, Avg.(s), the deviation to the best solution of the experiment, Dev.(%), and the number of best solutions found, #Best. Similarly, the same metrics are reported for the completion time and turnover time . Observing the results, the configuration that obtained the highest number of best solutions in either picking time, completion time, or turnover time was $\beta = 1.5$ with a good performance in the other two indicators. Therefore, we used this configuration for the final experiments.

		β **value**		
		0.5	**1.0**	**1.5**
Picking time	**Avg.(s)**	**41291**	41300	41293
	Dev.(%)	**0.17%**	0.31%	0.18%
	#Best	10	9	**13**
Completion time	**Avg.(s)**	41955	41967	**41950**
	Dev.(%)	0.20%	0.36%	**0.19%**
	#Best	7	7	**10**
Maximum turnover time	**Avg.(s)**	36385	36157	**36180**
	Dev.(%)	**6,55%**	7,28%	8,61%
	#Best	7	7	**10**

Table 4: Study of different values for the β parameter in FTW_ZH method for picking time and completion time.

6.2.2. Selection of parameter α in method VTW_HE

The time-window method proposed in Henn (2012) uses a parameter α to determine the time window before starting the picking of the next batch under construction. We refer the reader to Section 4.2 for a detailed description of this parameter.

In this experiment, different values of α (25%, 50%, 75%, and 100%) are compared. The value 0% for α is not evaluated as it would be equivalent to the FTW_NW method that is already included in the final experiments.

In Table 5, we show the three previously introduced quality indicators for the considered values of α for the three objective functions. In this case, it can be observed that the best value for the parameter α is 50% in the three indicators for two out of the three objective functions studied. For this reason, $\alpha = 50\%$ is selected for the configuration of the method VTW_HE in the final experiments.

		α value			
		25%	**50%**	**75%**	**100%**
Picking time	**Avg.(s)**	40515	**40430**	40483	40518
	Dev.(%)	0.61%	**0.51%**	0.57%	0.73%
	#Best	6	**12**	7	5
Completion time	**Avg.(s)**	42121	**42022**	42085	42111
	Dev.(%)	1.06%	**0.83%**	0.97%	1.02%
	#Best	3	**7**	3	3
Maximum turnover time	**Avg.(s)**	36895	37254	38035	**36887**
	Dev.(%)	**7,46%**	10,06%	14,67%	9,44%
	#Best	6	5	2	**11**

Table 5: Study of different values for the α parameter in VTW_HE method for picking time and completion time.

6.2.3. Study of the capacity threshold in VTW_M1

The VTW_M1 method determines the probability that the next order arriving in the system will fit into the available capacity in the batch under construction. In this experiment, we study different thresholds to start picking if the probability is below that threshold. Specifically, we have evaluated threshold values between 20% and 70% (in steps of 10%). Extreme values have not been evaluated, as a threshold of 0% indicates that the next order would not fit in the batch, therefore there is no point in waiting for it. Similarly, a value of 100% would equate to the batch being empty.

In Table 6, we report the same quality indicators as previously introduced for the different threshold values evaluated, both for the picking time and for the completion time. In this case, the best value for this parameter in the three proposed indicators, for both objective functions, is 60%. Therefore, this value has been selected for the configuration of VTW_M1 in the final experiments.

		Threshold (%)					
		20%	**30%**	**40%**	**50%**	**60%**	**70%**
Picking time	**Avg.(s)**	40738	40793	40779	40856	**40721**	40813
	Dev.(%)	0,24%	0,38%	0,36%	0,31%	**0,21%**	0,33%
	#Best	7	4	4	6	**8**	5
Completion time	**Avg.(s)**	41725	41776	41760	41839	**41701**	41799
	Dev.(%)	0,23%	0,37%	0,34%	0,30%	**0,19%**	0,34%
	#Best	4	1	5	**6**	**6**	2
Maximum turnover time	**Avg.(s)**	36792	36541	36314	36851	36469	**35910**
	Dev.(%)	15,51%	14,45%	15,85%	14,61%	13,85%	**7,42%**
	#Best	1	2	4	5	4	**8**

Table 6: Study of different threshold values in VTW_M1 method for the picking time and completion time.

6.2.4. Study of the capacity threshold in VTW_M2

The VTW_M2 method determines the probability that the next order arriving to the system will fit into the available capacity (as VTW_M1) but including several additional rules. Specifically, we have evaluated the threshold values between 20% and 70% (in steps of 10%). Again, the extreme values have not been evaluated as a threshold of 0% indicates that the next order would not fit in the batch; therefore, there is no point in waiting for it. Similarly, a value of 100% would be equivalent to waiting always.

In Table 7, we show the same three quality indicators presented previously. In this case, the best value for this parameter, for two out of the three objective functions, is 40% for all the proposed indicators, except for the number of best solutions in the case of completion time. Therefore, this value has been selected for the configuration of VTW_M2 for the final experiments.

6.3. Final experiments

The aim of this work is to evaluate the behavior of the strategies selected previously to determine the time window in different scenarios. To that

		Threshold (%)					
		20%	**30%**	**40%**	**50%**	**60%**	**70%**
Picking time	**Avg.(s)**	40359	40335	**40285**	40334	40310	40295
	Dev.(%)	0,43%	0,47%	**0,27%**	0,36%	0,34%	0,41%
	#Best	**6**	4	**6**	**6**	5	3
Completion time	**Avg.(s)**	40858	40838	**40784**	40837	40815	40790
	Dev.(%)	0,41%	0,47%	**0,28%**	0,35%	0,35%	0,39%
	#Best	4	**5**	3	**5**	**5**	2
Maximum turnover time	**Avg.(s)**	38408	39093	38720	38589	38641	**38316**
	Dev.(%)	**4,20%**	9,35%	5,52%	6,73%	7,56%	5,15%
	#Best	6	2	**7**	2	4	3

Table 7: Study of different threshold values in VTW_M2 method for the picking time and completion time.

aim, we have divided our final experimentation in four sections where we vary: The routing strategy, the batching strategy, or the characteristics of the instance (number of pickers and congestion in the arrival of orders). Our objectives are: 1) study the influence of the time-window algorithm on the performance of the methods depending on the objective function; 2) identify the best strategy for determining the time window in each scenario; and 3) determine if there exists a dependency of the waiting strategy with respect to either the batching or routing strategies, or other factors such as the number of pickers or the congestion in the arrival of orders.

Notice that all experiments use the same selection and assignment strategies (see Section 5). Additionally, in those experiments where the number of pickers in the warehouse is not explicitly indicated, it is considered that there is only one picker. All experiments in this section have been performed on the whole data set of instances introduced in 6.1. Finally, in Section 6.3.5, we have performed a statistical analysis of the results.

6.3.1. Impact of time-window algorithms on the performance of the studied methods for different objective functions, when combined with several routing strategies

In this experiment, we study the picking time and the completion time objective functions when varying the routing and waiting strategies. Particularly, we combined a GVNS batching method (see Section 5.1.3) with three different routing algorithms (S-Shape, Largest-Gap, and Combined) introduced in Section 5.2, and with eight time-window algorithms (VTW_B1,

VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW-_ZH) introduced in Section 4.

In Figure 1, we represent the different algorithmic variants compared in terms of completion time and picking time. Particularly, we report the average among all instances of the minimum completion time or the minimum picking time. It can be observed that the time-window method substantially influences the results obtained (either in terms of picking time or in terms of completion time) regardless of the routing method. Also, it is observed that the influence of each time-window strategy on the final result (both in picking time and in completion time) remains constant for each routing method. That is, the best time-window method when using Combined, Largest Gap, or S-Shape routing method is always VTW_B3 in terms of picking time. It is also observed that the best time-window method to minimize the picking time (VTW_B3) is not the best time-window method to minimize the completion time. In this case, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH, behave similarly one to each other, and they can be considered the best methods among the compared ones for the completion time.

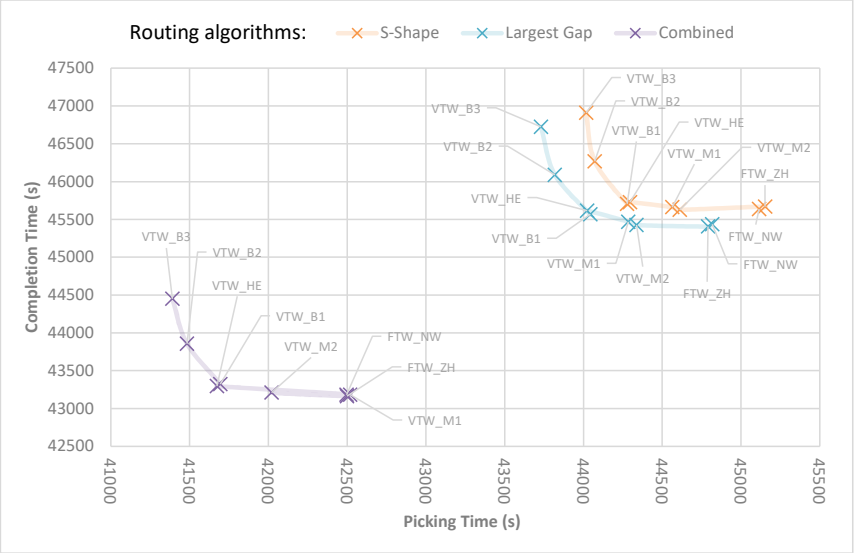


Figure 1: Behavior of the picking time and completion time when combining the GVNS batching algorithm, with different routing strategies (S-Shape, Largest-Gap, and Combined) and different time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

Similarly, in Figure 2, we compare the performance of the previous methods in terms of maximum turnover time, compared to the completion time. In this case, it is confirmed that, whatever the routing method is, the use of one or another time-window method substantially influences the result obtained in terms of maximum turnover time. Again, we report the average values obtained among all instances. In this case, similarly to what happens with the completion time, the methods VTW_M1, VTW_M2, FTW_NW, and FTW_ZH perform alike and they can be considered as the best ones among the compared methods, for minimizing the maximum turnover time.

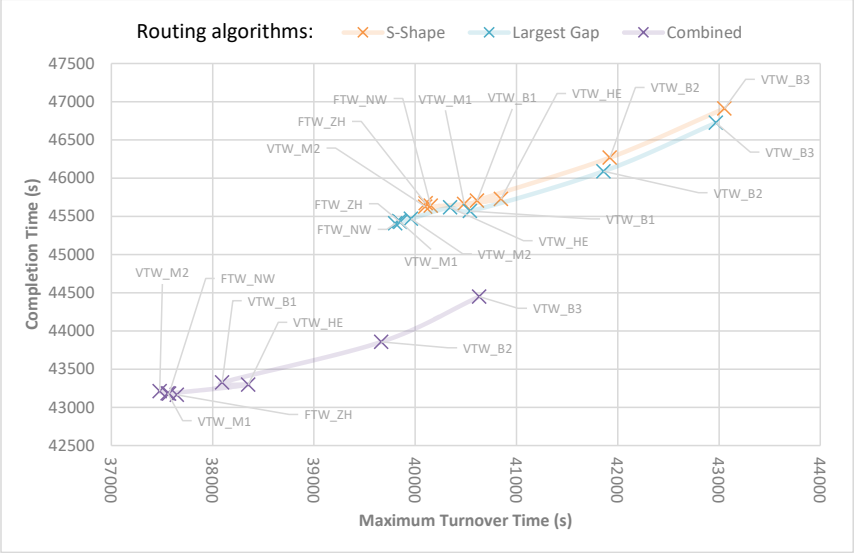


Figure 2: Behavior of the maximum turnover time and completion time when combining the GVNS batching algorithm, with different routing strategies (S-Shape, Largest-Gap, and Combined) and different time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

The detailed results to elaborate Figure 1 and Figure 2 can be found in Appendix A.

6.3.2. Impact of time-window algorithms on the performance of the studied methods for different objective functions, when combined with several batching strategies

In this experiment, we study the picking time and the completion time objective functions when varying the batching and waiting strategies. Particularly, we combined three different batching methods (GVNS, ILS, and

FCFS) described in Section 5.1, with the S-Shape routing strategy (see Section 5.2.1), and with eight time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) introduced in Section 4.

In Figure 3, we represent the different algorithmic variants compared in terms of completion time and picking time. As it can be observed, whatever the batching method is, the use of one or another time-window method substantially influences the results obtained (either in terms of picking time or in terms of completion time). Also, it is observed that regardless of the chosen batching method, the influence of each time-window strategy on the final result (both in the picking time and in the completion time) remains constant. That is, the best time-window method for picking time when using GVNS or FCFS is VTW_B3, while VTW_B2 is slightly better than VTW_B3 when combined with ILS. It is also observed that the best time-window method to minimize the picking time (VTW_B3) is not the best time-window method to minimize the completion time. In this case, again, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH are the best methods and behave similarly one to each other for any of the batching methods compared.

Similarly, in Figure 4, we compare the performance of the previous methods in terms of maximum turnover time when compared with the completion time. In this case, it is confirmed that, whatever the batching method is, the use of one or another time-window method substantially influences the result obtained in terms of maximum turnover time. In this case, similar to what happens with the completion time, the methods VTW_M1, VTW_M2, FTW_NW, and FTW_ZH perform alike and they can be considered as the best ones among the compared methods, for minimizing the maximum turnover time, for any of the batching strategies compared.

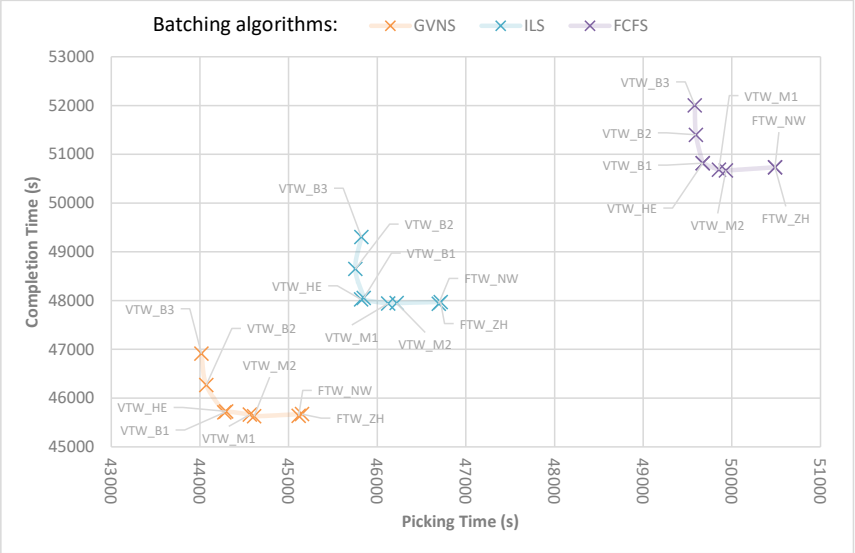


Figure 3: Behavior of the picking time and completion time when combining different batching algorithms (GVNS, ILS, and FCFS), with several time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the same routing algorithm (S-Shape).

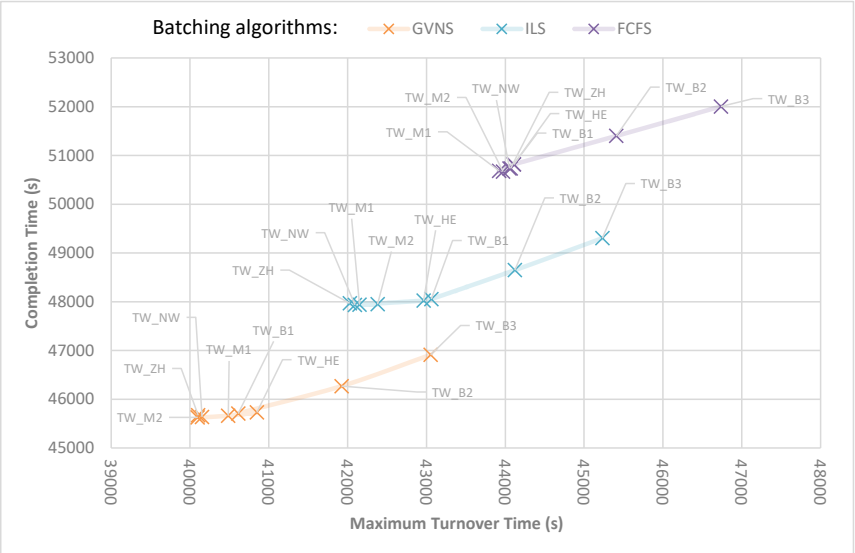


Figure 4: Behavior of the maximum turnover time and completion time when combining different batching algorithms (GVNS, ILS, and FCFS), with several time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the same routing algorithm (S-Shape).

The detailed results to elaborate Figure 3 and Figure 4 can be found in Appendix B.

6.3.3. Impact of time-window algorithms on the performance of the studied methods for different objective functions, when varying the number of pickers

In this experiment, we study the picking time and the completion time objective functions, when varying the number of pickers and the waiting strategies. Particularly, we combined the GVNS batching strategy (see Section 5.1.3), with the S-Shape routing strategy (see Section 5.2.1), and with eight time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) introduced in Section 4 for scenarios with a different number of pickers (1, 2, 3, 4, and 5). The results of this experiment are presented in Figure 5.

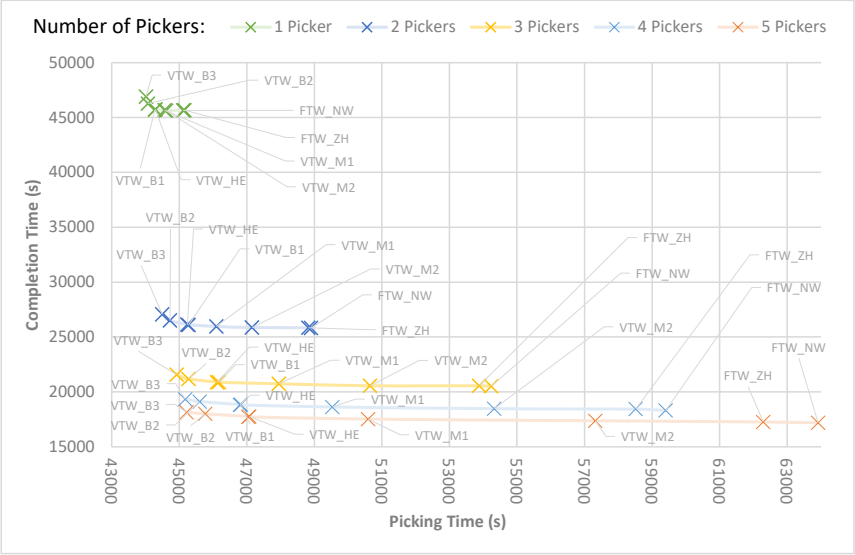


Figure 5: Behavior of the picking time and completion time when increasing the number of pickers (1, 2, 3, 4, and 5 pickers) for different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH), using a GVNS batching algorithm and a S-Shape routing strategy.

As expected, there is an increase in the picking time when increasing the number of pickers, but also a decrease in the completion time. On the other hand, it is observed that the method configured with VTW_B3 is the best

in terms of picking time for any scenario (1, 2, 3, 4, and 5 pickers). Similarly, the methods FTW_NW, FTW_ZH, VTW_M1, and VTW_M2 present a similar behavior in terms of completion time, and can be considered as the best methods for the minimization of this objective function, in any of the scenarios studied (1, 2, 3, 4, and 5 pickers).

Finally, it is observed that the differences between the best and the worst methods for the same number of pickers are very small when considering the completion time, regardless of the time-window method. However, in the case of picking time, the differences increase substantially as the number of pickers increases.

Similarly, in Figure 6, the same methods are compared in terms of the maximum turnover time and the completion time. In the case of the maximum turnover time, it is generally observed that it decreases as the number of pickers increases. More specifically, observing the algorithm configured with the same time-window strategy but for a different number of pickers, both the maximum turnover time and completion time decrease as the number of pickers increases. Furthermore, when the number of pickers increases, the influence of the time-window strategy is more relevant in terms of maximum turnover time, since the difference between the worst and the best methods also increases. Derived from the experiments carried out, the methods FTW_NW, FTW_ZH, VTW_M1, and VTW_M2 (which behavior is similar in terms of maximum turnover time) can be considered the best methods for the minimization of the maximum turnover time objective function, in any of the scenarios studied (1, 2, 3, 4, and 5 pickers).

The detailed results to elaborate Figure 5 and Figure 6 can be found in Appendix C.

6.3.4. Impact of time-window algorithms on the performance of the studied methods for different objective functions, when varying the congestion in the arrival of orders

In this last experiment, we compare the picking time, the completion time, and the maximum turnover time for different variants of an algorithm configured with the GVNS batching strategy (see Section 5.1.3), the S-Shape routing strategy (see Section 5.2.1), and each of the eight compared time-window strategies (see Section 4), for instances with different number of orders received within the same time horizon (i.e., varying the congestion in the system). Specifically, the set of instances extracted from Dataset #2 in Henn (2012) has a size which varies from 40 to 100 orders, while the set of

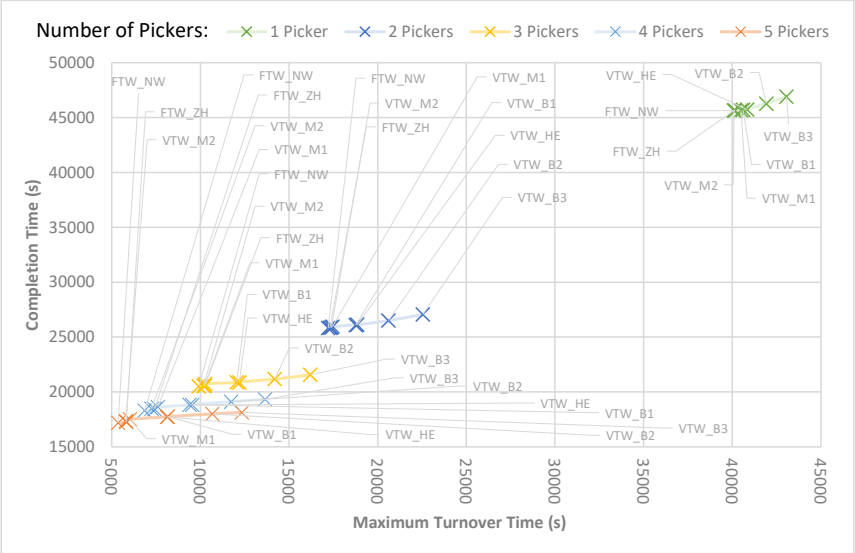


Figure 6: Behavior of the maximum turnover time with respect to the completion time when the number of pickers is increased (1, 2, 3, 4, and 5 pickers), for several time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH), using a GVNS batching algorithm and a S-Shape routing strategy.

instances extracted from Dataset #1 in Albareda-Sambola et al. (2009) has a size which varies from 50 to 250 orders. The results are presented in Tables 8-13 ordered by the objective function (picking time, completion time, and maximum turnover time) and set of instances. Additionally, for each table, the results are organized according to the number of orders.

		Picking time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
40	**Avg (s)**	**15966**	16071	16544	16501	16919	16990	17647	17639
	Dev (%)	**0.22%**	0.94%	4.22%	3.90%	6.67%	7.25%	12.20%	12.16%
	#Best	**10**	5	1	0	0	0	0	0
60	**Avg (s)**	**23066**	23176	23352	23368	23574	23618	24123	24087
	Dev (%)	**0.13%**	0.72%	1.45%	1.63%	2.59%	2.77%	5.57%	5.40%
	#Best	**12**	3	0	0	1	0	0	0
80	**Avg (s)**	**30783**	30843	30956	30931	31134	31061	31372	31423
	Dev (%)	**0.14%**	0.35%	0.72%	0.65%	1.36%	1.19%	2.37%	2.56%
	#Best	**8**	6	0	1	0	0	1	1
100	**Avg (s)**	**36680**	36694	36817	36875	37066	37115	37225	37347
	Dev (%)	**0.28%**	0.31%	0.64%	0.78%	1.32%	1.44%	1.91%	2.18%
	#Best	**7**	**7**	2	0	0	0	0	0

Table 8: Behavior of the picking time when varying the number of orders (congestion), for several time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the GVNS batching algorithm and the S-Shape routing strategy on the instances extracted from Dataset #2 (Henn, 2012).

		Picking time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
50	**Avg (s)**	**20042**	20215	20744	20774	21506	21912	23604	23378
	Dev (%)	**0.13%**	1.89%	6.60%	7.34%	14.27%	20.11%	36.03%	35.56%
	#Best	**11**	4	0	0	1	1	0	1
100	**Avg (s)**	**39603**	39757	39917	39919	40172	40256	40655	40633
	Dev (%)	**0.13%**	0.72%	2.13%	2.16%	3.37%	3.80%	6.00%	5.80%
	#Best	**11**	2	0	1	0	1	2	0
150	**Avg (s)**	**58155**	**58115**	58261	58155	58332	58344	58795	58673
	Dev (%)	**0.27%**	0.44%	0.77%	0.65%	1.13%	1.07%	2.19%	1.98%
	#Best	**7**	4	2	0	2	1	0	0
200	**Avg (s)**	**76521**	76615	76588	76693	76823	76747	77155	76872
	Dev (%)	**0.26%**	0.27%	0.49%	0.56%	0.85%	0.76%	1.41%	1.25%
	#Best	3	**6**	2	2	1	0	0	2
250	**Avg (s)**	95337	**95154**	95317	95438	95547	95454	95796	95980
	Dev (%)	0.22%	**0.18%**	0.49%	0.57%	0.61%	0.61%	1.21%	1.21%
	#Best	**7**	4	2	1	0	2	0	0

Table 9: Behavior of the picking time when varying the number of orders (congestion), for several time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the GVNS batching algorithm and the S-Shape routing strategy on the instances extracted from Dataset #1 (Albareda-Sambola et al., 2009).

		Completion time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
40	**Avg (s)**	20527	19578	18511	18527	18336	18328	18281	**18273**
	Dev (%)	14.08%	8.47%	2.12%	2.23%	1.17%	1.14%	0.75%	**0.72%**
	#Best	0	0	0	**4**	3	2	**4**	3
60	**Avg (s)**	25978	25287	24704	24728	24519	24557	24546	**24508**
	Dev (%)	7.60%	4.42%	1.54%	1.72%	0.83%	0.97%	0.91%	**0.75%**
	#Best	0	1	1	1	**6**	3	1	3
80	**Avg (s)**	32982	32390	31832	31811	31742	**31669**	31734	31792
	Dev (%)	5.07%	2.97%	0.90%	0.84%	0.64%	**0.47%**	0.67%	0.87%
	#Best	0	0	1	2	2	4	**5**	2
100	**Avg (s)**	38715	38108	37628	37684	37673	37721	**37624**	37747
	Dev (%)	3.73%	1.88%	0.43%	0.57%	0.53%	0.64%	**0.41%**	0.68%
	#Best	1	0	3	1	2	**4**	2	3

Table 10: Behavior of the completion time when varying the number of orders (congestion), for several time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the GVNS batching algorithm and the S-Shape routing strategy on the instances extracted from Dataset #2 (Henn, 2012).

		Completion time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
50	**Avg (s)**	26635	25521	24648	24633	24541	24381	24521	**24288**
	Dev (%)	13.24%	7.63%	2.73%	2.77%	2.06%	1.13%	1.19%	**0.70%**
	#Best	0	0	0	0	2	**7**	3	5
100	**Avg (s)**	42452	41848	41147	41146	41064	41089	40999	**40986**
	Dev (%)	5.93%	3.50%	1.16%	1.21%	0.77%	0.75%	0.65%	**0.51%**
	#Best	0	0	0	1	3	**4**	**4**	**4**
150	**Avg (s)**	59892	59407	59183	**59081**	59094	59113	59230	59114
	Dev (%)	2.67%	1.49%	0.84%	0.72%	0.66%	**0.63%**	0.86%	0.67%
	#Best	0	1	2	2	**5**	2	1	3
200	**Avg (s)**	78173	77922	77474	77581	77590	77490	77670	**77385**
	Dev (%)	1.80%	1.12%	0.45%	0.54%	0.56%	0.45%	0.56%	**0.40%**
	#Best	0	0	2	1	3	2	2	**6**
250	**Avg (s)**	96846	96351	**96224**	96362	96396	96297	96421	96647
	Dev (%)	1.17%	0.55%	0.41%	0.51%	0.41%	**0.40%**	0.66%	0.71%
	#Best	0	**4**	3	3	2	**4**	0	0

Table 11: Behavior of the completion time when varying the number of orders (congestion), for several time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the GVNS batching algorithm and the S-Shape routing strategy on the instances extracted from Dataset #1 (Albareda-Sambola et al., 2009).

		Maximum turnover time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
40	**Avg (s)**	17100	13604	11433	11847	10025	9628	10458	**9297**
	Dev (%)	154.41%	100.03%	54.76%	60.19%	22.83%	18.30%	23.75%	**7.97%**
	#Best	0	1	2	0	3	3	1	**6**
60	**Avg (s)**	22006	20437	19820	19586	18945	19005	19022	**18690**
	Dev (%)	38.42%	24.04%	19.27%	18.84%	15.23%	13.76%	16.55%	**9.76%**
	#Best	1	0	2	1	2	**5**	2	3
80	**Avg (s)**	27653	27940	26877	27079	27286	27065	**26781**	26917
	Dev (%)	15.17%	15.15%	11.55%	11.85%	12.51%	11.45%	**10.00%**	11.67%
	#Best	**3**	0	1	**3**	1	**3**	**3**	2
100	**Avg (s)**	34425	34161	33085	33300	33989	32864	**32452**	33536
	Dev (%)	15.08%	13.09%	9.29%	10.21%	12.46%	8.17%	**6.46%**	10.48%
	#Best	1	0	1	3	0	**4**	2	5

Table 12: Behavior of the maximum turnover time when varying the number of orders (congestion), for several time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the GVNS batching algorithm and the S-Shape routing strategy on the instances extracted from Dataset #2 (Henn, 2012).

		Maximum turnover time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
50	**Avg (s)**	23219	21542	17319	16898	15952	15490	**14223**	15420
	Dev (%)	361.54%	311.83%	144.50%	128.32%	33.13%	23.38%	**4.51%**	9.54%
	#Best	0	0	1	2	0	4	**7**	4
100	**Avg (s)**	37576	36891	34410	35450	**34145**	34973	34912	34827
	Dev (%)	48.68%	42.05%	18.58%	23.60%	**9.18%**	10.92%	15.57%	14.37%
	#Best	0	0	3	2	**4**	2	**4**	1
150	**Avg (s)**	56722	55610	55806	54965	55661	**55196**	55506	55520
	Dev (%)	8.80%	5.59%	4.83%	5.17%	4.28%	**3.42%**	4.62%	4.59%
	#Best	0	1	1	2	3	**6**	0	3
200	**Avg (s)**	75048	74190	**73723**	74862	74633	74113	74592	74075
	Dev (%)	6.22%	5.10%	**3.11%**	5.29%	4.78%	4.88%	4.65%	3.98%
	#Best	0	2	**4**	1	2	3	1	3
250	**Avg (s)**	93737	92925	93034	93645	93731	**92555**	92986	93099
	Dev (%)	4.08%	2.52%	2.19%	2.97%	2.65%	**2.08%**	3.03%	2.17%
	#Best	1	1	1	1	3	**4**	3	2

Table 13: Behavior of the maximum turnover time when varying the number of orders (congestion), for several time-window strategies (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) using the GVNS batching algorithm and the S-Shape routing strategy on the instances extracted from Dataset #1 (Albareda-Sambola et al., 2009).

Notice that the congestion in the system increases when, on the same time horizon, the number of orders received is increased. Thus, it is observed that the picking time, the completion time, and the maximum turnover time also increase with the increase of congestion.

For the same number of orders, in all evaluated scenarios, we observe significant differences between the worst and best time-window methods. On

the other hand, the differences between the methods are greater with fewer orders in the same time horizon. As far as objective functions are concerned, it can be stated that the differences between the best and worst methods are more accentuated in the case of minimizing the maximum turnover time. This situation can be partially explained by the fact that it looks for the minimization of a maximum value.

Finally, in the case of minimizing the picking time, regardless of the size of the instance, the best time-window method was VTW_B3. Similarly, for completion time, the most competitive methods were: VTW_M2, FTW_ZH, and FTW_NW. Finally, in the case of maximum turnover time, the best methods were: VTW_M1, VTW_M2, FTW_ZH, and FTW_NW.

6.3.5. Statistical analysis

To end this empirical comparison, we performed several statistical tests to corroborate if the differences found among the methods are statistically significant. We have compiled all the different executions performed in this final experimentation, and we have performed a Friedman Rank Test. The obtained p_value of 0.000 indicates that there are differences among the methods. In Table 14 we report the rank reported by the test, where we can observe that VTW_M2 is ranked at the first position, while VTW_B3 is ranked the last one. Further than the overall differences, we would like to observe if there are differences between the methods that are closely ranked. To that aim we have performed a pairwise comparison of the methods using the Wilcoxon's Signed Test. In Table 15 we report the obtained p_value for each comparison. As we can observe, the two variants of the proposed methods in this paper VTW_M2 and VTW_M1 do not show statistically significant differences (with a p_value higher than 0.05) between them, however, there are differences between the two proposed methods and any other of the compared ones, including the third one in the ranking VTW_HE. Also, we observe that there are almost no differences between: VTW_HE and VTW_B1, VTW_B1 and VTW_NW, and VTW_NW and VTW_ZH.

Finally, we would like to remark that the fact that VTM_M2 is ranked in the first position when considering all the compared scenarios together indicates a good average behavior regardless of the pursued objective function. However, we repeated the Friedman Rank Test comparing only the scenarios related to each of the objective functions separately. In this case, when ranking the methods according to the picking time, the VTW_B3 method ranked in the first position. Additionally, when ranking the methods according to

either the completion time or the turnover time, the FTW_NW ranked in the first position. In those scenarios, our best proposal (VTW_M2) ranked (6/8) in picking time, (2/8) in completion time, and (3/8) in turnover time.

Method	Rank
VTW_M2	4,09
VTW_M1	4,12
VTW_HE	4,26
VTW_B1	4,29
FTW_ZH	4,38
FTW_NW	4,46
VTW_B2	4,96
VTW_B3	5,44

Table 14: Friendman's Rank Test.

	VTW_B3	VTW_B2	VTW_B1	VTW_HE	VTW_M1	VTW_M2	FTW_NW	FTW_ZH
VTW_B3	-	0,000	0,000	0,000	0,000	0,000	0,000	0,000
VTW_B2	0,000	-	0,000	0,000	0,000	0,000	0,000	0,000
VTW_B1	0,000	0,000	-	0,518	0,002	0,000	0,101	0,003
VTW_HE	0,000	0,000	0,518	-	0,013	0,003	0,022	0,000
VTW_M1	0,000	0,000	0,002	0,013	-	0,629	0,000	0,000
VTW_M2	0,000	0,000	0,000	0,003	0,629	-	0,000	0,000
FTW_NW	0,000	0,000	0,101	0,022	0,000	0,000	-	0,172
FTW_ZH	0,000	0,000	0,003	0,000	0,000	0,000	0,172	-

Table 15: Comparison of each pair of methods using the Wilcoxon's Signed test.

7. Conclusions and future work

In this work, we have studied the influence of the time-window strategy on the performance of the algorithms for the OOBP. Specifically, we have reviewed the evolution of the concept of time window in the context of the OOBP during the years. Additionally, we have reviewed the most relevant previous strategies for determining the time window in the literature and selected the most prominent ones to be empirically evaluated. Also, we have proposed two new variable time-window strategies.

Our main conclusion is that it can be stated that there is a relevant influence of the time-window algorithm on the performance of the methods

for the OOBP when any of the following objective functions are studied: Picking time, completion time, or maximum turnover time. Also, we have identified that the time-window algorithm should be determined depending on the objective function tackled, since there is no efficient time-window algorithm for all objective functions.

We have also demonstrated that the influence of the strategy to determine the time window on the objective function does not depend on the batching or routing methods used. It has also been shown that the selection of the best time window for each scenario does not depend on the number of orders processed. However, when considering multiple pickers, the larger the number of pickers, the greater the differences among the compared methods, when minimizing the picking time or the turnover time. On the other hand, increasing the number of pickers results in a slight decrease of the differences among the compared methods, when studying the completion time.

The results obtained in the experimentation carried out during this study indicate that, in online contexts, the minimization of the picking time can be in conflict with the minimization of the completion time. This is supported by the fact that, when minimizing the picking time, we look for batches that can be efficiently collected. To this end, it might be necessary to wait longer for the arrival of new orders, so batches can be better composed. On the contrary, waiting longer for the arrival of new orders implies that the total picking time is increased. This circumstance does not occur in offline contexts where the picking time and the completion time are aligned. Therefore, in future works related to the OOBP, the completion time and the picking time could be studied using a multiobjective approach.

The results obtained in this work indicate the importance of properly defining the time-window strategy within the OOBP context. On average, we observed that for a single picker and an arrival time horizon of 4 hours, it is possible to save up to 17 minutes in the picking time, 21 minutes in the completion time, and 49 minutes in the maximum turnover time, simply by choosing an appropriate time-window algorithm. Furthermore, some of the above savings increase as the number of pickers also increases. For example, with 5 pickers, the differences in the picking time, choosing one or the other time-window method, can be up to 311 minutes (> 5 hours).

Among the methods evaluated for determining the time window, when minimizing the picking time, it is recommended to use the VTW_B3 method, while in the case of minimizing the completion time or the maximum turnover time, it is recommended to use one of the following methods: VTW_M1,

VTW_M2, FTW_ZH, or FTW_NW. Despite the fact that some methods in the literature perform very well for specific objective functions, the methods proposed in this paper (VTW_M1 and VTW_M2) have a very good behavior for any of the objective functions studied in this paper.

Finally, we would like to highlight the importance of choosing adequately the time-window method depending on the studied objective function, but also on the context conditions (such as the number of pickers or the congestion of the system). On the contrary, the selection of a time-window method does not depend on the batching or routing method chosen. Given the relevance of the time window, it might be interesting to explore new time-window methods and their influence on other objective functions related to the OOBP, not studied in this work, such as minimizing the blocking time or minimizing the cost. Additionally, it would be interesting to study the effect of the dynamic update of the batches, even if the tour to collect them has already started.

Appendix A. Detailed results of the influence of time-window algorithms on the performance of the studied methods for different objective functions, when combined with different routing strategies

		Picking time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
S-Shape	**Avg (s)**	**44017**	44071	44277	44295	44564	44611	45152	45115
	Dev (%)	**0.20%**	0.64%	1.95%	2.03%	3.57%	4.33%	7.65%	7.57%
	#Best	**76**	41	9	5	5	5	3	4
Largest-Gap	**Avg (s)**	**43730**	43818	44043	44024	44284	44337	44790	44817
	Dev (%)	**0.15%**	0.76%	2.04%	2.06%	3.54%	4.43%	7.47%	7.46%
	#Best	**93**	27	4	11	4	3	2	2
Combined	**Avg (s)**	**41390**	41483	41694	41673	42519	42020	42499	42497
	Dev (%)	**0.13%**	0.62%	1.91%	1.87%	7.77%	4.35%	7.73%	7.74%
	#Best	**90**	29	6	15	2	1	1	2

Table A.16: Behavior in terms of picking time of a GVNS batching algorithm combined with different routing strategies (S-Shape, Largest-Gap, and Combined) and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

		Completion time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
S-Shape	**Avg (s)**	46911	46268	45706	45728	45662	**45627**	45670	45638
	Dev (%)	6.14%	3.56%	1.18%	1.23%	0.85%	0.73%	0.74%	**0.67%**
	#Best	1	6	12	15	28	**32**	22	29
Largest-Gap	**Avg (s)**	46724	46090	45568	45615	45469	45425	**45406**	45436
	Dev (%)	11.77%	9.10%	6.74%	8.76%	6.25%	6.13%	**5.99%**	6.00%
	#Best	0	0	5	**52**	28	19	16	26
Combined	**Avg (s)**	44450	43856	43324	43296	43183	43210	**43163**	43181
	Dev (%)	6.19%	3.58%	1.25%	1.17%	0.48%	0.67%	**0.45%**	0.47%
	#Best	2	3	15	14	24	**41**	22	30

Table A.17: Behavior in terms of completion time of a GVNS batching algorithm combined with different routing strategies (S-Shape, Largest-Gap, and Combined) and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

		Maximum turnover time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
S-Shape	**Avg (s)**	43054	41922	40612	40848	40485	**40099**	40104	40154
	Dev (%)	72.49%	57.71%	29.79%	29.60%	13.00%	10.71%	9.90%	**8.28%**
	#Best	6	5	16	15	18	**34**	23	29
Largest-Gap	**Avg (s)**	42970	41860	40542	40347	39960	39853	**39804**	39843
	Dev (%)	69.44%	48.88%	30.82%	30.60%	11.43%	11.09%	**9.40%**	9.68%
	#Best	6	7	16	19	**25**	**25**	24	24
Combined	**Avg (s)**	40633	39665	38094	38351	37557	**37477**	37647	37571
	Dev (%)	74.10%	54.93%	30.44%	28.95%	**8.44%**	10.15%	8.80%	8.71%
	#Best	9	8	14	13	23	27	24	**32**

Table A.18: Behavior in terms of maximum turnover time of a GVNS batching algorithm combined with different routing strategies (S-Shape, Largest-Gap, and Combined) and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

Appendix B. Detailed results of the influence of time-window algorithms on the performance of the studied methods for different objective functions, when combined with different batching strategies

		Picking time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
GVNS	**Avg (s)**	**44017**	44071	44277	44295	44564	44611	45152	45115
	Dev (%)	**0.20%**	0.64%	1.95%	2.03%	3.57%	4.33%	7.65%	7.57%
	#Best	**76**	41	9	5	5	5	3	4
ILS	**Avg (s)**	45820	**45752**	45849	45818	46123	46218	46715	46693
	Dev (%)	**1.15%**	1.22%	2.05%	1.96%	3.38%	4.38%	7.60%	7.58%
	#Best	41	**42**	27	26	12	13	4	8
FCFS	**Avg (s)**	**49582**	49595	49671	49671	49854	49932	50486	50486
	Dev (%)	**0.18%**	0.25%	0.68%	0.68%	1.54%	2.35%	5.49%	5.49%
	#Best	**97**	89	72	72	52	51	18	18

Table B.19: Behavior in terms of picking time of different batching algorithms (GVNS, ILS, and FCFS) combined with S-Shape routing strategy and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

		Completion time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
GVNS	**Avg (s)**	46911	46268	45706	45728	45662	**45627**	45670	45638
	Dev (%)	6.14%	3.56%	1.18%	1.23%	0.85%	0.73%	0.74%	**0.67%**
	#Best	1	6	12	15	28	**32**	22	29
ILS	**Avg (s)**	49303	48648	48055	48022	47940	47948	47968	**47934**
	Dev (%)	6.61%	4.08%	1.80%	1.71%	1.11%	**1.08%**	1.14%	1.12%
	#Best	2	5	17	14	**35**	20	26	30
FCFS	**Avg (s)**	52006	51401	50818	50818	50683	**50669**	50734	50734
	Dev (%)	6.06%	3.65%	1.24%	1.24%	0.59%	**0.48%**	0.64%	0.64%
	#Best	0	5	16	19	60	**61**	53	41

Table B.20: Behavior in terms of completion time of different batching algorithms (GVNS, ILS, and FCFS) combined with S-Shape routing strategy and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

		Maximum turnover time							
		VTW_B3	VTW_B2	VTW_B1	VTW_HE	VTW_M1	VTW_M2	FTW_ZH	FTW_NW
GVNS	Avg (s)	43054	41922	40612	40848	40485	**40099**	40104	40154
	Dev (%)	72.49%	57.71%	29.79%	29.60%	13.00%	10.71%	9.90%	**8.28%**
	#Best	6	5	16	15	18	**34**	23	29
ILS	Avg (s)	45231	44122	43062	42963	42150	42380	**42027**	42093
	Dev (%)	68.20%	45.85%	30.75%	28.84%	9.52%	9.58%	**6.92%**	8.12%
	#Best	14	16	18	15	18	20	24	**27**
FCFS	Avg (s)	46737	45408	44108	44108	**43923**	43972	44053	44068
	Dev (%)	65.41%	36.62%	10.82%	10.82%	8.05%	7.64%	**5.71%**	5.76%
	#Best	2	9	22	21	**53**	44	**53**	48

Table B.21: Behavior in terms of maximum turnover time of different batching algorithms (GVNS, ILS, and FCFS) combined with S-Shape routing strategy and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH).

Appendix C. Detailed results of the influence of time-window algorithms on the performance of the studied methods for different objective functions, when varying the number of pickers

		Picking time							
		VTW_B3	VTW_B2	VTW_B1	VTW_HE	VTW_M1	VTW_M2	FTW_ZH	FTW_NW
1 Picker	Avg (s)	**44017**	44071	44277	44295	44564	44611	45152	45115
	Dev (%)	**0.20%**	0.64%	1.95%	2.03%	3.57%	4.33%	7.65%	7.57%
	#Best	**76**	41	9	5	5	5	3	4
2 Pickers	Avg (s)	**44498**	44726	45272	45240	46096	47154	48824	48904
	Dev (%)	**0.15%**	1.26%	3.95%	3.79%	8.17%	15.47%	26.09%	26.25%
	#Best	**110**	23	5	5	0	0	2	0
3 Pickers	Avg (s)	**44924**	45275	46169	46121	47946	50646	53877	54237
	Dev (%)	**0.15%**	1.57%	5.36%	5.20%	12.76%	27.38%	45.96%	46.69%
	#Best	**108**	21	3	6	0	2	6	1
4 Pickers	Avg (s)	**45174**	45604	46822	46797	49531	54327	58518	59401
	Dev (%)	**0.16%**	1.77%	6.30%	6.10%	16.28%	39.18%	62.00%	64.06%
	#Best	**117**	11	5	1	0	1	9	0
5 Pickers	Avg (s)	**45220**	45773	47074	47043	50588	57319	62291	63918
	Dev (%)	**0.15%**	1.91%	6.56%	6.60%	18.53%	47.80%	74.45%	78.13%
	#Best	**118**	14	1	2	1	0	9	0

Table C.22: Behavior in terms of picking time of a GVNS batching algorithm, combined with S-Shape routing strategy and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) when varying the number of pickers.

		Completion time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
1 Picker	**Avg (s)**	46911	46268	45706	45728	45662	**45627**	45670	45638
	Dev (%)	6.14%	3.56%	1.18%	1.23%	0.85%	0.73%	0.74%	**0.67%**
	#Best	1	6	12	15	28	**32**	22	29
2 Pickers	**Avg (s)**	27061	26501	26095	26125	25954	25869	25840	**25819**
	Dev (%)	8.07%	4.98%	2.66%	2.80%	1.75%	1.24%	1.02%	**0.83%**
	#Best	1	1	12	10	32	28	**34**	**34**
3 Pickers	**Avg (s)**	21568	21169	20889	20874	20743	20549	20555	**20505**
	Dev (%)	7.58%	5.22%	3.52%	3.49%	2.56%	1.35%	1.17%	**0.89%**
	#Best	0	1	9	7	18	**48**	47	40
4 Pickers	**Avg (s)**	19334	19113	18837	18810	18625	18463	18431	**18309**
	Dev (%)	7.02%	5.79%	4.04%	3.90%	2.61%	1.58%	1.13%	**0.54%**
	#Best	2	0	4	9	23	32	49	**58**
5 Pickers	**Avg (s)**	18141	18003	17748	17721	17522	17371	17256	**17181**
	Dev (%)	6.72%	5.87%	4.28%	4.13%	2.76%	1.71%	0.84%	**0.45%**
	#Best	1	0	5	5	21	29	**65**	59

Table C.23: Behavior in terms of completion time of a GVNS batching algorithm, combined with S-Shape routing strategy and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) when varying the number of pickers.

		Maximum turnover time							
		VTW_B3	**VTW_B2**	**VTW_B1**	**VTW_HE**	**VTW_M1**	**VTW_M2**	**FTW_ZH**	**FTW_NW**
1 Picker	**Avg (s)**	43054	41922	40612	40848	40485	**40099**	40104	40154
	Dev (%)	72.49%	57.71%	29.79%	29.60%	13.00%	10.71%	9.90%	**8.28%**
	#Best	6	5	16	15	18	**34**	23	29
2 Pickers	**Avg (s)**	22547	20602	18766	18843	17451	17388	17310	**17208**
	Dev (%)	226.61%	154.18%	93.41%	95.33%	25.49%	19.86%	11.67%	**10.54%**
	#Best	5	2	10	14	19	20	37	**51**
3 Pickers	**Avg (s)**	16200	14191	12055	12199	10259	10175	10254	**9910**
	Dev (%)	354.37%	249.60%	138.32%	145.16%	32.72%	25.01%	15.95%	**6.14%**
	#Best	1	4	7	6	20	23	49	**62**
4 Pickers	**Avg (s)**	13649	11733	9390	9543	7594	7248	7415	**6857**
	Dev (%)	421.39%	313.18%	175.75%	177.23%	43.66%	32.19%	31.63%	**4.41%**
	#Best	4	1	4	5	16	16	65	**75**
5 Pickers	**Avg (s)**	12333	10667	8155	8115	6017	5798	5818	**5358**
	Dev (%)	479.54%	379.97%	204.43%	199.38%	51.45%	37.71%	35.19%	**3.15%**
	#Best	1	1	3	2	11	21	**81**	78

Table C.24: Behavior in terms of maximum turnover time of a GVNS batching algorithm, combined with S-Shape routing strategy and different time-window algorithms (VTW_B1, VTW_B2, VTW_B3, VTW_HE, VTW_M1, VTW_M2, FTW_NW, and FTW_ZH) when varying the number of pickers.

References

Aerts, B., Cornelissens, T., Sörensen, K., 2021. The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. Computers & Operations Research 129, 105168.

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, 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, 101–107.

Bukchin, Y., Khmelnitsky, E., Yakuel, P., 2012. Optimizing a dynamic order-picking process. European Journal of Operational Research 219, 335–346.

Chew, E.P., Tang, L.C., 1999. Travel time analysis for general item location assignment in a rectangular warehouse. European Journal of Operational Research 112, 582–597.

Coyle, J.J., Bardi, E.J., Langley, C.J., 1996. The management of business logistics. volume 6. West Publishing Company Minneapolis/St. Paul.

De Koster, R.B.M., Van Der Poort, E.S., 1998. Routing order pickers in a warehouse: a comparison between optimal and heuristic solutions. IIE Transactions 30, 469–480.

Drury, J., 1988. Towards more efficient order picking. IMM monograph 1.

Duda, J., Stawowy, A., 2019. A VNS Approach for Batch Sequencing and Route Planning in Manual Picking System with Time Windows. Lecture Notes in Computer Science. International Conference on Variable Neighborhood Search 12010, 167–177.

Fontin, J.R., Lin, S.W., 2020. A joint comparative analysis of routing heuristics and paperless picking technologies using simulation and data envelopment analysis. Applied Sciences 10, 8777.

Gademann, N., Velde, V.S., 2005. Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions 37, 63–75.

Giannikas, V., Lu, W., Robertson, B., Mc. Farlane, D., 2017. An interventionist strategy for warehouse order picking: Evidence from two case studies. International Journal of Production Economics 189, 63–76.

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, 295–325.

Gil-Borrás, S., Pardo, E.G., Alonso-Ayuso, A., Duarte, A., 2021. A heuristic approach for the online order batching problem with multiple pickers. Computers & Industrial Engineering 160, 107517.

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

Henn, S., 2012. Algorithms for on-line order batching in an order picking warehouse. Computers and Operations Research 39, 2549–2563.

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, 82–105.

Il-Choe, K., Sharp, G.P., 1991. Small parts order picking: design and operation. Technical Report. School of Industrial and Systems Engineering. Georgia Institute of Technology. Atlanta, Georgia, EEUU.

Koch, S., Wäscher, G., 2016. A grouping genetic algorithm for the Order Batching Problem in distribution warehouses. Journal of Business Economics 86, 131–153.

Leung, K.H., Lee, C.K.M., Choy, K.L., 2020. An integrated online pick-to-sort order batching approach for managing frequent arrivals of b2b e-commerce orders under both fixed and variable time-window batching. Advanced Engineering Informatics 45, 101–125.

Lourenço, H.R., Martin, O.C., Stützle, T., 2003. Iterated local search. International Series in Operations Research & Management Science. Handbook of metaheuristics 57, 320–353.

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

Menéndez, B., Pardo, E.G., Alonso-Ayuso, A., Molina, E., Duarte, A., 2017b. 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.

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

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

Petersen, C.G., 1995. Routeing and storage policy interaction in order picking operations. Decision Science Institute Proceedings 31, 1614–1616.

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

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

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

Poisson, S.D., 1837. Recherches sur la probabilité des jugements en matière criminelle et en matière civile: précédées des règles générales du calcul des probabilités. Bachelier.

Quinn, E.B., 1983. Simulation of order processing waves. Material Handling Focus 83.

Rushton, A., Croucher, P., Baker, P., 2022. The Handbook of Logistics and Distribution Management: Understanding the supply chain. Kogan Page Limited.

Shah, B., Khanzode, V., et al., 2017. A comprehensive review of warehouse operational issues. International Journal of Logistics Systems and Management 26, 346–378.

Tang, L.C., Chew, E.P., 1997. Order picking systems: Batching and storage assignment strategies. Computers & Industrial Engineering 33, 817 – 820.

Van Nieuwenhuyse, I., De Koster, R.B.M., 2009. Evaluating order throughput time in 2-block warehouses with time window batching. International Journal of Production Economics 121, 654–664.

Xu, X., Liu, T., Li, K., Dong, W., 2014. Evaluating order throughput time with variable time window batching. International Journal of Production Research 52, 2232–2242.

Yu, M.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, 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, 271 – 284.

Zhang, J., Wang, X., Huang, K., 2016. Integrated on-line scheduling of order batching and delivery under b2c e-commerce. Computers & Industrial Engineering 94, 280 – 289.

Chapter 7

Conclusions and future work

In this chapter, we present the conclusions and future work of this Doctoral Thesis. First, in the Section 7.1, we present the general conclusions of the thesis and also the conclusions related of each variant of the problems tackled. Then, in Section 7.2, we present the future work that can be devised from this Doctoral Thesis. Finally, in Section 7.3, we summarize the publications obtained as the result of the research carried out. Finally,

7.1 Conclusions

In this Doctoral Thesis, we have tackled several variants of the Online Order Batching Problem with a single picker and with multiple pickers. For each variant, we have deeply studied the state of the art of the problem and proposed new strategies which outperformed previous ones in the literature. We have studied four different objective functions for the family of problems studied such as: minimizing the picking time, minimizing the total completion time, minimizing the maximum turnover time, and minimizing the differences in the workload of the pickers. In addition, we have analyzed the behavior of our algorithms by analyzing the influence of the increase of congestion in the rate of orders that arrive to the system; and the suitable number of pickers for the collection of orders depending on other parameters of the warehouse. This analysis has allowed us to understand the foundations of the problem for improving the performance of the proposed algorithms, depending on the warehouse characteristics and objectives pursued. It also allowed us to provide insights about the configuration of some operational parameters of the warehouse.

Among the processes involved in the resolution of the Online Order Batching Problem, we have identified an important task which, surprisingly, has been neglected by practitioners in the field: the time window determination. As we could observe, the waiting time before start a new picking route is a key factor in the performance of online systems, where the arrival of orders is continuous. This process has a deep impact in the batching task since waiting might provide new orders which can help to perform an improved batch configuration.

From a managerial point of view, the optimization problems studied in this Doctoral Thesis might result in an increase of the benefits and quality service of the picking operations in a warehouse. Each objective function studied improves different aspects of order picking systems. In particular, reducing the picking time reduces energy consumption, while reducing the completion time or the maximum turnover time results in faster product service for customers. Similarly, the balance of workload results in a healthier and safer work environment and prevents the overload of the machinery.

Furthermore, the state-of-the-art work in this Doctoral Thesis has greatly contributed to the field of the optimization of processes related to Order Batching. Particularly, in the literature related to order batching, we can find more than one hundred publications in top-level journals.

However, there has been little work in terms of sorting and classifying those papers. This fact has obstructed researchers to identify previous papers in the literature that are directly related to their research and, furthermore, to identify gaps in the field not visited yet. For instance, we have identified many papers lacking of a proper comparison of their findings with other previous proposals. Similarly, we have identified 36 different variants of the OBP, however only 18 of them have been studied. To overcome these problems, we proposed a new taxonomy which compiles all the variants of the problem, on the basis of the processes involved in the problem. Also we have compiled the state of the art related to any of the variants and we have highlighted those which have never been tackled before. The taxonomy proposed is designed to grow. Researchers in the field can extend the taxonomy with new constraints, objective functions, or processes, making any new contribution to the literature clearer. In addition, we emphasize that the family of Order Batching Problems is a growing group of optimization problems of high economic importance to the industry.

7.1.1 Online Order Batching Problem with a Single Picker

The OOBP with a single picker is the first variant of the problem studied in this Doctoral Thesis. This problem is an online variant of the studied family of problems, which present the difficulty that all orders that must be retrieved are not available at the beginning of the process, but they arrive at the warehouse at any time. The problem has been solved for different rectangular warehouses with a single block and a single picker. The studied scenario considers that the arrival of orders occurs in a period of four hours, but the picker might need longer time to collect all the orders arrived in that time horizon.

This variant of the problem is probably the most studied one among the online variants and it is usually devoted to minimize the completion time and/or the turnover time. As the result of the research performed for this problem we have published two journal articles.

The first article, titled "New VNS Variants for the Online Order Batching Problem" ([89]), study the minimization of the maximum time that an order remains in the system before being served. To address this problem, we proposed a Basic Variable Neighborhood Search algorithm. This BVNS was successful in comparison with the state-of-the-art classical greedy constructive methods previously used for other variants of the OBP. Our algorithm was able to improve more than 30% the results obtained by previous heuristics.

The second article for the problem, titled "GRASP with Variable Neighborhood Descent for the online order batching problem" ([92]), can be considered as an extension of the previous one. In this article, we studied two different objective functions. The first objective function looks for the minimization of the maximum completion time, that is, the minimization of how long it takes to collect all orders that arrive at the system. The second objective function looks for the minimization of the maximum turnover time, i.e. the minimization of the maximum time that an order remains in the system. In this case, the algorithmic proposal was a combination of two metaheuristics for the batching task: i) GRASP used as a constructive method to generate a different initial solution for each execution; ii) VND used as a local search procedure. For the routing task, we proposed an S-shaped algorithm to calculate the route to collect the orders grouped in each batch. In this case, our algorithmic proposal improved the state of the art by more than 3%, on average, for the first objective function an by more than 6%, on average, for the second objective function. In both cases, the significance of the results has been corroborated by statistical tests. More specifically, we used the t-test for the difference of means in paired samples. The results of this test support the conclusion that there are significant differences between the results found.

The study of this variant of the OOBP has devised the identification of several successful strategies. First, we observed that the use of several neighborhoods is an important strategy

when dealing with the batching task to avoid the method to escape from local optima. This is due to the fact that, when dealing with real warehouses, the size of the batch is reduced and, therefore, it is not always easy to perform moves within the available space in batches. In this sense many methods easily fall in local optima and the use of alternative neighborhoods helps to avoid them.

Also, we observed that reducing the number of batches results in an improvement in the objective functions studied. Therefore, an alternative objective function which helps to optimize others, consists of minimizing the the wasted space in each batch. To achieve this goal, a naive but successful strategy to generate the initial solution consists of sorting the orders in descending size/weight, so the larger orders are assigned first to a batch and the smaller orders might complete the partially completed batches.

7.1.2 Online Order Batching Problem with Multiple Pickers

The Online Order Batching Problem with multiple pickers can be considered as the general case of the version with a single picker. As in the single picker variant, online variants present the difficulty that all orders to retrieved are not available at the beginning of the process, but they arrive at the warehouse at any time. In this case, the collection of orders is carried out by several pickers simultaneously.

We have studied the problem for instances representing rectangular-shaped warehouses with a single block. In this context, we considered different time horizons for the arrival of orders and different number of pickers. Also, we have studied the influence of the congestion in the performance of the algorithms. As a result of our research, we published two journal articles for this variant.

The first article, titled "Basic VNS for a Variant of the Online Batching Problem" studied the minimization of the total time needed to collect all orders. Particularly, in this paper, we tackled this problem for two pickers. To that aim, we proposed several search heuristics in a BVNS framework, which was successfully compared with several classical seed algorithms presented in the literature of the problem [304]. In this article, we proposed to compare three well-known routing algorithms (S-Shape, Largest-Gap, and Combined). We also noticed that the use of the Combined routing method was the most effective among the ones considered for this problem. Overall, the results were very promising, since we obtained an improvement between 0.60% and 5% depending on the routing algorithm used.

This second article published for this variant was titled "A heuristic approach for the online order batching problem with multiple pickers". In this article, we studied three objective functions: minimizing completion time, minimizing picking time, and minimizing differences in workload among pickers. We expand the number of objective functions studied since previous papers do not report all of them. Also, note that the difference between the completion time and the picking time is that the completion time include waiting times. We have proposed two heuristic approaches based on a multi-start VNS to tackle the problem considering all identified objective functions for the problem. Our approaches have been successfully compared with previous methods of the state-of-the-art. We compared the results obtained using statistical tests and found that there are significant differences between them. We studied scenarios varying the number of pickers, the congestion of the system, and the objective function.

We typically use the objective function to guide the search procedures. However, this paper studies more than one objective function at the same time. For this reason, in this work, we study two different guide functions (workload balance and picking time). We observed that both guide functions are good at reducing the completion time.

We detected that the number of pickers available in the system and the congestion are closely related factors and have a strong influence on the efficiency of order picking strategies.

Particularly, increasing the number of pickers results in a shorter completion time and lower workload for each picker. However, the total picking time increases since there are more pickers collecting fewer items on each collection route. This results in an increase in the energy consumption. When the congestion rate in the arrival of orders increases, we observed that the picking time and the completion time also increase, but the workload balance remains stable. To design modern and flexible warehouses, we need to determine the number of pickers needed per shift, depending on the warehouse congestion. For example, having fewer pickers than necessary could result in delays in completion time. However, having more pickers than necessary might result in more dead time in the warehouse and worsening of the picking time.

7.1.3 Online Order Batching Problem with Time Window

Online arrival of orders means that not all orders that must be retrieved are available at the beginning of the picking process. In this scenario, once a batch is configure, the use of a time window means to wait for a particular period of time before start the picking. During this waiting time, new orders might arrive at the warehouse and a better batch configuration can be performed.

Our studies demonstrated that the waiting time has a large impact on the quality of the solutions obtained in the context of the OOBP. The results obtained over the waiting time were published in two papers.

The first article, titled "Fixed versus variable time window warehousing strategies in real time" studied the influence of the waiting time, when minimizing the completion time and the picking time on a single picker scenario. In this case, we compared several well-known Fixed time window, and Variable Time Window strategies of the literature. This work demonstrates the potential of using the Time Window within the context of the OOBP. Additionally, we identified that the choice of the best time window strategy for the problem depends on the objective function evaluated. Also, we noticed that the influence of the time window strategy is greater in elaborated batching strategies than in naive heuristics such as FCFS.

This second article related to time window can be considered as an extension of the previous one and it is titled "A comparative study of the influence of the time-window strategy in the Online Order Batching Problem". In this article, we study three objective functions: minimizing completion time, minimizing picking time, and minimizing maximum turnover time, and six well-known time window strategies. Additionally, we introduce two new time-window strategies for the problem. Here, we demonstrated the existence of the independence of the waiting method with respect to the batching and routing strategies used. In addition, the two new time-window methods proposed improved the methods previously presented in the state of the art in various scenarios.

We found that the choice of the time-window algorithm should be based on the objective function being optimized, since there is no efficient time-window algorithm for all objective functions.

As we have already pointed out, the minimization of the picking time can be in conflict with the minimization of the completion time in online contexts. This happens because the longer the waiting time, the fewer space in the generated batches, and thus they are of better quality, and then picking time improves. However, a longer waiting time only affects the completion time, by increasing it. I would like to remember that the completion time is the picking time together with the waiting times.

To point out the importance of Time Window in the OOBP, we present some general results obtained in the experimentation performed. On average, we observed that for a single picker and an arrival time horizon of 4 hours, it is possible to save up to 17 minutes in the picking time, 21 minutes in the completion time, and 49 minutes in the maximum turnover time, simply

by choosing an appropriate time-window algorithm. Furthermore, some of the above savings increase as the number of pickers also increases.

7.2 Future work

The research carried out in this Doctoral Thesis opens the door to several future works related to the OOBP. Particularly, the order batching is a large family of optimization problems which opens new research opportunities for practitioners in the field.

First, it is important to highlight that each of the papers obtained as the result of the research of this Doctoral Thesis include future work. However, and generally speaking, we believe that future research in the context of order batching should focus in more realistic scenarios, such as: online arrival of orders, existence of multiple pickers, and existence of multiple objectives. The development of new algorithms should also consider the simultaneous resolution of several tasks. In this sense, we have identified that half of the variants of the problem, have never been studied in the literature, which results in a large research opportunity. In particular, most of the online joint versions of the problem have never been studied.

Additionally, we have identified only a few studies on multi-objective optimization variants of order batching problems. We propose to advance in this direction. For example, researchers could simultaneously address the picking time and completion time in a context with time window. In general, optimizing several objective functions at the same time will provide companies with non-dominated sets of solutions that could be useful in different applied scenarios.

The development of new and more efficient exact algorithms or matheuristics should be explored for scenarios with low congestion and small size of instances.

Another new field to explore is the proposal of new time-window methods and the study of their influence on other objective functions related to the OOBP, not studied yet, such as minimizing the blocking time or minimizing the cost. Additionally, it is also interesting to study new variants of the problem which consider a dynamic update of the batches once the tour to collect them has already started, or the possibility of splitting the orders in different batches.

7.3 Publications

In this section we present a summary of the publications obtained as the result of this Doctoral Thesis. Particularly, we present the publications in two different ways. First, in Table 7.1 we have grouped the publications on the basis of the variant of the problem tackled. For each publication, we compile the year, the type of publication (SJR journal, JCR journal, International conference, or National conference), and the reference.

Second, in Figure 7.1, we introduce a chronological representation of the publications over a timeline with the different milestones reached in this Doctoral Thesis. In this figure, we can find the month and year of each milestone and which variant of the problem/topic is addressed (classified using colors). In addition, we can distinguish the type of publication (Journal / Conference) and, in the case of the journals, the ranking of the journal. In the case of conferences, we can differentiate whether it is a national conference or an international one.

State of the art

2022 [94]	Journal JCR Q1	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A. Order Batching Problems: taxonomy and literature review. Submited to European Journal of Operational Research, **Article under review (2^{nd} round)**, 2022.

Online Order Batching Problem with a Single Picker

2017 [96]	International conference	Gil-Borrás S., Pardo E. G., Alonso-Ayuso A., Duarte A. Online Order batching problem. [p.366] Metaheuristics: Proceeding of the MIC and MAEB 2017 Conferences (MAEB2017). Del 4 al 7 junio de 2017, Barcelona (España). ISBN: 978-84-697-4275-1.
2018 [98]	International conference	Gil-Borrás S., Pardo E. G., Alonso-Ayuso A., Duarte A. New VNS variants for the Online Order Batching Problem. 6th International Conference on Variable Neighborhood Search (ICVNS2018). From 3 to 7 October 2018, Sithonia (Greece).
2018 [97]	National conference	Gil-Borrás S., Pardo E. G., Alonso-Ayuso A., Duarte A. Búsqueda de Vecindad Variable para el problema de la agrupación y recogida de pedidos online en almacenes logísticos. XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA2018). Del 23 al 26 octubre de 2018, Granada (España).
2019 [89]	Journal SJR Q2	Gil-Borrás S., Pardo E. G., Alonso-Ayuso A., Duarte A. New VNS Variants for the Online Order Batching Problem. In: Sifaleras A., Salhi S., Brimberg J. (eds) Variable Neighborhood Search. Lecture Notes in Computer Science, vol 11328. (p. 89-100) Springer, Cham, 2019.
2020 [92]	Journal JCR Q1	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A. GRASP with Variable Neighborhood Descent for the online order batching problem. Journal of global optimization. 78, (p. 295–325), 2020.
2022 [103]	National conference	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A., Matheurísticas aplicadas al problema de recogida de pedidos por lotes en contextos offline. XXXIX Congreso Nacional de Estadística de Investigación Operativa y de las XIII Jornadas de Estadística Pública (SEIO2022). Del 7 al 10 de junio de 2022, Granada (España). ISBN: 978-84-09-41628-8.

Online Order Batching Problem with Multiple Pickers

2019 [217]	International conference	Pardo, E. G., Gil-Borrás, S., Alonso-Ayuso, A., Duarte A. Multipicker Order Batching Problem in Dynamic Environments. 2019 INFORMS-ALIO International Conference (ALIO2019). From 9 to 12 June 2019, Cancún (México).

Continued on the next page

2019 [100]	National conference	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A. Recogida de pedidos por lotes en entornos dinámicos con mulltiples operarios. XXXVIII Congreso Nacional de Estadística e Investigación Operativa (SEIO2019). Del 3 al 6 de septiembre 2019, Alcoy (España). ISBN: 978-84-09-13580-6.
2019 [99]	International conference	Gil-Borrás S., Pardo E. G., Alonso-Ayuso A., Duarte A. Basic VNS for a variant of the Online Order Batching Problem. 7th International Conference on Variable Neighborhood Search (ICVNS2019). From 3 to 7 October 2019, Rabat (Morocco).
2020 [90]	Journal SJR Q3	Gil-Borrás S., Pardo E. G., Alonso-Ayuso A., Duarte A. Basic VNS for a Variant of the Online Order Batching Problem. In: Benmansour R., Sifaleras A., Mladenović N. (eds) Variable Neighborhood Search. Lecture Notes in Computer Science, vol 12010. pp 17-36 Springer, Cham, 2020.
2021 [93]	Journal JCR Q1	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A. A heuristic approach for the online order batching problem with multiple pickers. Computers & Industrial Engineering, 160 (p. 107517), 2021.
		Online Order Batching Problem with Time Window
2020 [91]	Journal SJR Q2	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A. Fixed versus variable time window warehousing strategies in real time. Progress in Artificial Intelligence. (9), (p. 315–324), 2020.
2021 [101]	International conference	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A. Time Window for the Online Order Batching Problem with Variable Neighborhood Search. 8th International Conference on Variable Neighborhood Search ICVNS 2021. From 22 to 24 March, 2021, Abu Dhabi, (U.A.E). ISBN: 978-3-030-69624-5.
2021 [102]	National conference	Gil-Borrás, S., Pardo, E. G., Alonso-Ayuso, A., Duarte A., Jiménez-Merino, E., Algoritmos de estimación del tiempo de ventana en la recogida de pedidos en almacenes logísticos. XIX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA20/21) & XIV Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirado (MAEB 2021). Del 22 al 24 septiembre 2021 Málaga (España). ISBN: 978-84-09-30514-8.
2022 [95]	Journal JCR Q1	Gil-Borrás, S., Pardo, E. G., Jiménez, E. Kenneth, S. A comparative study of the influence of the time-window strategy in online order batching problems. Submited to Computers & Industrial Engineering, **Article under review (2^{nd} round)**, 2022.

Table 7.1: Publications related to Online Order Batching Problem grouped by variant.

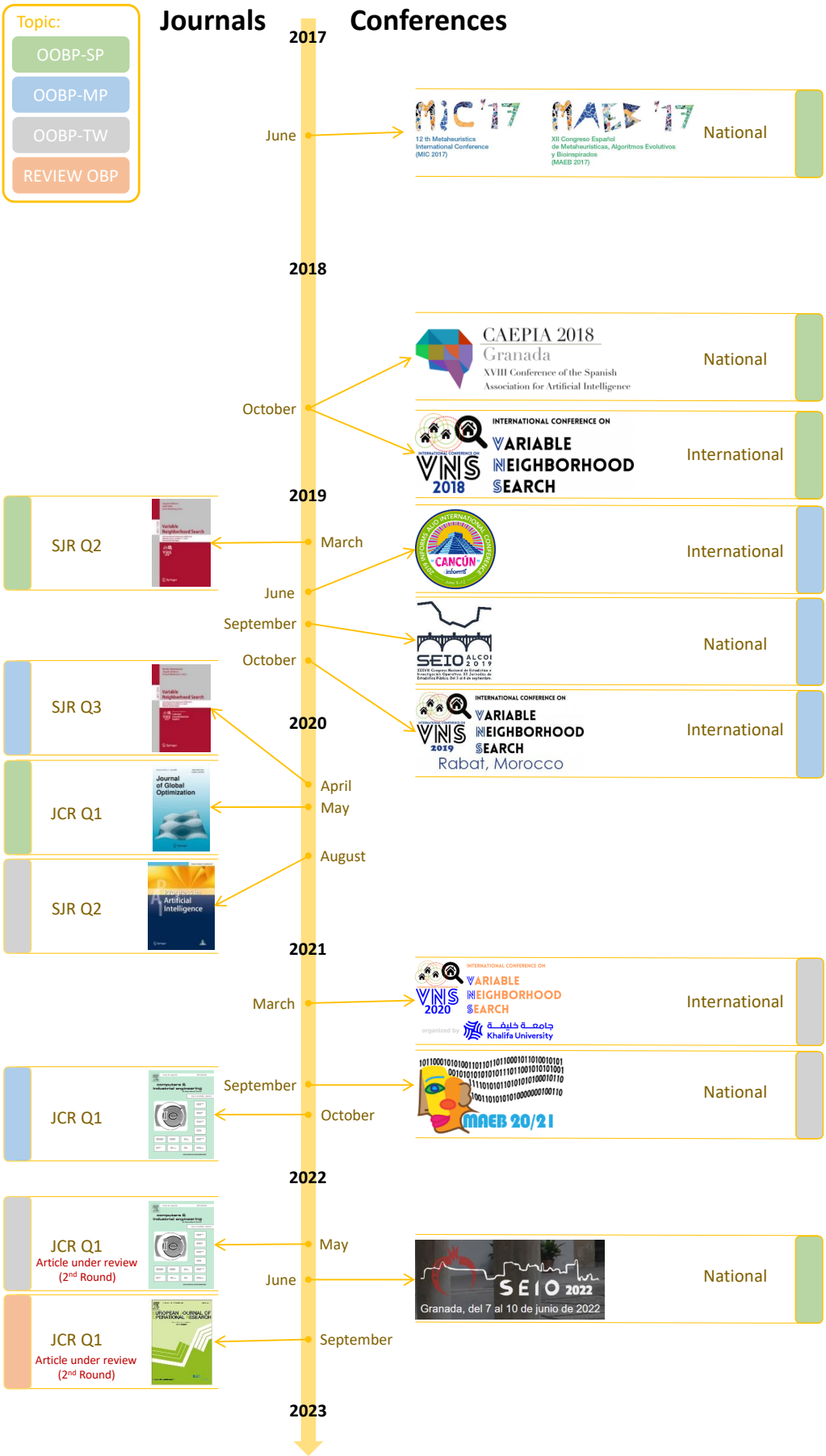


Figure 7.1: Timeline with milestones of the publications obtained in this Doctoral Thesis.

Bibliography

[1] A. H. F. Aboelfotoh. *Optimizing the multi-objective order batching problem for warehouses with cluster picking*. PhD thesis, Ohio University, 2019.

[2] B. Aerts, T. Cornelissens, and K. Sörensen. The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. *Computers & Operations Research*, 129:105168, 2021.

[3] A. R. Ahmadi Keshavarz, D. Jaafari, M. Khalaj, and P. Dokouhaki. Order picking process: State-of-the-art on classification. *Journal of Quality Engineering and Production Optimization*, 6(1):15–48, 2021.

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

[5] M. Alipour, Y. Z. Mehrjedrdi, and A. Mostafaeipour. 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, 2020.

[6] E. Ardjmand, H. Shakeri, M. Singh, and O. S. Bajgiran. Minimizing order picking makespan with multiple pickers in a wave picking warehouse. *International Journal of Production Economics*, 206:169–183, 2018.

[7] E. Ardjmand, O. S. Bajgiran, and E. Youssef. Using list-based simulated annealing and genetic algorithm for order batching and picker routing in put wall based picking systems. *Applied Soft Computing*, 75:106–119, 2019.

[8] E. Ardjmand, I. Ghalehkhondabi, W. A. Young II, A. Sadeghi, R. Y. Sinaki, and G. R. Weckman. A hybrid artificial neural network, genetic algorithm and column generation heuristic for minimizing makespan in manual order picking operations. *Expert Systems with Applications*, 159:113566, 2020.

[9] R. D. Armstrong, W. D. Cook, and A. L. Saipe. Optimal batching in a semi-automated order picking system. *Journal of the operational research society*, 30(8):711–720, 1979.

[10] P. Atchade-Adelomou, G. Alonso-Linaje, J. Albo-Canals, and D. Casado-Fauli. qrobot: A quantum computing approach in mobile robot order picking and batching problem solver optimization. *Algorithms*, 14(7), 2021. ISSN 1999-4893.

[11] M. Y. N. Attari, A. T. Ebadi, B. Malmir, and E. J. Neyshabouri. Robust possibilistic programming for joint order batching and picker routing problem in warehouse management. *International Journal of Production Research*, 59(14):4434–4452, 2021.

[12] A. H. Azadnia, S. Taheri, P. Ghadimi, M. Z. Mat Saman, and K. Y. Wong. Order batching in warehouses by minimizing total tardiness: A hybrid approach of weighted association rule mining and genetic algorithms. *The Scientific World Journal*, 2013(1–13):246578, 2013.

[13] M. S. Bazaraa. *Nonlinear programming : Theory and algorithms*. Wiley, New York, 1979. ISBN 0471786101.

[14] J. Birge. *Introduction to stochastic programming*. Springer, New York Berlin Heidelberg, 2011. ISBN 9781461402367.

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

[16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[17] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 8(1):67–127, 2007.

[18] N. Boysen, R. B. M. De Koster, and F. Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396 – 411, 2019. ISSN 0377-2217.

[19] Y. A. Bozer and J. W. Kile. Order batching in walk-and-pick order picking systems. *International Journal of Production Research*, 46(7):1887–1909, 2008.

[20] J. Branke. *Multiobjective Optimization : Interactive and evolutionary approaches*. Springer-Verlag, Berlin, 2008. ISBN 9783540889076.

[21] O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A. L. Ladier, and M. Ogier. An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, 285(2):497–512, 2020.

[22] M. Bué, D. Cattaruzza, M. Ogier, and F. Semet. An integrated order batching and picker routing problem. *A View of Operations Research Applications in Italy, 2018. AIRO Springer Series*, 2:3–18, 2018.

[23] Y. Bukchin, E. Khmelnitsky, and P. Yakuel. Optimizing a dynamic order-picking process. *European Journal of Operational Research*, 219(2):335–346, 2012.

[24] A. Burinskiene. Order picking process at warehouses. *International Journal of Logistics Systems and Management*, 6(2):162–178, 2010.

[25] B. Cals, Y. Zhang, R. M. Dijkman, and C. Van Dorst. Solving the online batching problem using deep reinforcement learning. *Computers & Industrial Engineering*, 156:107221, 2021.

[26] J. A. Cano. Parameters for a genetic algorithm: An application for the order batching problem. *IBIMA Business Review*, 2019(1–10):802597, 2019.

[27] J. A. Cano, A. A. Correa-Espinal, and R. A. Gómez-Montoya. A review of research trends in order batching, sequencing and picker routing problems. *Revista Espacios*, 39(04), 2018.

[28] J. A. Cano, A. A. Correa-Espinal, and R. A. Gómez-Montoya. Solución del problema de conformación de lotes en almacenes utilizando algoritmos genéticos. *Información tecnológica*, 29(6):235–244, 2018.

[29] J. A. Cano, A. A. Correa-Espinal, and R. A. Gómez-Montoya. Mathematical programming modeling for joint order batching, sequencing and picker routing problems in manual order picking systems. *Journal of King Saud University-Engineering Sciences*, 32(3):219–228, 2020.

[30] J. A. Cano, P. Cortés Achedad, E. A. Campo, and A. A. Correa-Espinal. Solving the order batching and sequencing problem with multiple pickers: A grouped genetic algorithm. *International Journal of Electrical and Computer Engineering*, 11(3):2516–2524, 2021.

[31] Ç. Cergibozan and A. S. Tasan. Order batching operations: An overview of classification, solution techniques, and future research. *Journal of Intelligent Manufacturing*, 30(1): 335–349, 2019.

[32] Ç. Cergibozan and A. S. Tasan. Genetic algorithm based approaches to solve the order batching problem and a case study in a distribution center. *Journal of Intelligent Manufacturing*, pages 1–13, 2020.

[33] T. Chabot, J. F. Côté, J. Renaud, and L. C. Coelho. *Mathematical models, heuristic and exact method for order picking in 3d-narrow aisles*. Faculté des sciences de l'administration, Université Laval, 2015.

[34] F. Chen, H. Wang, C. Qi, and Y. Xie. An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers and Industrial Engineering*, 66(1):77–85, 2013. ISSN 03608352.

[35] F. Chen, H. Wang, Y. Xie, and C. Qi. An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse. *Journal of Intelligent Manufacturing*, 27:389–408, 2016. ISSN 09565515.

[36] F. Chen, Y. Wei, and H. Wang. A heuristic based batching and assigning method for online customer orders. *Flexible Services and Manufacturing Journal*, 30(4):640–685, 2 2018.

[37] M.-C. Chen and H.-P. Wu. An association-based clustering approach to order batching considering customer demand patterns. *Omega*, 33(4):333–343, 2005.

[38] T.-L. Chen, C.-Y. Cheng, Y.-Y. Chen, and L.-K. Chan. An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. *International Journal of Production Economics*, 159:158–167, 2015. ISSN 09255273.

[39] C.-Y. Cheng, Y.-Y. Chen, T.-L. Chen, and J. Jung-Woon Yoo. Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem. *International Journal of Production Economics*, 170:805–814, 2015. ISSN 09255273.

[40] E. P. Chew and L. C. Tang. Travel time analysis for general item location assignment in a rectangular warehouse. *European Journal of Operational Research*, 112(3):582–597, 1999.

[41] S. H. Chung, I. G. Lee, and S. W. Yoon. Two-stage storage assignment to minimize travel time and congestion for warehouse order picking operations. *Computers & Industrial Engineering*, pages 106–129, 2019.

[42] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[43] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[44] D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price. *New ideas in optimization*. McGraw-Hill Ltd., UK, 1999.

[45] J. J. Coyle, E. J. Bardi, and C. J. Langley. *The management of business logistics*, volume 6. West Publishing Company Minneapolis/St. Paul, 1996.

[46] H. Crowder, E. L. Johnson, and M. W. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[47] J. Culberson. Iterated greedy graph coloring and the difficulty landscape. Technical report, University of Alberta, 1992.

[48] J. C. Culberson and F. Luo. Exploring the k-colorable landscape with iterated greedy. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, 26:245–284, 1996.

[49] P. Damodaran, O. Ghrayeb, and M. C. Guttikonda. Grasp to minimize makespan for a capacitated batch-processing machine. *The International Journal of Advanced Manufacturing Technology*, 68(1):407–414, 2013.

[50] R. B. M. De Koster. Distribution strategies for online retailers. *IEEE Transactions on Engineering Management*, 50(4):448–457, 2003.

[51] R. B. M. De Koster and E. S. Van Der Poort. Routing order pickers in a warehouse: A comparison between optimal and heuristic solutions. *IIE Transactions*, 30(5):469–480, May 1998.

[52] R. B. M. De Koster, K. J. Roodbergen, and R. Van Voorden. Reduction of walking time in the distribution center of De Bijenkorf. *Lecture Notes in economics and mathematical systems. New Trends in Distribution Logistics*, 480:215–234, 1999.

[53] R. B. M. De Koster, E. S. Van Der Poort, and M. Wolters. Efficient order batching methods in warehouses. *International Journal of Production Research*, 37(7):1479–1504, 1999.

[54] R. B. M. De Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European journal of operational research*, 182(2): 481–501, 2007.

[55] M. Den Besten, T. Stützle, and M. Dorigo. Design of iterated local search algorithms. In *Workshops on Applications of Evolutionary Computation*, pages 441–451. Springer, 2001.

[56] M. Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*, 1992.

[57] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[58] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for hard optimization: Methods and case studies*. Springer Science & Business Media, 2006.

[59] J. Drury. Towards more efficient order picking. *IMM monograph*, 1, 1988.

[60] A. Duarte, J. J. Pantrigo, E. G. Pardo, and J. Sánchez-Oro. Parallel variable neighbourhood search strategies for the cutwidth minimization problem. *IMA Journal of Management Mathematics*, 27(1):55–73, 2013.

[61] A. Duarte, J. J. Pantrigo, E. G. Pardo, and N. Mladenović. Multi-objective variable neighborhood search: An application to combinatorial optimization problems. *Journal of Global Optimization*, 63(3):515–536, 2015.

[62] A. Duarte, N. Mladenovic, J. Sánchez-Oro, and R. Todosijević. Variable neighborhood descent, 2018.

[63] J. Duda and A. Stawowy. A VNS Approach for batch sequencing and route planning in manual picking system with time windows. *Lecture Notes in Computer Science. International Conference on Variable Neighborhood Search*, 12010:167–177, 2019.

[64] R. Duffin. *Geometric programming : Theory and application*. John Wiley, New York, 1967.

[65] G. Dukic and C. Oluic. Order-picking methods: Improving order-picking efficiency. *International Journal of Logistics Systems and Management*, 3(4):451–460, 2007.

[66] G. Dukic and T. Opetuk. Warehouse layouts. In *Warehousing in the Global Supply Chain*, pages 55–69. Springer, 2012.

[67] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pages 39–43. Ieee, 1995.

[68] M. Ehrgott. *Multicriteria optimization*. Springer, Berlin New York, 2005. ISBN 9783540213987.

[69] E. A. Elsayed. Algorithms for optimal material handling in automatic warehousing systems. *The International Journal of Production Research*, 19(5):525–535, 1981.

[70] E. A. Elsayed and M. K. Lee. Order processing in automated storage/retrieval systems with due dates. *IIE transactions*, 28(7):567–577, 1996.

[71] E. A. Elsayed and O. I. Unal. Order batching algorithms and travel-time estimation for automated storage/retrieval systems. *The International Journal of Production Research*, 27(7):1097–1114, 1989.

[72] E. A. Elsayed, M. K. Lee, S. Kim, and E. Scherer. Sequencing and batching procedures for minimizing earliness and tardiness penalty of order retrievals. *The International Journal of Production Research*, 31(3):727–738, 1993.

[73] S. Ene and N. Öztürk. Storage location assignment and order picking optimization in the automotive industry. *The international journal of advanced manufacturing technology*, 60 (5):787–797, 2012.

[74] B. Esra and A. Nil. The order batching problem: A state-of-the-art review. *Sigma Journal of Engineering and Natural Sciences*, 40(2):402–420, 2022.

[75] X. Feng and X. Hu. A heuristic solution approach to order batching and sequencing for manual picking and packing lines considering fatiguing effect. *Scientific Programming*, 2021(1–17):8863391, 2021.

[76] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 03 1995.

[77] T. A. Feo, M. G. C. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.

[78] D. Filson. The impact of e-commerce strategies on firm value: Lessons from amazon. com and its early competitors. *The Journal of Business*, 77(S2):S135–S154, 2004.

[79] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*, 2013.

[80] D. B. Fogel and L. J. Fogel. An introduction to evolutionary programming. In *European conference on artificial evolution*, pages 21–33. Springer, 1995.

[81] D. B. Fogel, L. J. Fogel, and J. W. Atmar. Meta-evolutionary programming. In *Conference record of the twenty-fifth asilomar conference on signals, systems & computers*, pages 540–541. IEEE computer Society, 1991.

[82] E. Frazelle. *Supply Chain Strategy: The Logistics of Supply Chain Management*. Logistics Management Library. McGraw-Hill, 2002. ISBN 9780071375993.

[83] R. M. Freund. Introduction to semidefinite programming (sdp). *Massachusetts Institute of Technology*, pages 8–11, 2004.

[84] N. Gademann and V. S. Velde. Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75, 2005. ISSN 0740-817X.

[85] N. Gademann, J. P. Van Den Berg, and H. H. Van Der Hoff. An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE transactions*, 33(5):385–398, 2001.

[86] C. J. Georgiou and P. S. Stefaneas. Strategies for accelerating the worldwide adoption of e-commerce. *Communications of the ACM*, 45(4):145–151, 2002.

[87] V. Giannikas, W. Lu, B. Robertson, and D. Mc. Farlane. An interventionist strategy for warehouse order picking: Evidence from two case studies. *International Journal of Production Economics*, 189:63–76, 2017.

[88] D. R. Gibson and G. P. Sharp. Order batching procedures. *European Journal of Operational Research*, 58(1):57–67, 1992.

[89] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and A. Duarte. New VNS variants for the online order batching problem. *Lecture Notes in Computer Science*, 11328:89–100, 2018.

[90] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and A. Duarte. Basic VNS for a variant of the online order batching problem. *Lecture Notes in Computer Science*, 12010:17–36, 2019.

[91] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and A. Duarte. Fixed versus variable time window warehousing strategies in real time. *Progress in Artificial Intelligence*, 9:315–324, 2020.

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

[93] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and A. Duarte. A heuristic approach for the online order batching problem with multiple pickers. *Computers & Industrial Engineering*, 160:107517, 2021.

[94] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and A. Duarte. Order batching problems. taxonomy and literature review. *Technical Report. Submitted to European Journal of Operational Research*, pages 1–40, 2022.

[95] S. Gil-Borrás, K. Sorensen, E. Jiménez, and E. G. Pardo. A comparative study of the influence of the time-window strategy in online order batching problems. *Technical Report. Submitted to Computers & Industrial Engineering*, pages 1–47, 2022.

[96] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. Online order batching problem. In *Metaheuristics: Proceeding of the MIC and MAEB 2017 Conferences*, page 366, Barcelona (España), Del 4 al 7 junio 2017. ISBN: 978-84-697-4275-1.

[97] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. Búsqueda de vecindad variable para el problema de la agrupación y recogida de pedidos online en almacenes logísticos. In *XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA2018)*, Granada (España), Del 23 al 26 Octubre 2018.

[98] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. New VNS variants for the online order batching problem. In *6th International Conference on Variable Neighborhood Search (ICVNS 2018)*, Sithonia (Grecia), From 3 to 7 October 2018.

[99] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. Basic VNS for a variant of the online order batching problem. In *7th International Conference on Variable Neighborhood Search (ICVNS 2019)*, Rabat (Morocco), From 3 to 7 October 2019.

[100] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. Recogida de pedidos por lotes en entornos dinámicos con mulltiples operarios. In *XXXVIII Congreso Nacional de Estadística e Investigación Operativa (SEIO2019)*, Alcoy (España), Del 3 al 6 septiembre 2019. ISBN: 978-84-09-13580-6.

[101] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. Time window for the online order batching problem with variable neighborhood search. In *8th International Conference on Variable Neighborhood Search (ICVNS2021)*, Abu Dhabi, (U.A.E), From 22 to 24 March 2021. ISBN: 978-3-030-69624-5.

[102] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, D. A., and E. Jiménez-Merino. Algoritmos de estimación del tiempo de ventana en la recogida de pedidos en almacenes logísticos. In *XIX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA20/21) & XIV Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirado (MAEB2021)*, Málaga (España), Del 22 al 24 septiembre 2021. ISBN: 978-84-09-30514-8.

[103] S. Gil-Borrás, E. G. Pardo, A. Alonso-Ayuso, and D. A. Matheurísticas aplicadas al problema de recogida de pedidos por lotes en contextos offline. In *XXXIX Congreso Nacional de Estadística de Investigación Operativa y de las XIII Jornadas de Estadística Pública (SEIO2022)*, Granada (España), Del 7 al 10 de Junio 2022. ISBN: 978-84-09-41628-8.

[104] F. Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[105] F. Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[106] F. Glover. Genetic algorithms and scatter search: Unsuspected potentials. *Statistics and Computing*, 4(2):131–140, 1994.

[107] F. Glover. A template for scatter search and path relinking. In *European conference on artificial evolution*, pages 1–51. Springer, 1997.

[108] F. Glover and M. Laguna. Tabu search. *Handbook of combinatorial optimization*, 1–3: 2093–2229, 1998.

[109] F. Glover, J. P. Kelly, and M. Laguna. Genetic algorithms and tabu search: Hybrids for optimization. *Computers & Operations Research*, 22(1):111–134, 1995.

[110] F. W. Glover and G. A. Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

[111] R. A. Gómez-Montoya, A. A. Correa-Espinal, and J. D. Hernández-Vahos. Picking routing problem with k homogenous material handling equipment for a refrigerated warehouse. *Revista Facultad de Ingeniería Universidad de Antioquia*, 0(80):9–20, 2016.

[112] N. I. M. Gould and P. L. Toint. A quadratic programming bibliography. *Numerical Analysis Group Internal Report*, 1:32, 2000.

[113] M. Groner, R. Groner, and W. F. Bischof. Approaches to heuristics: A historical review. *Methods of heuristics*, pages 1–18, 1983.

[114] J. Gu, M. Goetschalckx, and L. F. Mc. Ginnis. Research on warehouse operation: A comprehensive review. *European journal of operational research*, 177(1):1–21, 2007.

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

[116] S. Hahn and A. Scholz. Order picking in narrow-aisle warehouses: A fast approach to minimize waiting times. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, 2017.

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

[118] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In *Meta-heuristics*, pages 433–458. Springer, 1999.

[119] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

[120] P. Hansen and N. Mladenović. A tutorial on variable neighborhood search. *Les Cahiers du GERAD ISSN*, 711:2440, 2003.

[121] P. Hansen, N. Mladenović, and D. Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001.

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

[123] P. Hansen, N. Mladenović, R. Todosijević, and S. Hanafi. Variable neighborhood search: Basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454, 2017.

[124] P. Hansen, N. Mladenović, J. Brimberg, and J. A. Moreno-Pérez. Variable neighborhood search. In *Handbook of metaheuristics*, pages 57–97. Springer, 2019.

[125] S. Henn. Algorithms for on-line order batching in an order picking warehouse. *Computers and Operations Research*, 39(11):2549–2563, 2012. ISSN 03050548.

[126] S. Henn. Variable neighborhood search for the order batching and sequencing problem with multiple pickers. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, 2012.

[127] S. Henn. 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, 2015. ISSN 19366590.

[128] S. Henn and V. Schmid. Metaheuristics for order batching and sequencing in manual order picking systems. *Computers & Industrial Engineering*, 66(2):338–351, 2013.

[129] S. Henn and G. Wäscher. Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, 222(3):484–494, 2012.

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

[131] S. Henn, S. Koch, and G. Wäscher. Order batching in order picking warehouses: A survey of solution approaches. In *Warehousing in the global supply chain*, pages 105–137. Springer, 2012.

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

[133] Y. C. Ho, T. S. Su, and Z. B. Shi. Order-batching methods for an order-picking warehouse with two cross aisles. *Computers & Industrial Engineering*, 55(2):321–347, 2008.

[134] F. M. Hofmann. *Order picking optimisation on a unidirectional cyclical picking line*. PhD thesis, Stellenbosch: Stellenbosch University, 2020.

[135] F. M. Hofmann and S. E. Visagie. Picking location metrics for order batching on a unidirectional cyclical picking line. *ORiON*, 35(2):161–186, 2019.

[136] F. M. Hofmann and S. E. Visagie. The effect of order batching on a cyclical order picking system. In *International Conference on Computational Logistics*, pages 252–268. Springer, 2021.

[137] L. Hojaghania, J. Nematian, A. A. Shojaiea, and M. Javadi. Metaheuristics for a new minlp model with reduced response time for on-line order batching. *Scientia Iranica*, 28 (5):2789–2811, 2019.

[138] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, Cambridge, Mass, 1992.

[139] S. Hong and Y. Kim. A route-selecting order batching model with the S-shape routes in a parallel-aisle order picking system. *European Journal of Operational Research*, 257(1): 185–196, 2017. ISSN 03772217.

[140] S. Hong, A. L. Johnson, and B. A. Peters. Large-scale order batching in parallel-aisle picking systems. *IIE Transactions*, 44(2):88–106, 2012.

[141] S. Hong, A. L. Johnson, and B. A. Peters. Batch picking in narrow-aisle order picking systems with consideration for picker blocking. *European Journal of Operational Research*, 221(3):557–570, 2012.

[142] H. H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for sat. *Artificial Intelligence*, 112(1-2):213–232, 1999.

[143] H. H. Hoos and T. Stützle. *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[144] M. Horvat. *An approach to order picking optimization in warehouses*. PhD thesis, M. Horvat, 2012.

[145] L.-F. Hsieh and Y.-C. Huang. New batch construction heuristics to optimise the performance of order picking systems. *Intern. Journal of Production Economics*, 131(2):618–630, 2011. ISSN 0925-5273.

[146] C.-M. Hsu, K.-Y. Chen, and M.-C. Chen. Batching orders in warehouses by minimizing travel distance with genetic algorithms. *Computers in Industry*, 56(2):169–178, 2005. ISSN 01663615.

[147] M. Huang, Q. Guo, J. Liu, and X. Huang. Mixed model assembly line scheduling approach to order picking problem in online supermarkets. *Sustainability*, 10(11):3931, 2018.

[148] H. Hwang and D. G. Kim. Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. *International Journal of Production Research*, 43(17): 3657–3670, 2005.

[149] K. Il-Choe and G. P. Sharp. Small parts order picking: Design and operation. Technical report, School of Industrial and Systems Engineering. Georgia Institute of Technology, Atlanta, Georgia, EEUU, 1991.

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

[151] X. Jiang, Y. Zhou, Y. Zhang, L. Sun, and X. Hu. Order batching and sequencing problem under the pick-and-sort strategy in online supermarkets. *Procedia computer science*, 126: 1985–1993, 2018.

[152] X. Jiang, L. Sun, Y. Zhang, and X. Hu. Order batching and sequencing for minimising the total order completion time in pick-and-sort warehouses. *Expert Systems with Applications*, 187:115943, 2022.

[153] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

[154] V. Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Citeseer, 1992.

[155] J. Karasek. An overview of warehouse optimization. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, 2(3):111–117, 2013.

[156] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[157] K. E. Kinnear, W. B. Langdon, L. Spector, P. J. Angeline, and U.-M. O'Reilly. *Advances in genetic programming*, volume 3. MIT press, 1994.

[158] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[159] V. Klee and G. J. Minty. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

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

[161] E. B. Korobkov. Warehouse order-picking process review. *Science and Education of the Bauman (MSTU)*, 15(3), 2015.

[162] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[163] P. Kübler, C. H. Glock, and T. Bauernhansl. A new iterative method for solving the joint dynamic storage location assignment, order batching and picker routing problem in manual picker-to-parts warehouses. *Computers & Industrial Engineering*, 147:106645, 2020.

[164] H. Kuhn, D. Schubert, and A. Holzapfel. Integrated order batching and vehicle routing operations in grocery retail–a general adaptive large neighborhood search algorithm. *European Journal of Operational Research*, 294(3):1003–1021, 2021.

[165] O. Kulak, Y. F. Sahin, and M. E. Taner. Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, 24(1):52–80, 2012. ISSN 19366582.

[166] G. Lai, H. Liu, W. Xiao, and X. Zhao. "Fulfilled by Amazon": A strategic perspective of competition at the e-commerce platform. *Manufacturing & Service Operations Management*, 2022.

[167] P. Langley, J. H. Gennari, and W. Iba. Hill-climbing theories of learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 312–323. Elsevier, 1987.

[168] P. Larranaga. A review on estimation of distribution algorithms. *Estimation of distribution algorithms*, pages 57–100, 2002.

[169] P. Larrañaga and J. A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2 of *Genetic Algorithms and Evolutionary Computation*. Springer Science & Business Media, 2001.

[170] E. L. Lawler. *The Traveling salesman problem : A guided tour of combinatorial optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons, Chichester West Sussex New York, 1985. ISBN 9780471904137.

[171] T. Le Duc and R. de Koster. Travel distance estimation in single-block abc-storage strategy warehouses. *Lecture Notes in Economics and Mathematical Systems*, pages 185–202, 2004.

[172] T. Le-Duc and R. B. M. De Koster. Travel time estimation and order batching in a 2-block warehouse. *European Journal of Operational Research*, 176(1):374–388, 2007. ISSN 03772217.

[173] J. Legriel, C. Le Guernic, S. Cotton, and O. Maler. Approximating the pareto front of multi-criteria optimization problems. In J. Esparza and R. Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 69–83, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12002-2.

[174] N. Lenoble, Y. Frein, and R. Hammami. Optimization of order batching in a picking system with a vertical lift module. *Lecture Notes in Business Information Processing. International Conference on Information Systems, Logistics and Supply Chain*, 262:153–167, 2016.

[175] N. Lenoble, Y. Frein, and R. Hammami. Optimization of order batching in a picking system with carousels. *IFAC-PapersOnLine*, 50(1):1106–1113, 2017. 20th IFAC World Congress, 20th World Congress of the International Federation of Automatic Control.

[176] N. Lenoble, Y. Frein, and R. Hammami. Order batching in an automated warehouse with several vertical lift modules: Optimization and experiments with real data. *European Journal of Operational Research*, 267(3):958–976, 2018.

[177] K. H. Leung, C. K. M. Lee, and K. L. Choy. An integrated online pick-to-sort order batching approach for managing frequent arrivals of b2b e-commerce orders under both fixed and variable time-window batching. *Advanced Engineering Informatics*, 45:101–125, 2020.

[178] J. Li, R. Huang, and J. B. Dai. Joint optimisation of order batching and picker routing in the online retailer's warehouse in China. *International Journal of Production Research*, 55 (2):447–461, 2016. ISSN 0020-7543.

[179] C.-C. Lin, J.-R. Kang, C.-C. Hou, and C.-Y. Cheng. Joint order batching and picker Manhattan routing problem. *Computers and Industrial Engineering*, 95:164–174, 2016. ISSN 03608352.

[180] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. *International Series in Operations Research & Management Science. Handbook of metaheuristics*, 57:320–353, 2003.

[181] T. Ma and P. Zhao. A review of algorithms for order batching problem in distribution center. In *International Conference on Logistics Engineering, Management and Computer Science*, pages 172–175. Atlantis Press, 2014.

[182] K. V. Manjunath and B. Ravishankar. Development of algorithm and flowchart for the operation optimization in warehouse. *International Journal of Science & Engineering Development Research*, 1(7):296–313, 2016.

[183] M. Masae, C. H. Glock, and P. Vichitkunakorn. A method for efficiently routing order pickers in the leaf warehouse. *International Journal of Production Economics*, 234:108069, 2021.

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

[185] M. Matusiak, R. B. M. De Koster, and J. Saarinen. Utilizing individual picker skills to improve order batching in a warehouse. *European Journal of Operational Research*, 263 (3):888–899, 2017. ISSN 0377-2217.

[186] T. Melinda, Nazaruddin, and R. Ginting. Design of warehousing system in order picking process: Literature review. *IOP Conference Series: Materials Science and Engineering*, 801(1):012126, 2020.

[187] B. Menéndez, E. G. Pardo, A. Duarte, A. Alonso-Ayuso, and E. Molina. General variable neighborhood search applied to the picking process in a warehouse. *Electronic Notes in Discrete Mathematics*, 47:77–84, 2015. ISSN 15710653.

[188] B. Menéndez, M. Bustillo, E. G. Pardo, and A. Duarte. General variable neighborhood search for the order batching and sequencing problem. *European Journal of Operational Research*, 263(1):82–93, 2017.

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

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

[191] F. M. Miguel, M. Frutos, F. Tohmé, and D. Rossit. A memetic algorithm for the integral OBP/OPP problem in a logistics distribution center. *Uncertain Supply Chain Management*, 7(2):203–214, 2019.

[192] F. M. Miguel, M. Frutos, M. Méndez, and F. Tohmé. Order batching and order picking with 3d positioning of the articles: Solution through a hybrid evolutionary algorithm. *Mathematical Biosciences and Engineering*, 19(6):5546–5563, 2022.

[193] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, 1960.

[194] F. Misni and L. S. Lee. A review on strategic, tactical and operational decision planning in reverse logistics of green supply chain network design. *Journal of Computer and Communications*, 5(8):83–104, 2017.

[195] S. Mitra. A white paper on scenario generation for stochastic programming. *Optirisk systems: White paper series, ref. No. OPT004*, 2006.

[196] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

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

[198] N. Mladenović, M. Dražić, V. Kovačevic-Vujčić, and M. Čangalović. General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3):753–770, 2008.

[199] D. Molina, J. Poyatos, J. D. Ser, S. García, A. Hussain, and F. Herrera. Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. *Cognitive Computation*, 12(5):897–939, 2020.

[200] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. *International Series in Operations Research & Management Science. Handbook of metaheuristics*, 57: 105–144, 2003.

[201] D. F. Murad, W. Ratnasari, B. Y. Saputra, and B. D. Wijanarko. Warehouse management system for smart digital order picking systems. *International Journal of New Media Technology*, 6(2):74–80, 2019.

[202] I. Muter and T. Öncan. An exact solution approach for the order batching problem. *IIE Transactions*, 47(7):728–738, 2015.

[203] L. Nicolas, F. Yannick, and H. Ramzi. Order batching in an automated warehouse with several vertical lift modules: Optimization and experiments with real data. *European Journal of Operational Research*, 267(3):958–976, 2018.

[204] T. Öncan. 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, 2015. ISSN 03772217.

[205] T. Öncan and M. Cağırıcı. MILP formulations for the order batching problem in low-level picker-to-part warehouse systems. *IFAC Proceedings Volumes*, 46(9):471–476, 2013. ISSN 1474-6670.

[206] I. H. Osman and J. P. Kelly. Meta-heuristics: An overview. *Meta-heuristics*, pages 1–21, 1996.

[207] J. Oxenstierna, J. Malec, and V. Krueger. Layout-agnostic order-batching optimization. In *International Conference on Computational Logistics*, pages 115–129. Springer, 2021.

[208] S. G. Ozden, A. E. Smith, and K. R. Gue. A computational software system to design order picking warehouses. *Computers & Operations Research*, 132:105311, 2021.

[209] Ö. Öztürkoğlu and D. Hoser. A discrete cross aisle design model for order-picking warehouses. *European Journal of Operational Research*, 275(2):411–430, 2019.

[210] Ö. Öztürkoğlu and A. Mağara. A new layout problem for order-picking warehouses. In *Proceedings of the 9th international conference on industrial engineering and operations management. Bangkok, Thailand*, pages 1047–1058, 2019.

[211] M. W. Padberg. A note on zero-one programming. *Operations Research*, 23(4):833–837, 1975.

[212] J. C.-H. Pan and S. Y. Liu. A comparative study of order batching algorithms. *Omega*, 23(6):691–700, 1995. ISSN 0305-0483.

[213] J. C.-H. Pan and M.-H. Wu. Order batching in a picker-to-part warehousing system of a supply chain. In *Proceedings of ISER 135th Intenational Conference*, pages 48–51, Saint Petersburg, Russian Federation, 2018. The International Society for Engineers and Researchers.

[214] J. C.-H. Pan, P.-H. Shih, and M.-H. Wu. Order batching in a pick-and-pass warehousing system with group genetic algorithm. *Omega*, 57:238–248, 2015.

[215] L. Pansart, N. Catusse, and H. Cambazard. Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127, 2018.

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

[217] E. G. Pardo, S. Gil-Borrás, A. Alonso-Ayuso, and D. A. Multipicker order batching problem in dynamic environments. In *2019 INFORMS-ALIO International Conference (ALIO2019)*, Cancún (México), From 9 to 12 June 2019.

[218] R. Pérez-Rodríguez and A. Hernández-Aguirre. An estimation of distribution algorithm-based approach for the order batching problem. *Research in Computing Science*, 93: 141–150, 2015.

[219] R. Pérez-Rodríguez, A. Hernández-Aguirre, and S. Jöns. A continuous estimation of distribution algorithm for the online order-batching problem. *The International Journal of Advanced Manufacturing Technology*, 79(1):569–588, 2015. ISSN 1433-3015.

[220] C. G. Petersen. Routeing and storage policy interaction in order picking operations. *Decision Science Institute Proceedings*, 31(3):1614–1616, 1995.

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

[222] C. G. Petersen. An evaluation of order picking policies for mail order companies. *Production and operations management*, 9(4):319–335, 2000.

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

[224] U. Pferschy and J. Schauer. Order batching and routing in a non-standard warehouse. *Electronic Notes in Discrete Mathematics*, 69:125–132, 2018.

[225] A. R. F. Pinto and M. S. Nagano. An approach for the solution to order batching and sequencing in picking systems. *Production Engineering*, 13(3-4):325–341, 2019.

[226] A. R. F. Pinto and M. S. Nagano. A comprehensive review of batching problems in low-level picker-to-parts systems with order due dates: Main gaps, trade-offs, and prospects for future research. *Journal of Manufacturing Systems*, 65:1–18, 2022.

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

[228] S. D. Poisson. *Recherches sur la probabilité des jugements en matière criminelle et en matière civile: Précédées des règles générales du calcul des probabilités*. Bachelier, 1837.

[229] J. T. Postema. Metaheuristics for order batching in ecommerce warehouses. In *27th Twente Student Conference on IT*, pages 1–7, Enschede, The Netherlands, July 2017. University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

[230] K. V. Price. Differential evolution: A fast and simple numerical optimizer. In *Proceedings of North American fuzzy information processing*, pages 524–527. IEEE, 1996.

[231] N. J. Radcliffe and P. D. Surry. Formal memetic algorithms. In T. C. Fogarty, editor, *Evolutionary Computing*, pages 1–16, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[232] P. Raghuram and M. K. Arjunan. Design framework for a lean warehouse–a case study-based approach. *International Journal of Productivity and Performance Management*, 2021.

[233] S. A. B. Rasmi, Y. Wang, and H. Charkhgard. Wave order picking under the mixed-shelves storage strategy: A solution method and advantages. *Computers & Operations Research*, 137:105556, 2022.

[234] H. D. Ratliff and A. S. Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31(3):507 – 521, 1983.

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

[236] K. J. Roodbergen and R. B. M. De Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001. ISSN 0020-7543.

[237] K. J. Roodbergen and R. B. M. De Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001.

[238] S. Røpke. Adaptive large neighbourhood search. In *MITACS/CORS 2010 Annual Conference*, 2010.

[239] M. B. Rosenwein. An application of cluster analysis to the problem of locating items within a warehouse. *IIE transactions*, 26(1):101–103, 1994.

[240] M. B. Rosenwein. A comparison of heuristics for the problem of batching orders for warehouse selection. *International Journal of Production Research*, 34(3):657–664, 1996.

[241] R. A. Ruben and F. R. Jacobs. Batch construction heuristics and storage assignment strategies for walk/ride and pick systems. *Management Science*, 45(4):575–596, 1999.

[242] Y. Ruberg and A. Scholz. A mathematical programming formulation for the single-picker routing problem in a multi-block layout. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, 2016.

[243] J. I. U. Rubrico, T. Higashi, H. Tamura, and J. Ota. Online rescheduling of multiple picking agents for warehouse management. *Robotics and Computer-Integrated Manufacturing*, 27 (1):62 – 71, 2011. ISSN 0736-5845.

[244] J. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual (2nd Edition)*. Pearson Higher Education, 2004. ISBN 0321245628.

[245] A. Rushton, P. Croucher, and P. Baker. *The Handbook of logistics and distribution management*. Kogan Page Limited, 2006.

[246] A. Rushton, P. Croucher, and P. Baker. *The Handbook of logistics and distribution management: Understanding the supply chain*. Kogan Page Limited, 2022.

[247] M. Schiffer, N. Boysen, P. S. Klein, G. Laporte, and M. Pavone. Optimal picking policies in e-commerce warehouses. *Management Science*, 2022.

[248] M. Schleyer and K. R. Gue. Throughput time distribution analysis for a one-block warehouse. *Transportation Research Part E: Logistics and Transportation Review*, 48(3): 652–666, 2012.

[249] A. Scholz. An exact solution approach to the single-picker routing problem in warehouses with an arbitrary block layout. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, 2016.

[250] A. Scholz and G. Wäscher. 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, 2017. ISSN 1613-9178.

[251] A. Scholz, S. Henn, M. Stuhlmann, and G. Wäscher. A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, 253(1):68–84, 2016.

[252] A. Scholz, D. Schubert, and G. Wäscher. 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, 2017.

[253] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[254] A. H. Schrotenboer, S. Wruck, I. F. A. Vis, and K. J. Roodbergen. Integration of returns and decomposition of customer orders in e-commerce warehouses. *arXiv preprint arXiv:1909.01794*, 2019.

[255] D. Schubert, A. Scholz, and G. Wäscher. Integrated order picking and vehicle routing with due dates. *OR Spectrum*, 40(4):1109–1139, 2018.

[256] J. Sebo and J. Busa Jr. Comparsion of advanced methods for picking path optimization case study of dual-zone warehouse. *Methodology*, 3:14–17, 2020.

[257] B. Shah, V. Khanzode, et al. A comprehensive review of warehouse operational issues. *International Journal of Logistics Systems and Management*, 26(3):346–378, 2017.

[258] A. Shapiro. *Lectures on stochastic programming : Modeling and theory*. Society for Industrial and Applied Mathematics Mathematical Programming Society, Philadelphia, 2009. ISBN 9780898716870.

[259] F. H. N. Shavaki and F. Jolai. A rule-based heuristic algorithm for joint order batching and delivery planning of online retailers with multiple order pickers. *Applied Intelligence*, 51:3917—3935, 2021.

[260] G. Sierksma. *Linear and integer programming: Theory and practice*. CRC Press, 2001.

[261] K. Sörensen and F. Glover. Metaheuristics. *Encyclopedia of operations research and management science*, 62:960–970, 2013.

[262] J. Sprave. Linear neighborhood evolution strategy. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 42–51. World Scientific, 1994.

[263] H. Stegherr, M. Heider, and J. Hähner. Classifying metaheuristics: Towards a unified multi-level classification system. *Natural Computing*, pages 1–17, 2020.

[264] R. Storn. On the usage of differential evolution for function optimization. In *Proceedings of north american fuzzy information processing*, pages 519–523. Ieee, 1996.

[265] T. Stützle. Local search algorithms for combinatorial problems. *Darmstadt University of Technology PhD Thesis*, 20, 1998.

[266] L. C. Tang and E. P. Chew. Order picking systems: Batching and storage assignment strategies. *Computers & Industrial Engineering*, 33(3):817 – 820, 1997.

[267] C. Thomas and P. Schaus. Revisiting the self-adaptive large neighborhood search. In W.-J. van Hoeve, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 557–566, Cham, 2018. Springer International Publishing.

[268] X. Tian, L. Zhou, and J. Yang. Research on two-stage order picking sequencing for intensive shelf. In *MATEC Web of Conferences*, volume 296, page 02003. EDP Sciences, 2019.

[269] P. F. J. Tielemans and R. Kuik. An exploration of models that minimize leadtime through batching of arrived orders. *European Journal of Operational Research*, 95(2):374–389, 1996.

[270] J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. *Facilities planning*. John Wiley & Sons, 2010.

[271] C. Y. Tsai, J. H. Liou, and T. M. Huang. Using a multiple-GA method to solve the batch picking problem: Considering travel distance and order due time. *International Journal of Production Research*, 46(22):6533–6555, 2008.

[272] C. A. Valle and J. E. Beasley. Order batching for picker routing using a distance approximation. *arXiv preprint arXiv:1808.00499*, 2018.

[273] C. A. Valle and J. E. Beasley. Order batching using an approximation for the distance travelled by pickers. *European Journal of Operational Research*, 284(2):460–484, 2020.

[274] C. A. Valle, J. E. Beasley, and A. S. Da Cunha. Modelling and solving the joint order batching and picker routing problem in inventories. *Lecture Notes in Computer Science. Combinatorial Optimization: 4th International Symposium, ISCO*, 9849:81–97, 2016.

[275] C. A. Valle, J. E. Beasley, and A. S. Da Cunha. Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, 262(3):817–834, 2017.

[276] J. P. Van Der Gaast, B. Jargalsaikhan, and K. J. Roodbergen. Dynamic batching for order picking in warehouses. In *15th IMHRC Proceedings*, pages 1–7, Savannah, Georgia. USA, 2018.

[277] T. Van Gils, K. Braekers, K. Ramaekers, B. Depaire, and A. Caris. Improving order picking efficiency by analyzing the combination of storage, batching, zoning and routing policies. *Lecture Notes in Computer Science. Computational Logistics. ICCL 2016*, 9855: 427–442, 2016.

[278] T. Van Gils, K. Ramaekers, K. Braekers, B. Depaire, and A. Caris. Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. *International Journal of Production Economics*, 197:243–261, 2018.

[279] T. Van Gils, A. Caris, K. Ramaekers, and K. Braekers. Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, 277(3):814–830, 2019.

[280] P. J. Van Laarhoven and E. H. Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

[281] I. Van Nieuwenhuyse and R. B. M. De Koster. Evaluating order throughput time in 2-block warehouses with time window batching. *International Journal of Production Economics*, 121(2):654–664, 2009.

[282] I. Van Nieuwenhuyse, R. B. M. De Koster, and J. Colpaert. Order batching in multi-server pick-and-sort warehouses. *Katholieke Universiteit Leuven, Department of Decision Sciences and Information Management*, 180(140):367–8869, 2007.

[283] S. Vanheusden, T. Van Gils, A. Caris, K. Ramaekers, and K. Braekers. Operational workload balancing in manual order picking. *Computers & Industrial Engineering*, 141: 106269, 2020.

[284] T. Vossen, M. Verhoeven, H. ten Eikelder, and E. Aarts. A quantitative analysis of iterated local search. *Computing Science Reports*, 95(06), 1995.

[285] C. Voudouris. Guided local search—an illustrative example in function optimisation. *BT Technology Journal*, 16(3):46–50, 1998.

[286] C. Voudouris and E. Tsang. Guided local search joins the elite in discrete optimisation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 57:29–40, 2001.

[287] S. Wagner and L. Mönch. A variable neighborhood search approach to solve the order batching problem with heterogeneous pick devices. *European Journal of Operational Research*, 2022.

[288] G. Wäscher and A. Scholz. A solution approach for the joint order batching and picker routing problem in a two-block layout. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, 2015.

[289] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 2039–2046. IEEE, 1999.

[290] F. Weidinger, N. Boysen, and M. Schneider. Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, 274(2):501–515, 2019.

[291] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[292] D. Whitley and T. Starkweather. Genitor ii: A distributed genetic algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(3):189–214, 1990.

[293] I. M. Whittley and G. D. Smith. The attribute based hill climber. *Journal of Mathematical Modelling and Algorithms*, 3(2):167–178, 2004.

[294] D. Williamson. *The design of approximation algorithms*. Cambridge University Press, Cambridge New York, 2011. ISBN 9780521195270.

[295] G. J. Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial optimization—eureka, you shrink!*, pages 185–207. Springer, 2003.

[296] J. Won and S. Olafsson. Joint order batching and order picking in warehouse operations. *International Journal of Production Research*, 43(7):1427–1442, 2005.

[297] L. Xie, H. Li, and L. Luttmann. Formulating and solving integrated order batching and routing in multi-depot AGV-assisted mixed-shelves warehouses. *arXiv preprint arXiv:2101.11473*, 2021.

[298] X. Xu, T. Liu, K. Li, and W. Dong. Evaluating order throughput time with variable time window batching. *International Journal of Production Research*, 52(8):2232–2242, 2014.

[299] J. Yang, L. Zhou, and H. Liu. Hybrid genetic algorithm-based optimisation of the batch order picking in a dense mobile rack warehouse. *Plos one*, 16(4):e0249543, 2021.

[300] N. Yang. Evaluation of the joint impact of the storage assignment and order batching in mobile-pod warehouse systems. *Mathematical Problems in Engineering*, 2022, 2022.

[301] P. Yang, Z. Zhao, and H. Guo. Order batch picking optimization under different storage scenarios for e-commerce warehouses. *Transportation Research Part E: Logistics and Transportation Review*, 136:101897, 2020.

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

[303] J. Zhang, X. Wang, and K. Huang. Integrated on-line scheduling of order batching and delivery under b2c e-commerce. *Computers & Industrial Engineering*, 94:280 – 289, 2016. ISSN 0360-8352.

[304] J. Zhang, X. Wang, F. T. S. Chan, and J. Ruan. On-line order batching and sequencing problem with multiple pickers: A hybrid rule-based algorithm. *Applied Mathematical Modelling*, 45(Supplement C):271 – 284, 2017. ISSN 0307-904X.

[305] J. Zhang, X. Wang, and K. Huang. On-line scheduling of order picking and delivery with multiple zones and limited vehicle capacity. *Omega*, 79:104 – 115, 2018. ISSN 0305-0483.

[306] J. Zhang, X. Zhang, and Y. Zhang. A study on online scheduling problem of integrated order picking and delivery with multizone vehicle routing method for online-to-offline supermarket. *Mathematical Problems in Engineering*, 2021, 2021.

[307] X. Zhao, N. Liu, S. Zhao, J. Wu, K. Zhang, and R. Zhang. Research on the work-rest scheduling in the manual order picking systems to consider human factors. *Journal of Systems Science and Systems Engineering*, pages 1–12, 2019.

[308] I. Žulj, S. Kramer, and M. Schneider. A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, 264(2):653–664, 2018. ISSN 0377-2217.

[309] I. Žulj, H. Salewski, D. Goeke, and M. Schneider. Order batching and batch sequencing in an AMR-assisted picker-to-parts system. *European Journal of Operational Research*, 2021.

[310] E. Zunic, A. Besirevic, R. Skrobo, H. Hasic, K. Hodzic, and A. Djedovic. Design of optimization system for warehouse order picking in real environment. In *2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–6. IEEE, 2017.

[311] E. Zunic, K. Hodzic, H. Hasic, R. Skrobo, A. Besirevic, and D. Donko. Application of advanced analysis and predictive algorithm for warehouse picking zone capacity and content prediction. In *2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–6, 2017.

[312] C. A. Zuniga, E. Olivares-Benitez, A. M. Tenahua, and M. A. Mujica. A methodology to solve the order batching problem. *IFAC-PapersOnLine*, 48(3):1380–1386, 2015.