Integrating order picking and vehicle routing decisions in a dynamic e-commerce setting

Peer-reviewed author version

D'HAEN, Ruben; RAMAEKERS, Katrien; Archetti, Claudia & BRAEKERS, Kris (2024) Integrating order picking and vehicle routing decisions in a dynamic e-commerce setting. In: Computers & operations research, 170 (Art N° 106762).

DOI: 10.1016/j.cor.2024.106762
Handle: http://hdl.handle.net/1942/43456

Integrating order picking and vehicle routing decisions in a dynamic e-commerce setting

Ruben D'Haen[1,2], Katrien Ramaekers[1], Claudia Archetti[3], and Kris Braekers[1]

[1]Research Group Logistics, UHasselt, Agoralaan Building D, 3590 Diepenbeek, Belgium
[2]Research Foundation Flanders (FWO), Leuvenseweg 38, 1000 Brussel, Belgium
[3]Department of Information Systems, Decision Sciences and Statistics, ESSEC Business School, 3 Avenue Bernard Hirsch, 95000 Cergy, France

June 26, 2024

E-mail: ruben.dhaen@uhasselt.be (corresponding author), katrien.ramaekers@uhasselt.be, claudia.archetti@essec.edu, kris.braekers@uhasselt.be

Abstract

Responding to e-commerce orders as quickly as possible is indispensable for companies competing in the current retailing landscape. After a customer orders online, the requested items should be picked in the warehouse and delivered to the customer's location. Previous research showed the benefits of integrating the order picking and vehicle routing decisions in a static context. To achieve shorter customer response times, we propose multiple new algorithms in this study, able to handle the integrated problem of picking and delivery in a setting with dynamic order arrivals and considering more complex picking policies. We develop and implement four online large neighbourhood search algorithms, differing in their degree of integration between picking and routing decisions. The algorithms are tested on different operational settings. Differences between the algorithms and the impact of the operating context are analysed by use of an ANOVA. The results highlight the importance of integrated decision-making, as well as

the large impact of the operating context on the operational efficiency. Furthermore, we demonstrate the environmental impact of stringent customer requests. These insights can be used by companies to optimise their operations.

Keywords

Order picking; Vehicle routing; Integrated decision making; Order batching; Metaheuristics

1 Introduction

E-commerce sales have rapidly grown during the past decade. The share of companies offering an e-commerce sales channel has grown accordingly, increasing the competition to attract and retain customers. Customers may be persuaded by offering a lower price for home deliveries or by shorter delivery delays [Alnaggar et al., 2021]. Same-day deliveries are no exception anymore, with some companies even going as far as promising delivery within one or two hours [Ulmer, 2017]. To be able to attain these customer service levels, and to do so at reasonable costs, efficient order handling processes are indispensable.

Two important steps in order handling are the order picking and order delivery processes. In the order picking process, ordered items are collected from their storage locations. Afterwards, the order delivery process transports the picked orders to the customer's location. Traditionally, these processes are optimised individually. Extensive research is available on the optimisation of order picking (see reviews of de Koster et al. [2007] and van Gils et al. [2018]) and vehicle routing individually (e.g., see Toth and Vigo [2014] and Braekers et al. [2016] for a general overview of the vehicle routing problem, and Archetti and Bertazzi [2021] for a discussion on recent developments and challenges related to same-day deliveries). In the traditional optimisation approach, order picking and delivery are considered as two separate processes, where every order is assigned a cutoff time between picking and delivery (i.e., a picking deadline) based on a simple rule of thumb. This traditional optimisation approach ignores the interaction between both problems, leading to suboptimal solutions for the combined order handling process. For example, orders are only considered to be available for delivery after their picking deadline, although some orders may have been picked long before this time. By considering order picking and delivery as an integrated optimisation problem, an intelligent choice for this cutoff time can be made. If an order is only scheduled for routing very late, the picking operations may be postponed. Similarly, if an order is already picked very early during the planning period, it may be added to a delivery route starting long before the cutoff time under the traditional approach.

In this paper we study how order picking and vehicle routing decisions can be optimised jointly. This leads to a complex optimisation problem in which order picking and vehicle routing decisions are interdependent. Within a vehicle routing context, the synchronisation of delivery or pick-up operations has been studied before (see the literature reviews of Drexl [2012] and Soares et al. [2023]). Moreover, synchronisation has been considered within two-echelon vehicle routing problems (see the literature review of Sluijk et al. [2023]). However, the synchronisation of order picking and vehicle routing decisions has some additional particularities, e.g., the same order picking completion time for all orders within a single pick tour, and multiple locations that need to be visited in the warehouse for a single order. Previous research already studied the integrated order picking and vehicle routing problem (IOPVRP) and highlighted the possible efficiency improvements of integrated optimisation [Moons et al., 2018, Schubert et al., 2018]. The order picking problem involved was kept quite simple in this research, as orders are picked individually, i.e., no order batching is allowed, despite the large benefits associated with order batching [Chen and Wu, 2005, de Koster et al., 2007]. Order batching was considered later by Kuhn et al. [2021], showing a highly improved performance on the combined order picking and vehicle routing problem. Still, in each of these contributions, the routing of the order pickers in the warehouse is determined by an S-shape heuristic, avoiding the real integration of order batching, picker routing and batch scheduling [D'Haen et al., 2023]. With an S-shape routing policy, order pickers have to traverse the entire aisle if there is at least one item to pick in that aisle. While this policy is relatively efficient if there are on average many picks per aisle, an optimised routing policy allows for increased flexibility, e.g., by turning around within an aisle if this reduces the picking distance. In this study, we go beyond the state-of-the-art by considering the integrated order batching, picker routing (with an optimised routing policy) and batch scheduling problem in the picking part of the IOPVRP, i.e., a complex picker-to-parts setting is considered.

Additionally, up to now, research on the IOPVRP considered a static problem setting in which all orders are known at the start of the planning period. In the e-commerce setting that many companies face nowadays, new orders arrive while the operations are ongoing. Therefore, we consider a setting in which orders arrive dynamically during the picking and routing operations. The contributions of this study are therefore multiple:

- We extend the IOPVRP to consider the integrated order batching, picker routing and batch scheduling problem in the order picking part of the problem.
- A procedure to handle dynamic order arrivals is proposed, making the setting more applicable in practice, i.e., the online integrated order picking and vehicle routing problem (o-IOPVRP) is considered.
- We introduce several new metaheuristic algorithms to solve the o-IOPVRP. These algorithms differ in the level of integration between both subproblems and in the

way this integration is achieved (e.g., iterating between optimising the subproblems, or defining neighbourhoods over the joint problem). The algorithms' performance is analysed to understand the impact of their components and how to best integrate the decisions. Additionally, differences in the solution structure of the algorithms are analysed to obtain insights into the characteristics of good solutions.

- Insights in the benefits of integration are deduced from the algorithms' performance on a series of test instances, leading to valuable knowledge for academia and practitioners. We show that integrated decision-making can lead to substantial improvements in solution quality compared to sequential decision-making (average reductions in tardiness and distance traveled of 74% and 17%, respectively). Furthermore, our results indicate that the best performance is achieved when alternating between global and local neighbourhoods during the optimisation, providing valuable insights for tackling similar integrated optimisation problems.

- We test the algorithms on a diverse set of instances, constructed to mimic different operating contexts. The results lead to an understanding of the importance of the operating context and its impact on the the operating efficiency as well as the environmental impact of the operations. Customers being a bit more flexible regarding their delivery preferences, i.e., accepting a slightly longer delivery delay and a wider delivery time window, has a large impact on the travel distance. Reductions in distance of up to 18% can be achieved, highlighting large efficiency improvements, leading to both savings for the company and a reduction in environmental impact of the delivery operations. This finding is valuable for practitioners and may be used to convince customers in being less demanding to reduce their environmental impact.

The remainder of this paper is structured as follows. First, Section 2 gives an overview of the literature related to this study. Next, Section 3 gives a detailed description of the problem studied in this paper. Section 4 proposes the different metaheuristic algorithms, followed by the computational experiments in Section 5. Finally, conclusions are presented in Section 6.

2 Literature review

This section discusses the most relevant literature for the IOPVRP. First, literature on the subproblems is discussed. Section 2.1 refers to the literature on integrated solution approaches for the order picking problem, more specifically, where order batching, picker routing and batch scheduling decisions are made in an integrated manner. Next, an overview of vehicle routing problems with release dates, the VRP most closely related

to the one encountered in the IOPVRP, is given in Section 2.2. Finally, literature on the IOPVRP is discussed in Section 2.3.

2.1 Order picking problem

This section gives a brief overview of the most relevant literature regarding the integrated (online) optimisation of the order picking problem. For a more elaborate discussion regarding online order batching problems and integrated order picking, we refer to D'Haen et al. [2023] and van Gils et al. [2018], respectively. The order picking problem considered within the IOPVRP is the integrated order batching, picker routing and batch scheduling problem. In this problem, the decisions regarding which orders to combine in a single pick tour (order batching), in which sequence the item locations of a pick tour are visited (picker routing) and the assignment and sequence of the tasks to pickers (batch scheduling) are made in an integrated manner. The integrated problem including all three subproblems was first studied by Chen et al. [2015] for a setting with a single order picker. Later, the setting was extended to include multiple order pickers by Scholz et al. [2017] and van Gils et al. [2019]. In an e-commerce setting, the dynamic arrival of orders should be accounted for. Literature on integrated decision-making in the online context is rare. Rubrico et al. [2011] consider the integration of order batching and picker routing with dynamic order arrivals. Batch scheduling is performed by simple scheduling heuristics, preventing real integration. Real integration in the online context was introduced by D'Haen et al. [2023], who studied the integration of order batching, picker routing and batch scheduling for the first time in the online context.

2.2 Vehicle routing problem with release dates

After picking an order, the order becomes available for delivery. Since the picking completion time is known once an order is scheduled for picking, the time of availability of this order for delivery is known in advance. The VRP considered is therefore a VRP with release dates (VRP-rd). Furthermore, since new orders arrive during the planning period, an online VRP with release dates is considered in our setting.

The VRP-rd has only recently received attention in the literature. An overview of the literature is given in Table 1. In the first article on the VRP-rd, Archetti et al. [2015] study a setting with a single vehicle, as well as a setting with an infinite number of vehicles. The objective of the study is proving the complexity of the VRP with release dates, rather than solving the problem. In specific settings, i.e., a line or star graph, the problem can be solved in polynomial time. This research was extended by Reyes et al. [2018] to consider order due times. A first solution approach for the single vehicle VRP-rd on a general graph is proposed by Archetti et al. [2018] and is based on an iterated local search algorithm.

Table 1: Literature on vehicle routing problems with release dates.

	Vehicles		**Trips/vehicle**		**Due times**	**TW**	**Dynamic**
	Single	Multiple	Single	Multiple			
Archetti et al. [2015]	x	x		x	x		
Cattaruzza et al. [2016]		x		x		x	
Shelbourne et al. [2017]		x	x		x		
Archetti et al. [2018]	x			x			
Reyes et al. [2018]	x	x		x	x		
Archetti et al. [2020]	x			x			x
Li et al. [2020]		x		x			
Soman and Patil [2020]		x	x		x		
Li et al. [2021]		x		x		x	
Yang et al. [2021]		x	x		x		

Most of the literature on the VRP-rd considers multiple vehicles. Some authors consider a setting in which every vehicle can make a single trip only [Shelbourne et al., 2017, Soman and Patil, 2020, Yang et al., 2021]. In these studies, orders have a due time before which they should be delivered. In other articles, vehicles are allowed to perform multiple trips. This setting is studied by Li et al. [2020] without order due times. Cattaruzza et al. [2016] consider a setting with multiple vehicles, each allowed to perform multiple trips, where orders should be delivered within a time window. This setting is most closely related to the VRP encountered in our work. Yet, all orders are static in their work, contrary to the problem studied in this paper (and order picking is not considered).

While extensive research exists on the dynamic VRP [Ojeda Rios et al., 2021], many of the associated decisions, e.g., waiting strategies or the use of stochastic information, are not as useful in our problem context. Since orders should be picked before delivery can begin, vehicles can already wait for the picking completion times (i.e., the release dates in the VRP) of known orders. Adding a waiting strategy for the VRP for unknown orders would lead to even longer waiting times, and may not be feasible in highly dynamic settings as considered in this study. For literature reviews on dynamic vehicle routing, we refer to Pillac et al. [2013], Psaraftis et al. [2016] and Ojeda Rios et al. [2021]. To the best of our knowledge, only a single paper considers the VRP-rd in a dynamic setting. In the work of Archetti et al. [2020] a single vehicle performs the delivery of dynamically arriving orders. Contrary to our work, stochastic information about future orders is available, and this information is shown to lead to better performance than the deterministic scenario.

Some optimisation problems in the literature are closely related to a VRP-rd. For example, the dynamic dispatch waves problem, studied by Klapp et al. [2018] has some similar characteristics, with orders with an associated release date that should be delivered to the customers. However, not all orders should be delivered, leading to order acceptance decisions that should be made, taking stochastic information on future requests into account. A single vehicle is performing the delivery operations. Additionally, the routing problem's complexity is reduced, since all customers are located on a single line, leading to the trip's duration being set by this longest distance. In the same-day delivery literature, related problems can be found, where orders have release dates, based on stochastic information. Voccia et al. [2017] consider a same-day delivery problem in which the number of requests that can be fulfilled should be maximised, considering stochastic information on future requests. Furthermore, the vehicle departures may be postponed based on the expected future orders. Finally, van Heeswijk et al. [2019] consider the dispatching decision at the depot, where they decide on when a vehicle should be dispatched, based on stochastic information on future requests.

2.3 Integrated order picking and vehicle routing

An overview of the literature on the integrated order picking and vehicle routing problem is given in Table 2. The integration of order picking and delivery decisions was first studied by Zhang et al. [2016, 2018, 2019]. In these studies, order picking operations are optimised by use of a batching algorithm and S-shape picker routing. The delivery decisions are very basic, however, since a real VRP is not included in the optimisations. Rather, orders have a fixed departure time, set by a third-party logistics provider [Zhang et al., 2016], should only be routed to a certain zone, without considering the individual customer locations [Zhang et al., 2018], or have a fixed cost in a certain vehicle without making an actual routing plan [Zhang et al., 2019]. Contrary to later research, the orders arrive dynamically in Zhang et al. [2016, 2018, 2019], which is characteristic for an e-commerce setting. The objective in Zhang et al. [2016] is to maximise the number of delivered orders, meaning that not all orders have to be handled, whereas Zhang et al. [2018, 2019] minimise the sum of the makespan and delivery cost.

Later research included a real VRP in the integrated problem. Moons et al. [2018] solve the IOPVRP with an exact algorithm in CPLEX, with cost minimisation as the objective function. Order picking operations are performed order-by-order, where order pickers follow an S-shape routing policy. The VRP considers heterogeneous capacitated vehicles, each allowed to perform a single trip, with time windows at the customer locations. The results highlight the benefits of integration: a cost reduction of approximately 12% is possible compared to an unintegrated approach. This work is extended with a heuristic algorithm in Moons et al. [2019], tested on instances of up to 100 orders. Cost reductions average 1.8% when integrating order picking and vehicle routing, combined with the possibility to offer a higher service level to the customers. Closely related work

Table 2: Literature on integrated order picking and vehicle routing problems.

	Batching	S-shape	VRP	Time windows	Online
Zhang et al. [2016]	x	x			x
Moons et al. [2018]		x	x	x	
Schubert et al. [2018]		x	x		
Zhang et al. [2018]	x	x			x
Moons et al. [2019]		x	x	x	
Zhang et al. [2019]	x	x			x
Kuhn et al. [2021]	x	x	x	x	
Schubert et al. [2021]		x	x	x	
Ostermeier et al. [2022]		x	x	x	

was conducted by Schubert et al. [2018], with the largest difference being the objective function: tardiness instead of costs are considered. The iterated local search algorithm is tested on instances of up to 200 orders, and shows tardiness improvements of 17.2% on average for the integrated solution approach compared to a sequential approach.

Additional restrictions were later added to the IOPVRP. Schubert et al. [2021] look at the delivery of large durable consumer goods, which are picked individually in the warehouse. Delivery scheduling should take the vehicle class into account, since different services require different vehicles. The results show benefits of the integrated solution approach in line with earlier research, with the cost improvements depending on the problem characteristics. A constraint regarding a limited intermediate storage area between picking and shipping was introduced by Ostermeier et al. [2022]. Again, a sequential approach is compared to an integrated approach, showing a much larger number of feasible solutions found by the integrated solution algorithm, highlighting the importance of integrated decision-making.

Kuhn et al. [2021] make an important contribution by introducing order batching into the picking part of the IOPVRP. The use of order batching allows for a more realistic problem setting, but at the cost of increased problem complexity. The proposed solution method is a general adaptive large neighbourhood search (GALNS) algorithm, a combination of an ALNS with a general variable neighbourhood search. The algorithm is applied on problem settings with up to 200 orders, with the objective to minimise tardiness. The results show some interesting insights. Compared to a sequential approach with discrete picking, tardiness can be reduced by 56% and 52% by introducing an order batching strategy and moving to an integrated solution approach, respectively. The combination of both integrated solving and batch picking, can reduce tardiness by 76% in the experimental setup, and 68% in a case study.

Finally, the integration of order picking and vehicle routing is considered by Rijal et al. [2023], but in a slightly different manner compared to the other articles. Although

order batching is considered, the warehouse operations are not modelled in detail. Instead, based on real-life data, the required time to pick a batch is estimated by use of an ordinary least squares regression model, where the batch picking time is based on the number of items in that batch. The use of an integrated order picking and vehicle routing algorithm was shown to lead to cost savings of approximately 10% compared to a sequential optimisation approach.

To conclude, previous research clearly highlights the benefits of integrated decision-making. Order batching is also proven to be very beneficial, even though the routing policy studied in the literature was always S-shape routing, while more efficient strategies exist. In this paper, we study a more complex setting by incorporating a state-of-the-art order picking strategy, in which picker routing is optimised as well, i.e., the integrated order batching, picker routing and batch scheduling problem is considered. Furthermore, a multi-trip VRP is examined in this study. Finally, a large contribution to the literature is the handling of dynamically arriving orders, which was not yet studied for the integrated order picking and vehicle routing problem.

3 Online integrated order picking and vehicle routing problem

In this section, we define the online integrated order picking and vehicle routing problem (o-IOPVRP) that is studied in this paper. The problem description is given in Section 3.1, followed by a mathematical model in Section 3.2.

3.1 Problem description

In the o-IOPVRP, customers, located at geographically dispersed locations, order some goods at a retailer operating from a single distribution center. First, the requested items have to be picked in the warehouse. Next, the picked items should be delivered to the customer's locations. The problem consists of finding the most efficient way to deliver all orders, while adhering to the following constraints.

When looking at the order picking part, all orders should be picked by human order pickers travelling through the warehouse (picker-to-parts). The warehouse can have any layout. Orders consist of one or multiple order lines and can be grouped in batches. Order splitting is not allowed, i.e., all order lines of an order should be allocated to the same batch. The maximum capacity of the picking device, i.e., the batch capacity, is expressed as a number of orders. Parallel to allocating orders to batches (order batching problem), the pickers' routes through the warehouse to visit all item locations in a batch have to be determined (picker routing problem) and the batches should be assigned and scheduled for a picker (batch scheduling problem). A mathematical problem formulation

for the order picking part can be found in D'Haen et al. [2023], based on work of van Gils et al. [2019].

The second part of the problem considers the vehicle routing decisions. The problem at hand has order release dates in the VRP, since orders can only be loaded in a vehicle after their picking completion time. The order picking schedule, and the picking completion time of every order in particular, is thus used as an input in the VRP. The vehicle fleet consists of a limited number of homogeneous capacitated vehicles, each allowed to perform multiple trips. Every trip starts and ends at the warehouse, where the departure time of a vehicle trip is at the earliest the maximum of the release times of the orders included in that trip. Every order belongs to a customer and should be delivered within the associated time window. If a vehicle arrives early, it has to wait for the start of the time window. Delivering late is possible, but at the cost of incurring tardiness. Without loss of generality, we assume that picked items can be loaded as soon as the picker has completed the associated pick tour, and that the vehicle loading process does not consume any time. All presented methods can easily deal with situations in which a fixed delay for packing or loading is incurred.

The objective of the o-IOPVRP is to deliver all orders with as little tardiness as possible. The objective function is hierarchical, however, with at the second level the total distance in the VRP. If two solutions have equal tardiness, the solution with a lower distance is preferred. Note that there is no objective regarding the order picking part of the problem. Yet, the solution for the order picking problem directly impacts the delivery schedule, since order release times for the VRP are derived from the order picking schedule. The underlying assumption is that order pickers (and vehicle drivers) represent sunk costs, i.e., they are paid for their complete shift, regardless of their actual working time.

Since an e-commerce setting is considered where customers expect very fast delivery, an online problem context is studied. Orders arrive dynamically during the planning period. It is assumed that no information on future order arrivals is available. If this information were available, taking it into consideration could improve the performance of the algorithm, as was shown for the order picking problem in D'Haen et al. [2023]. Once an order arrives it can be included in the schedule. However, note that rerouting is not allowed for both the picking and delivery problems, i.e., the batches that are being picked and the delivery trips that are being performed at the time of new order arrivals cannot be modified. A simple graphical representation of the problem is given in Figure 1, with a timeline of the process for a dynamically arriving order given in Figure 2.

3.2 Mixed integer linear programming model

A mixed integer linear programming model for the IOPVRP, i.e., the static counterpart of the o-IOPVRP, is developed here. The model considers the dynamic arrival of orders, by including order arrival times. This model assumes all order arrival times are known

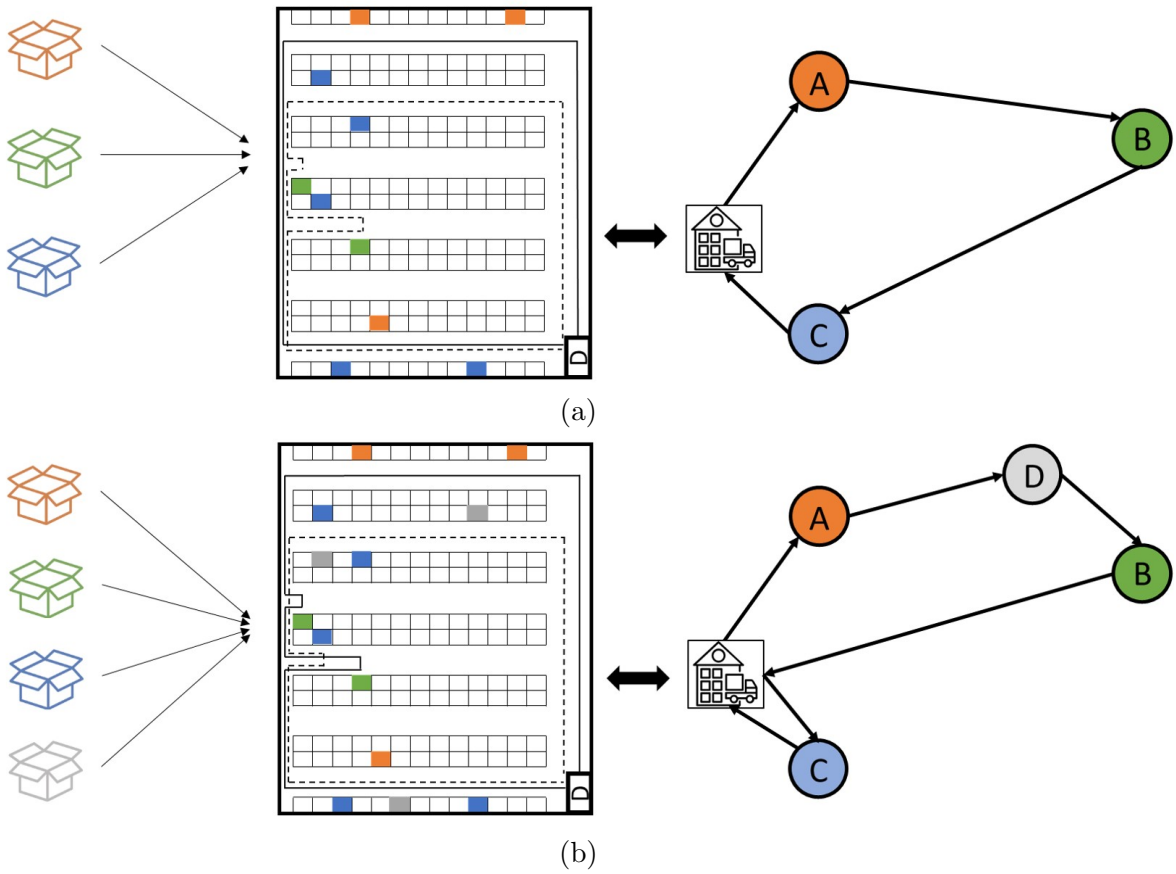


Figure 1: Graphical representation of a toy example for the online integrated order picking and vehicle routing problem. (a) Initially, three orders are known and scheduled for both picking and delivery. The orders are picked in two batches, graphically represented in the warehouse with a solid and dashed line. (b) The gray order is a dynamically arriving order. Assuming the previous orders have not yet been handled, the schedule is updated to include the new order. The batches in the order picking problem are modified, as well as the delivery routes to the customers' locations.

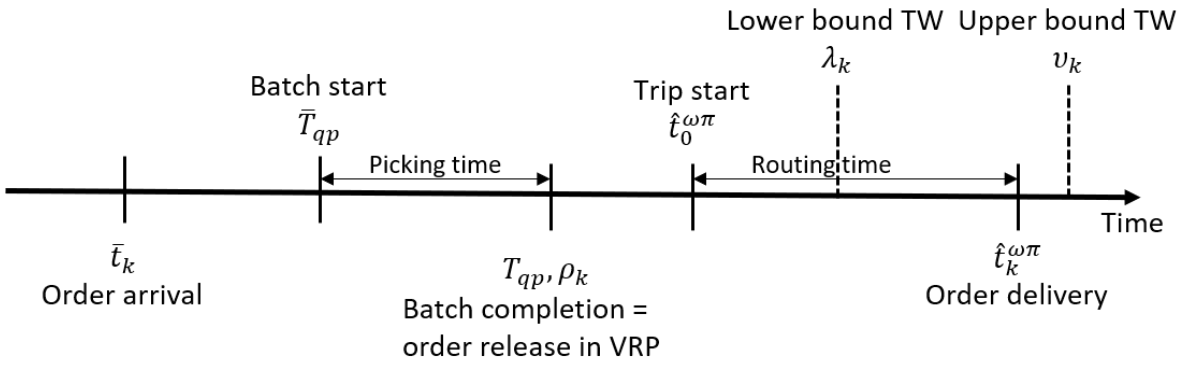


Figure 2: A timeline of the events from order arrival to order delivery. Depending on the operations, the time between order arrival and batch start, and between batch completion and trip start can be zero or positive, but never negative. The symbols used in the figure are introduced in Section 3.2.

at the start of the planning period. In practice, and in our experiments, this information is not known beforehand. Results of this mathematical model are thus lower bounds on the objective function value of the online problem that we consider, and are obtained by considering more information than is available in reality. Nevertheless, the formulation is presented to formally introduce the optimisation problem considered during the full planning period.

The optimisation problem clearly consists of an order picking and a vehicle routing part that are combined to obtain an integrated problem setting. Sets, parameters and decision variables for the optimisation problem are given in Table 3, with the subproblem for which an entry is relevant highlighted in the last column. Note that the set of customers is equivalent to the set of customer orders, since, without loss of generality, we assume every customer is associated with a single order.

Table 3: Notation MIP.

Sets	Explanation	OPP	VRP
$\kappa = \{1, 2, ..., \lvert\kappa\rvert\}$	Set of customers with index k	•	•
$\Omega = \{1, 2, ..., \lvert\Omega\rvert\}$	Set of vehicles with index ω		•
$\Pi = \{1, 2, ..., \lvert\Pi\rvert\}$	Set of trips of a vehicle with index π		•
$\Psi = \{0, 1, ..., \lvert\Psi\rvert + 1\}$	Set of vertices with index ψ (0 and $\lvert\Psi\rvert + 1$ represent the depot, 1 to $\lvert\Psi\rvert$ are the locations of all customers in κ)		•
$\mathrm{A} = \{1, 2, ..., \lvert\mathrm{A}\rvert\}$	Set of arcs with index α		•
$Q = \{1, 2, ..., \lvert Q\rvert\}$	Set of order pickers with index q	•	
$P = \{1, 2, ..., \lvert P\rvert\}$	Set of batches of a picker with index p	•	
$V = \{0, 1, ..., \lvert V\rvert\}$	Set of vertices with index v (picker depot is 0)	•	
$W = \{V^1, V^2, ..., V^S\}$	Set with all possible subsets of vertices $V^s \subset V \setminus 0 : \lvert V^s\rvert > 1$	•	

$E = \{1, 2, ..., \lvert E \rvert\}$	Set of arcs with index e connecting a start and end vertex $\in V$	•	
$E_{V^s} \subset E$	Subset of arcs with e connecting a start and end vertex $\in V^s$	•	
$E_v^- \subset E$	Subset of arcs ending in a vertex $v \in V$	•	
$E_v^+ \subset E$	Subset of arcs starting in a vertex $v \in V$	•	
$V_k \subset V$	Subset of vertices that should be visited to collect the items for customer $k \in \kappa$	•	
Parameters	Explanation	OPP	VRP
$\mathcal{M}$	A large number (Big M notation)	•	•
λ_k	Lower bound of the time window for customer $k \in \kappa$		•
υ_k	Upper bound of the time window for customer $k \in \kappa$		•
C	Vehicle capacity		•
t_{ij}	Travel time when travelling from location i $\in \Psi$ to location j $\in \Psi$		•
c_k	Size of order $k \in \kappa$		•
o_k	Number of order lines of order $k \in \kappa$	•	
c	Batch capacity (in number of orders)	•	
t_e	Travel time when travelling across arc $e \in E$ (in seconds)	•	
t_{setup}	Batch setup time (in seconds)	•	
t_{search}	Search and pick time to visit a storage location (in seconds)	•	
$\bar{t}_k$	Arrival time to the system of customer order $k \in \kappa$ with respect to the start of the planning horizon ($t = 0$)	•	
Decision variables	Explanation	OPP	VRP
ρ_k	Release time of order $k \in \kappa$ for delivery (i.e., the picking completion time of this order)	•	•
$\chi_{ij}^{\omega\pi}$	Binary decision variable which is equal to 1 if and only if the arc between i and j is visited by vehicle $\omega \in \Omega$ in trip $\pi \in \Pi$		•
$\Upsilon_k^{\omega\pi}$	Binary decision variable which is equal to 1 if and only if vehicle $\omega \in \Omega$ visits customer $k \in \kappa$ in trip $\pi \in \Pi$		•
$\hat{t}_\psi^{\omega\pi}$	Arrival time of vehicle $\omega \in \Omega$ in trip $\pi \in \Pi$ at vertex $\psi \in \Psi$		•
τ_k	Tardiness of order $k \in \kappa$		•
X_{qpe}	Binary decision variable which is equal to 1 if and only if arc $e \in E$ is traversed by order picker $q \in Q$ in batch $p \in P$	•	
Y_{qpv}	The outdegree of vertex $v \in V$ (i.e., number of arcs leaving v) by order picker $q \in Q$ in batch $p \in P$	•	
Z_{qpv}	Binary decision variable which is equal to 1 if and only if vertex $v \in V$ is visited by order picker $q \in Q$ in batch $p \in P$	•	
R_{qpk}	Binary decision variable which is equal to 1 if and only if order $k \in \kappa$ is completed by order picker $q \in Q$ in batch $p \in P$	•	
T_{qp}	Completion time of the batch completed by order picker $q \in Q$ in batch $p \in P$	•	
$\bar{T}_{qp}$	Start time of the batch completed by order picker $q \in Q$ in batch $p \in P$	•	

The formulation of the problem is given below. First, the objective functions are introduced. Since the objective function of the integrated optimisation problem is minimising tardiness in the vehicle routing problem, the constraints of the VRP are introduced first, and are based on Archetti et al. [2018] but adapted to consider multiple vehicles, multiple trips and time windows. Next, the order picking problem is discussed, followed by a linking constraint between the subproblems.

Objective function:

$$\text{Primary objective:} \qquad \min \sum_{k \in \kappa} \tau_k \tag{1}$$

$$\text{Secondary objective:} \qquad \min \sum_{\omega \in \Omega} \sum_{\pi \in \Pi} \sum_{i \in \Psi} \sum_{j \in \Psi} t_{ij} \chi_{ij}^{\omega\pi} \tag{2}$$

$$\sum_{\omega \in \Omega} \sum_{\pi \in \Pi} \Upsilon_k^{\omega\pi} = 1 \quad \forall k \in \kappa \tag{3}$$

$$\sum_{k \in \kappa} c_k \Upsilon_k^{\omega\pi} \leq C \quad \forall \omega \in \Omega, \pi \in \Pi \tag{4}$$

$$\hat{t}_j^{\omega\pi} \geq \hat{t}_i^{\omega\pi} + t_{ij} \chi_{ij}^{\omega\pi} - \mathcal{M}(1 - \chi_{ij}^{\omega\pi}) \quad \forall (i,j) \in \mathrm{A}, \omega \in \Omega, \pi \in \Pi \tag{5}$$

$$\hat{t}_k^{\omega\pi} \geq \lambda_k \quad \forall k \in \kappa, \omega \in \Omega, \pi \in \Pi \tag{6}$$

$$\hat{t}_k^{\omega\pi} \leq \upsilon_k + \tau_k \quad \forall k \in \kappa, \omega \in \Omega, \pi \in \Pi \tag{7}$$

$$\sum_{j \in \Psi \setminus \{0\}} \chi_{kj}^{\omega\pi} = \sum_{j \in \Psi \setminus \{\Psi+1\}} \chi_{jk}^{\omega\pi} = \Upsilon_k^{\omega\pi} \quad \forall k \in \kappa, \omega \in \Omega, \pi \in \Pi \tag{8}$$

$$\sum_{k \in \kappa} \chi_{0k}^{\omega\pi} \leq 1 \quad \forall \omega \in \Omega, \pi \in \Pi \tag{9}$$

$$\hat{t}_0^{\omega\pi} \geq \hat{t}_{|\Psi|+1}^{\omega(\pi-1)} \quad \forall \omega \in \Omega, \pi \in \Pi \setminus \{1\} \tag{10}$$

$$\hat{t}_0^{\omega\pi} \geq \rho_k - \mathcal{M}(1 - \Upsilon_k^{\omega\pi}) \quad \forall k \in \kappa, \omega \in \Omega, \pi \in \Pi \tag{11}$$

$$\sum_{k \in \kappa} \chi_{0k}^{\omega\pi} \leq \sum_{k \in \kappa} \chi_{0k}^{\omega(\pi-1)} \quad \forall \omega \in \Omega, \pi \in \Pi \setminus \{1\} \tag{12}$$

$$\hat{t}_\psi^{\omega\pi} \geq 0 \quad \forall \psi \in \Psi, \omega \in \Omega, \pi \in \Pi \tag{13}$$

$$\chi_{ij}^{\omega\pi} \in \{0,1\} \quad \forall (i,j) \in \mathrm{A}, \omega \in \Omega, \pi \in \Pi \tag{14}$$

$$\Upsilon_k^{\omega\pi} \in \{0,1\} \quad \forall k \in \kappa, \omega \in \Omega, \pi \in \Pi \tag{15}$$

$$\tau_k \geq 0 \quad \forall k \in \kappa \tag{16}$$

$$\rho_k \geq \bar{t}_k \quad \forall k \in \kappa \tag{17}$$

The primary objective function (1) minimises the tardiness to deliver all orders. If two solutions have equal tardiness, the solution with the lowest distance, the secondary

objective function (2), is preferred. Constraints (3) ensure that every order is assigned to a single vehicle trip. Constraints (4) make sure not to exceed the vehicle capacity in any trip. The arrival time of a vehicle at a location should be larger than the arrival time in the previous vertex and should consider the travel time between the vertices, as enforced by constraints (5). Delivery at a customer's location can never occur before the start of the time window, while late delivery is allowed but leads to tardiness, ensured by constraints (6) and (7), respectively. Constraints (8) make sure that a vehicle travels to and from a customer's vertex in a given trip if this customer is assigned to that vehicle's trip. A vehicle can leave the depot at most once per trip, enforced by constraints (9). A trip cannot start before the previous trip of that vehicle arrived at the depot and not before all orders included in the trip have been released for delivery (i.e., picking has been completed), imposed by constraints (10) and (11), respectively. Symmetry breaking constraints are included by constraints (12). Although they are redundant, they drastically reduce the search space. Finally, the domain constraints are given by constraints (13) - (17).

Before the orders can be loaded into a vehicle, they have to be picked in the warehouse. The formulation of the order picking subproblem is strongly based on the formulation of van Gils et al. [2019], which in turn was based on the formulation of Valle et al. [2017], with some small modifications to allow for order arrival times (i.e., the time when a new, dynamic customer order arrives in the system), introduced to account for the dynamic arrival of orders. The relevant constraints are as follows.

$$\sum_{e \in E_v^-} X_{qpe} = \sum_{e \in E_v^+} X_{qpe} \quad \forall q \in Q, p \in P, v \in V \tag{18}$$

$$\sum_{e \in E_v^-} X_{qpe} = Y_{qpv} \quad \forall q \in Q, p \in P, v \in V \tag{19}$$

$$\sum_{v' \in V^s} Y_{qpv'} \geq Z_{qpv} + \sum_{e \in E_{V^s}} X_{qpe} \quad \forall q \in Q, p \in P, v \in V^s, V^s \in W \tag{20}$$

$$X_{qpe} \leq Z_{qpv} \quad \forall q \in Q, p \in P, v \in V, e \in E_v^+ \tag{21}$$

$$\sum_{e \in E_0^-} X_{qpe} = \sum_{e \in E_0^+} X_{qpe} = Z_{qp0} \quad \forall q \in Q, p \in P, e \in E \tag{22}$$

$$X_{qpe} \leq Z_{qp0} \quad \forall q \in Q, p \in P, e \in E \tag{23}$$

$$R_{qpk} \leq Z_{qp0} \quad \forall q \in Q, p \in P, k \in \kappa \tag{24}$$

$$\sum_{e \in E_v^+} X_{qpe} \geq R_{qpk} \quad \forall q \in Q, p \in P, k \in \kappa, v \in V_k \tag{25}$$

$$\sum_{k \in \kappa} R_{qpk} \geq Z_{qp0} \quad \forall q \in Q, p \in P \tag{26}$$

$$\sum_{k \in \kappa} R_{qpk} \leq c \quad \forall q \in Q, p \in P \tag{27}$$

$$\sum_{q \in Q} \sum_{p \in P} R_{qpk} = 1 \quad \forall k \in \kappa \tag{28}$$

$$T_{qp} = \bar{T}_{qp} + t_{setup} Z_{qp0} + t_{search} \sum_{k \in \kappa} o_k R_{qpk} + \sum_{e \in E} t_e X_{qpe} \quad \forall q \in Q, p \in P \tag{29}$$

$$T_{qp} \leq \mathcal{M} \sum_{k \in \kappa} R_{qpk} \quad \forall q \in Q, p \in P \tag{30}$$

$$\bar{T}_{qp} \geq T_{q(p-1)} \quad \forall q \in Q, p \in P \setminus \{1\} \tag{31}$$

$$\bar{T}_{qp} \geq \bar{t}_k - \mathcal{M}(1 - R_{qpk}) \quad \forall q \in Q, p \in P, k \in \kappa \tag{32}$$

$$X_{qpe}, R_{qpk} \in \{0, 1\} \quad \forall q \in Q, p \in P, k \in \kappa, e \in E \tag{33}$$

$$Z_{qpv} \in [0; 1] \quad \forall q \in Q, p \in P, v \in V \tag{34}$$

$$T_{qp}, \bar{T}_{qp} \geq 0 \quad \forall q \in Q, p \in P \tag{35}$$

Constraints (18) make sure that the number of in- and outbound arcs is equal for every vertex in every batch. The number of arcs leaving a vertex is computed for every batch by constraints (19), and this number is used in constraints (20) to avoid sub-tour creation (as discussed in van Gils et al. [2019]). Constraints (21) make sure to only visit vertices in a batch when an arc starts in the vertex. Constraints (22) ensure an incoming and outgoing arc from the depot if the depot is visited. Constraints (23) and (24) force at least one used arc or one added order to a batch if the depot is included in this batch. Constraints (25) make sure to visit the vertices of orders assigned to the batch. If the depot is visited, there should be at least one order assigned to the batch, which is enforced by constraints (26). Constraints (27) ensure not to exceed the batch capacity, and constraints (28) make sure an order is only added to a single batch. The completion time of a batch is set by constraints (29), and constraints (30) make sure a completion time is computed for all batches. The start time of a batch should be later than the completion time of the previous batch of that picker, which is enforced by constraints (31). As orders can have an arrival time, a batch cannot start before all orders in the batch have arrived, which is imposed by constraints (32). The domain constraints are given in constraints (33) - (35).

Finally, the order picking and vehicle routing problem should be linked to each other. An order can only be loaded into a vehicle once it has been deposited at the depot by the order picker picking the order's batch. Therefore, the order's release time in the VRP is based on its batch completion time from the order picking problem, enforced by constraints (36).

$$\rho_k \geq T_{qp} - \mathcal{M}(1 - R_{qpk}) \quad \forall q \in Q \quad \forall p \in P \quad \forall k \in \kappa \tag{36}$$

The formulation outlined in this section has been implemented in CPLEX 20.1. However, even very small instances with only eight orders take a very long time to solve or cannot be solved to optimality within four hours. As such, the mathematical model is used to clarify the problem description, but will not be used in the remainder of this paper. Instead, because of the computational complexity of the o-IOPVRP, a metaheuristic algorithm is developed to solve realistically sized instances.

4 Algorithm description

For the o-IOPVRP, multiple optimisation approaches are possible. We develop and test multiple algorithms, i.e., a sequential algorithm, an iterative algorithm and more involved, integrated algorithms will be compared to quantify the benefits of integrating picking and delivery operations. The results are used to highlight the benefits of integration and discuss in which operating context integration is most beneficial. This section gives an overview of the different algorithms proposed in this study. All solution algorithms are metaheuristic algorithms, since solving the o-IOPVRP with exact methods is not feasible within limited computation time. The algorithmic framework for every algorithm is large neighbourhood search (LNS), in which a solution is repeatedly partially destroyed and repaired, to move towards better solutions [Shaw, 1998, Pisinger and Ropke, 2007].

Since this study considers both picking and routing decisions, optimisation algorithms for the picking and routing subproblem individually are first proposed. Next, these algorithms can be used as components in the algorithms where order picking and vehicle routing decisions are integrated. To optimise the o-IOPVRP, four optimisation algorithms are proposed, each dealing with the problem in its own way. The objective is to find the best way to integrate both problems, in order to deal with the interaction between order picking and vehicle routing. The baseline is a sequential approach, in which the subproblems are optimised by using the picking and routing algorithm components sequentially. A second algorithm is an iterative algorithm, in which repeated optimisation of the individual subproblems is considered, i.e., a local optimisation approach, staying within a subproblem, is used. A third algorithm is an integrated algorithm, in which global optimisation is considered, changing both picking and routing decisions at the same time. Finally, a fourth algorithm, an integrated iterative algorithm, combines global and local optimisation.

The remainder of this section is structured as follows. First, the optimisation approach to deal with dynamic order arrivals is discussed in Section 4.1. Next, the algorithm components, i.e., the order picking and vehicle routing algorithms for the individual subproblems, are proposed in Section 4.2 and Section 4.3, respectively. The sequential algorithm is outlined in Section 4.4, followed by the iterative algorithm in Section 4.5. Next, the integrated solution algorithm is discussed in Section 4.6. Finally,

Section 4.7 introduces the integrated iterative algorithm.

4.1 Online optimisation strategy

Online optimisation has not been studied yet for the IOPVRP. Therefore, the online optimisation strategy proposed in this paper is based on the optimisation strategy of the online order picking problem as studied in D'Haen et al. [2023]. In this strategy, a sequence of reoptimisation steps is performed during the planning period. A reoptimisation step is performed every time an order picker returns to the depot and consists of solving a static problem, using all information available at the time of reoptimisation. At every reoptimisation step, a subset of all orders is known, forming the active order pool. An active order pool is available for both picking and routing separately, in which all orders that are known but not yet handled are inserted. Note that the active order pools of picking and routing can be different if orders have been, or are being, picked, but the orders have not yet left the warehouse in a vehicle. More specifically, once the picking operations have started, the active order pool of the picking problem is a subset of the one of the routing problem.

At the start of the planning period, a schedule for both the order picking and vehicle routing operations is constructed, containing all orders of the respective active order pool, with the vehicle routing operations taking the order completion times of the order picking problem into account. Order pickers are then leaving the depot to collect the orders, while vehicles can only leave the depot once all orders of their first trip have been picked.

When an order picker finishes his current batch, he returns to the depot and requests a new pick list. Once the first picker returns to the depot, the first reoptimisation step is performed, with a new reoptimisation step being performed every time a picker returns to the depot until the end of the planning period. Since rerouting is not allowed (for both pickers and vehicles), a picker returning to the depot is the first time that an order that arrived after the last reoptimisation step can be considered for picking and can be added to a picker's schedule. Furthermore, once it is added to the picking schedule, a picking completion time is available that allows for the creation of a routing schedule in which this order is considered.

In a reoptimisation step, the active order pools are first updated. Orders that have been picked or are currently being picked are removed from the active order pool for the picking problem, orders that left the depot in a delivery vehicle are removed from the active order pool for the routing problem and any new orders since the previous reoptimisation step are added to both active order pools. Next, a complete schedule is again constructed for both picking and routing, starting from the last available solution from which any batch or trip that has been started is removed. In a reoptimisation step, additional information should be taken into account compared to the first optimisation step, i.e., the completion times of pickers currently picking a batch in the warehouse

are considered and can be seen as a release time for this picker, with this time being the first time a new batch can be assigned to this picker. Similarly, in the routing problem, vehicles have completion times of their current trips, acting as release dates for the vehicles. Therefore, in the routing problem, both vehicles and orders have release times, with the former being set by the delivery trips being performed by the vehicles at the time of the reoptimisation, and the latter set by the order completion times in the picking problem. This situation, of having release dates for both multiple vehicles and orders is unique and not yet encountered in the literature on vehicle routing problems with release dates.

The reoptimisation procedure is graphically shown in Figure 3. This procedure is performed every time an order picker returns to the depot, as long as there are orders remaining to be picked. Once all orders have been picked, the optimisations stop, even though some orders may still need to be delivered. However, since the VRP optimisation algorithm takes the picking completion times into account, and since these completion times will not be modified if no new orders arrive, the final VRP schedule is still based on the most recent information.

The following sections discuss the developed optimisation algorithms used during the initial optimisation and the reoptimisation steps.

4.2 Order picking subproblem

In the order picking subproblem considered in this study, a schedule to pick all orders as efficiently as possible should be constructed, by using an integrated order batching, picker routing and batch scheduling algorithm. Every order has a release date and a due date in the order picking problem. The release date is the time the order arrives in the system, i.e., when the customer order is registered. The picking due date is caused by the vehicle routing operations following after picking, i.e., an order has to be loaded into the delivery vehicle before its trip starts and should thus be picked before this time. In the order picking subproblem, a surrogate objective is used to avoid repeated re-computation of the impact of changes in the picking schedule on the VRP solution. The objective function is hierarchical: the primary objective is minimising tardiness with respect to the picking due dates, the secondary objective is minimising the order pick time. The algorithm used in this study is based on the large neighbourhood search algorithm of D'Haen et al. [2023]. This algorithm is the state-of-the-art algorithm for the online integrated batching, routing and scheduling problem and was shown to have excellent performance regarding both solution quality and computation time. For a detailed description of the algorithm, we refer to D'Haen et al. [2023]. As the problem studied in the current paper adds complexity to the one studied in D'Haen et al. [2023], the algorithm has been adapted as follows.

1. The local search procedure is deactivated. While the local search procedure was

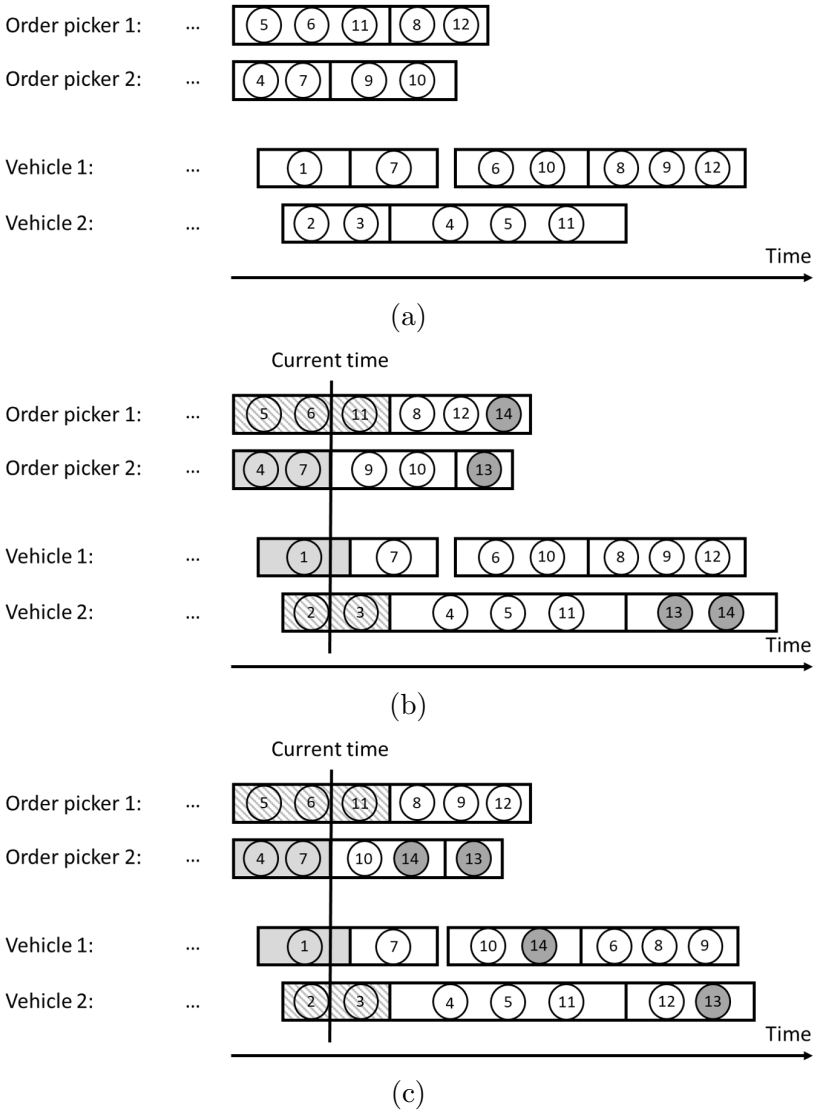


Figure 3: Graphical representation of the online optimisation strategy. Batches and trips are represented by a rectangle, orders by circles with a number indicating the order number. The current time is the time when reoptimisation happens, when an order picker returns to the depot. (a) Final schedule from the previous optimisation step. (b) Order picker 2 returns to the depot and requests a new pick list. Batches and trips that have been handled (shown in gray) are removed from the previous schedule. Batches and trips that are currently being picked and delivered (shown in shaded gray) are not allowed to be changed. Any new orders (shown in dark gray) are added to both the picking and routing schedule by two-regret insertion. (c) The optimisation algorithm reoptimises the schedule. Picker 2 is sent out to pick orders 10 and 14.

beneficial to reduce tardiness in the original problem, it is very time consuming. With the additional complexity of the integration of delivery operations into the order picking problem and since fast computation times are required in the online problem considered in this study, using the local search procedure is deemed infeasible. Although the deactivation of the local search will negatively influence the tardiness, it helps allowing for reduced computation time and a simpler algorithm. Furthermore, because of the high computational burden and the limited computation time assigned to each algorithm in an optimisation step (instead of a fixed number of iterations as in the original algorithm), the local search prohibits performing a meaningful number of LNS iterations, making a comparison between the different algorithms in this paper impossible.

2. The use of order anticipation is deactivated. In the study of D'Haen et al. [2023], orders belonged to certain shipping trucks, depending on their destination, and the expected number of orders was used to reserve space in the picking schedules. In the IOPVRP, however, orders do not belong to a single shipping truck. Instead, the routing decisions, including which vehicle will deliver the order and at what time the vehicle leaves, are part of the decision process. Therefore, it is much harder to predict where a future order's delivery location will be, when a vehicle will depart on a route visiting that destination, and thus before which time the order should be picked. Therefore, no stochastic information is taken into account while scheduling the operations.

4.3 Vehicle routing subproblem

The vehicle routing subproblem considered in this study is a multi-trip vehicle routing problem with release dates and time windows. Orders have unique destinations with an associated time window within which the delivery should occur. Early delivery is never allowed, late delivery is minimised, since the minimisation of tardiness is the primary objective. A secondary objective is the minimisation of the travel distance, but this objective is only relevant if two solutions have equal tardiness, i.e., the objective function is hierarchical.

In order to facilitate the integration with the LNS algorithm of the order picking subproblem, an LNS for the VRP seems a straightforward choice. However, since, to the best of our knowledge, no LNS algorithms exist for the online VRP with release dates, a new algorithm was developed. An initial solution for the first optimisation step is constructed by adding all orders to an individual trip, with the orders being assigned on the basis of their urgency (i.e., orders with the earliest time window start first). This solution is given as input to the LNS algorithm. In subsequent reoptimisation steps, the LNS algorithm starts from the last available solution from the previous optimisation step by removing handled orders and inserting any new orders with regret-2 insertion.

The new orders can be inserted in new trips before or after trips that already existed, or can be added to trips that already existed in the previous optimisation step. However, the orders cannot be added to trips that are currently being executed by a vehicle, as preemptive depot returns are not allowed.

The online VRP-rd algorithm is shown in Algorithm 1. In every iteration of the LNS, a random destroy and repair operator are selected. The implemented destroy operators are the following:

- *Order_Worst*: remove the orders that lead to the largest decrease in routing distance when removed from the solution.
- *Order_Earliness*: remove the orders with the most earliness, i.e., the largest positive difference of latest delivery time (i.e., the upper bound of the time window) minus the planned delivery time.
- *Order_Tardiness*: remove the orders with the most tardiness, i.e., the largest positive difference of planned delivery time minus latest delivery time.
- *Order_CenterOfGravity*: for all orders in all trips, calculate the distance to the center of gravity (i.e., the mean X- and Y-coordinates of the orders in that trip) of the trip the order is part of. Remove the orders with the largest distance to their trip's center point. This computation of the center of gravity of a route is similar to the computation in Reimann et al. [2004], but part of a completely different algorithm (LNS instead of an ant system) and ignores the number of requested items by a customer.
- *Order_RelatedDistance*: randomly select an initial order to be removed from the solution. Remove the orders that are closest located to the initial order. This operator is based on Pisinger and Ropke [2007].
- *Order_RelatedTime*: randomly select an initial order to be removed from the solution. Remove the orders with a latest delivery time closest to the initial order. This operator is based on Pisinger and Ropke [2007].
- *Order_Random*: randomly select orders to remove from the solution.
- *Trip_Tardiness*: for every trip, compute the tardiness of all orders included (an order that is delivered in time, has zero tardiness). Destroy the trips for which the sum of tardiness is highest.
- *Trip_Earliness*: for every trip, compute the earliness of all orders included (an order that is delivered late, has zero earliness). Destroy the trips for which the sum of earliness is highest.

- *Trip_Random*: randomly select trips to destroy.

First, the number of orders that should be removed is determined based on the destroy percentage and the size of the problem. Half of these orders are removed with the selected destroy operator (multiple orders or trips may be removed by a single operator to achieve the required destroy percentage), while the other half is removed by random order removals. The random order removals avoid performing the same deterministic iterations repeatedly and getting stuck in a local optimum. After achieving the required destroy percentage, the orders are reinserted in the partial solution based on their insertion cost. The insertion cost is expressed as the increase in tardiness, or, if no tardiness is involved, as the increase in routing distance. The following repair operators are implemented, based on the work of Pisinger and Ropke [2007]:

- Greedy insertion: randomly select the uninserted orders one by one and compute the insertion cost for every possible insertion position of every trip (including a new trip before or after any existing trip). Reinsert the selected order in the best position.
- Best insertion: for all removed orders, compute the insertion cost in every possible position of every trip. Insert the order with the lowest cost in its best position, and update the insertion costs for all remaining orders. Repeat updating the insertion costs and inserting the orders one by one until all orders are reinserted. Note that this operator differs from greedy insertion, since it computes the insertion costs for all orders that should be inserted and then inserts the order with the lowest cost in its best position, whereas greedy insertion first selects an order, computes the insertion costs for this specific order and inserts the order in its best position. This results in a higher computational complexity of best insertion compared to greedy insertion.
- Regret-2 insertion: for all removed orders, compute the insertion cost in every possible position of every trip. Insert the order with the largest cost difference between its best and second best position first and update the insertion costs for all other orders. Repeat until all orders are reinserted.

After applying a destroy and repair operator, simulated annealing is used to decide on the acceptance of the new solution. A solution with less tardiness (denoted by f^t in the algorithm) than the current solution is always accepted, while a tardiness increase is never allowed. If the tardiness of the new solution is equal to the current solution's tardiness, the new solution is always accepted if it has a lower distance (denoted by f^d), while it may be accepted if it has a higher distance, with the probability of acceptance decreasing for larger distance increases and lower temperatures. By slowly decreasing

Algorithm 1: LNS for the VRP subproblem.

Input: *MaxRunTimeVRP,* S^0
Initialise Best solution S^*, Current solution S^0, New solution S^1;
S^*, $S^1 \leftarrow S^0$;
while *RunTime* $<$ *MaxRunTimeVRP* **do**
    Select destroy operator randomly;
    Destroy S^1 with selected destroy operator;
    Select repair operator randomly;
    Repair S^1 with selected repair operator;
    if $(f^t(S^1) < f^t(S^*))$ **or** $(f^t(S^1) = f^t(S^*)$ **and** $f^d(S^1) < f^d(S^*))$ **then**
        $S^* = S^1$;
        $S^0 = S^1$;
    else if $(f^t(S^1) < f^t(S^0))$ **or** $(f^t(S^1) = f^t(S^0)$ **and** $f^d(S^1) < f^d(S^0))$ **then**
        $S^0 = S^1$;
    else if $(f^t(S^1) = f^t(S^0))$ **and** $(f^d(S^1) > f^d(S^0))$ **then**
        if $RandNum < e^{((f^d(S^0) - f^d(S^1))/Temperature)}$ **then**
            $S^0 = S^1$;
    Update Temperature;

the temperature during the runtime of the LNS, the optimisation is directed from diversification to intensification. The temperature is updated in every iteration, following Equation (37), where T is the temperature for the current LNS iteration, and T_0 is the initial temperature, based on a percentage of the distance of the initial solution. The temperature changes based on the percentage of the maximum runtime that has been used during the optimisation step.

$$T = T_0 \times TemperatureUpdatePercentage^{(100 \times ElapsedRunTime/MaxRunTime)} \tag{37}$$

4.4 Sequential algorithm

To optimise both picking and routing operations, a first step is the sequential solving of both subproblems. A sequential algorithm can be constructed using the algorithm components of Section 4.2 and Section 4.3. The order picking operations are optimised first, leading to a picking completion time for every order. These completion times are used as release dates for the orders during the optimisation of the routing operations. In Algorithm 2 the sequential solution procedure is detailed.

Note that this sequential algorithm is not really integrating the picking and routing decisions, but is performing better than using a fixed ratio between picking and routing for every order. Here, the ratio is set based on the difficulty of the picking and routing problem for a specific order. Under the sequential approach considered here, every order has its own cutoff time, based on the order's ratio between picking and routing. This cutoff time is still required when optimising the order picking operations, and functions as an artificial due time for an order. Although the primary (tardiness) and secondary (distance) objectives are both based on the VRP, a due time for the orders in the order picking problem is used to decide on the quality of an order picking solution, i.e., to compute the tardiness of that solution. By using this intermediate due time, evaluating the quality of a solution can be done immediately without needing to solve the complete VRP. This due time can be seen as a temporary cutoff time between picking and delivery.

The cutoff time of every order is based on the minimum picking and routing time for this order, i.e., it is the average of on the one hand the current time of the system plus the minimum time required to pick the order individually (i.e., the fastest way to pick the order) and on the other hand the latest delivery time of the order minus the minimum time required to deliver the order individually (i.e., the fastest way to deliver the order). This cutoff time is used as the order due time in the order picking problem. Once the picking operations have been scheduled, the picking completion time of an order is used as this order's release time for the VRP.

This sequential algorithm, considering an order-specific ratio between picking and routing, is used as a baseline to show the benefits of using algorithms that really integrate picking and routing.

Algorithm 2: Sequential solution algorithm.

1 ***Input:*** $TotalRunTime$
2 Initialise and construct initial picking solution S_P, routing solution S_R;
3 $MaxRunTimePicking = \frac{TotalRunTime}{2}$;`// Divide the available time evenly over both subproblems`
4 $MaxRunTimeRouting = \frac{TotalRunTime}{2}$;
5 $LNS_{Picking}(S_P, MaxRunTimePicking)$;
`// See details in Section 4.2`
6 Update order release times VRP based on picking completion times;
7 $LNS_{Routing}(S_R, MaxRunTimeRouting)$;
`// See details in Section 4.3`

4.5 Iterative algorithm

A first step towards integrating the order picking and vehicle routing decisions, is made by an iterative solution algorithm. In the iterative solution algorithm, the order picking and vehicle routing algorithms are used multiple times each, one after the other. Similar to the sequential algorithm, picking operations are optimised first, with the resulting picking completion times being used as order release dates for the VRP. Afterwards, the VRP is optimised. However, rather than stopping the optimisation after the VRP optimisation, the output of the VRP is now used as input for the picking optimisation algorithm, i.e., the vehicle departure time for an order's trip is used as the picking due time, since, to not delay the obtained schedule, every order should be picked before it can be loaded into a vehicle. The iterative algorithm alternates between picking and routing optimisation for a predefined number of iterations. The available computation time is divided evenly over both subproblems and over all iterations. The solution procedure is shown in Algorithm 3.

In optimising the order picking operations, an artificial due time for every order is required to decide on the quality of a solution for the order picking problem, as was detailed in Section 4.4. After solving the order picking problem, the picking completion times of all orders are available. These completion times are then used as an input for the VRP, as the picking completion times are the order release dates in the VRP. Based on these release dates and given the order's delivery time window at the customer's location, the VRP is solved. Now, the second iteration of the iterative procedure can begin. The solution of the VRP makes new information available for the order picking problem. Delivery trips can only start once all orders of that trip are picked. Therefore, for every order in a delivery trip, the vehicle departure time for that trip can be seen as the due time for the order picking problem, to keep the current vehicle routing schedule feasible. Based on these new order due times, the order picking problem can be solved, starting from the final solution of the previous iteration. The resulting new picking

Algorithm 3: Iterative solution algorithm.

Input: *TotalRunTime, MaxIterations*
Initialise and construct initial picking solution S_P, routing solution S_R;
$MaxRunTimePicking = \frac{TotalRunTime}{2*MaxIterations}$;// Divide the available time evenly over both subproblems and all iterations
$MaxRunTimeRouting = \frac{TotalRunTime}{2*MaxIterations}$;
$NoIterations = 0$;
while *NoIterations* $<$ *MaxIterations* **do**
    $LNS_{Picking}(S_P, MaxRunTimePicking)$;
    Update order release times VRP based on picking completion times;
    $LNS_{Routing}(S_R, MaxRunTimeRouting)$;
    Update order due times in picking problem based on vehicle departure times;
    NoIterations = NoIterations + 1;

completion time for every order can then be used as the new order release time for the VRP. This cycle continues until the maximum number of iterations, i.e., alternations between both subproblems, is reached.

Note that if only a single iteration is performed, i.e., solving the order picking problem, using the picking completion times as input for the VRP followed by solving the VRP, the iterative algorithm is reduced to the sequential algorithm of Section 4.4.

4.6 Integrated algorithm

As a third algorithm, we propose an integrated algorithm, considering the solutions of the order picking and vehicle routing problem together. The integrated algorithm is also based on an LNS framework. When destroying a solution, the algorithm removes orders from both problems at the same time. During the reinsertion, orders are reinserted in both problems. Afterwards, when all orders are reinserted, the decision on the acceptance of a solution is also made on the overall solution, not separately on the subproblems. An overview of the integrated approach is shown in Algorithm 4.

We introduce three sets of destroy operators, D^p, D^r and D^j, focusing on the picking, routing and joint problem, respectively. The use of different sets of destroy operators allows for a very thorough search procedure, with the objective to destroy inefficient parts of a solution in both the subproblems and in the overall problem. Set D^p consists of the operators of the order picking subproblem of Section 4.2 and as discussed in D'Haen et al. [2023], i.e., $D^p = \{$*Order_Random*, *Batch_DistanceSavings*, *Batch_Aisles*, *Batch_CoveringArea*$\}$. Set D^r consists of the operators of the vehicle routing subproblem as discussed in Section 4.3, i.e., $D^r = \{$*Order_Worst*, *Order_Earliness*, *Order_Tardiness*, *Order_CenterOfGravity*, *Order_RelatedDistance*, *Order_RelatedTime*, *Or-*

Algorithm 4: Integrated solution algorithm.

Input: *TotalRunTime*
Initialise Best, New and Current solutions (S^*, S^1, S^0, respectively);
Construct initial solution S^i;
S^*, S^0, $S^1 \leftarrow S^i$;
while *RunTime* $<$ *MaxRunTime* **do**
    Randomly select a set of destroy operators from $\{D^p, D^r, D^j\}$;
    For the chosen set, randomly select a destroy operator;
    Use destroy operator on S^1;
    Randomly select reinsertion in picking or routing schedule first;
    for *every removed order* **do**
        `// Sequence of orders is randomly chosen`
        Compute cutoff times to test for insertion;
        for *all cutoff times* **do**
            Greedy insertion in selected subproblem;
            Update release time or order due time based on first subproblem;
            Greedy insertion in second subproblem;
            Save insertion costs;
        Insert order in S^1 in its position with lowest insertion cost;
    if $(f^t(S^1) < f^t(S^*))$ **or** $(f^t(S^1) = f^t(S^1)$ **and** $f^d(S^1) < f^d(S^1))$ **then**
        $S^* = S^1$;
        $S^0 = S^1$;
    else if $(f^t(S^1) < f^t(S^0))$ **or** $(f^t(S^1) = f^t(S^0)$ **and** $f^d(S^1) < f^d(S^0))$ **then**
        $S^0 = S^1$;
    else if $(f^t(S^1) = f^t(S^0))$ **and** $(f^d(S^1) > f^d(S^0))$ **then**
        if $RandNum < e^{((f^d(S^0) - f^d(S^1))/Temperature)}$ **then**
            $S^0 = S^1$;
    Update Temperature;

der_Random, *Trip_Tardiness*, *Trip_Earliness*, *Trip_Random*}. Finally, set D^j consists of the operators of the joint problem, i.e., D^j = {*Order_Tardiness*, *Order_IdleTime*, *Order_Picking/RoutingDistribution*}. The operators of the joint problem work as follows:

- *Order_Tardiness*: compute the tardiness for every order, i.e., the positive difference between actual delivery time and latest delivery time. Remove the orders with the highest tardiness. This operator removes the same orders as the *Order_Tardiness* operator of the VRP subproblem.

- *Order_IdleTime*: for every order, compute the time between picking completion time of this order and the vehicle departure time for the order's delivery trip. Remove the orders with the largest idle time.

- *Order_Picking/RoutingDistribution*: for every order, compute (a) the time between arriving in the system (i.e., the release time in the order picking problem) and the picking completion time, (b) the time between its vehicle departure time at the depot and the order delivery time. Remove orders with the most skewed ratio between (a) and (b), i.e., where picking or routing requires a lot more time than the other subproblem.

First, one of these sets is selected randomly, after which a random operator from this set is selected. From the number of orders that should be removed, half of the orders is removed by the selected destroy operator, the other half is again removed by random order removals.

Irrespective the chosen set of destroy operators or the specific operator within a set, the orders are reinserted using the following approach. Testing all possible combinations of insertion positions in both subproblems before deciding on the best insertion position leads to a prohibitively large computational effort. Therefore, the orders are inserted in one of the subproblems first, randomly chosen between picking or routing. To reinsert the order in the first subproblem, a cutoff time between picking and routing is required, to be able to compute the insertion quality in different positions. If reinsertion happens in the order picking problem first, an order due time is required (the release time for the VRP is then based on the picking completion time). When reinserting in the VRP first, an order release time should be available (the due time for the order picking problem is then based on the order's actual vehicle departure time). Since it seems difficult to determine which cutoff time would lead to the best solution, multiple cutoff times are tested to insert the order. To keep the computation times reasonable, a greedy approach is implemented. First, one of the removed orders is chosen randomly. The decision on inserting the order first in the picking or routing problem is taken randomly. Insertion is then performed in the first subproblem in a greedy manner, i.e., the order is inserted in its lowest cost position, followed by a greedy insertion in the second problem. Then, for every cutoff time, the best insertion position and the associated costs for both the

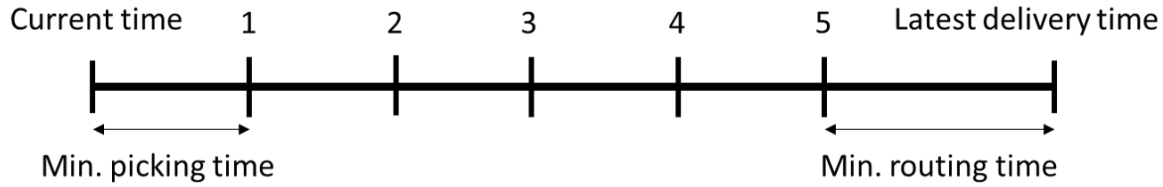


Figure 4: This figure shows how the cutoff times are set for a setting with five tested cutoff times, i.e., $k = 5$. The earliest possible cutoff time is based on the current time plus the minimum picking time (number 1 in the figure), the latest possible time is based on the minimum routing time (number 5 in the figure, or k in general). The times in between are set at equidistant intervals, depending on the number of tested cutoff times.

picking and routing part are saved. The cutoff time and position leading to the lowest insertion cost are then selected and used to actually reinsert the order in the picking and routing schedules. This process is repeated until all orders are inserted.

The possible cutoff times are set based on the number of cutoff times that should be used, denoted by the parameter k (with $k \geq 2$), and the minimum picking and routing time, graphically shown in Figure 4 for $k = 5$. The earliest cutoff time that may be used, is the required picking time to pick the order individually, without adding other orders to the batch, since this leads to the lowest possible picking time for this order and will be the earliest possible release time for the VRP. Similarly, the latest cutoff time is set based on the required routing time to deliver the order in a separate trip to the customer. The latest delivery time minus this routing time is chosen as latest cutoff time. The remaining cutoff times (i.e., $k - 2$) are set between the earliest and latest cutoff time in equidistant intervals.

To decide on the acceptance of a new solution, simulated annealing is included in the algorithm in the same way as in the VRP, i.e., tardiness increases are never allowed, while distance increases may be accepted with a decreasing probability.

4.7 Integrated iterative algorithm

Finally, an integrated iterative algorithm is developed, operating as a combination of the iterative and integrated algorithms discussed in Section 4.5 and Section 4.6, respectively. The integrated iterative algorithm operates as the iterative algorithm by performing multiple iterations over different subobjectives. However, in every iteration there are now three steps, as shown in Algorithm 5. In the first step, the order picking operations are optimised. Next, based on the final solution of the order picking problem, the order release dates for the VRP are set. In the second step, the VRP is optimised with the VRP algorithm. The final solution of the VRP is then used to set the order due times for the order picking problem. In the third step, the integrated algorithm is used to optimise

both problems at the same time. The final schedule is then used as input for the order picking problem for the next iteration. This iterative solving of order picking operations individually, vehicle routing operations individually and picking and routing together with the integrated algorithm is continued until the maximum number of iterations is reached.

Algorithm 5: Integrated iterative algorithm.

Input: *TotalRunTime, MaxIterations*
Initialise and construct initial picking solution S_P, routing solution S_R;
$MaxRunTimePicking = \frac{TotalRunTime}{3*MaxIterations}$`// Divide the available time evenly over the picking, routing and integrated algorithm and iterations`;
$MaxRunTimeRouting = \frac{TotalRunTime}{3*MaxIterations}$;
$MaxRunTimeIntegrated = \frac{TotalRunTime}{3*MaxIterations}$;
while $NoIterations < MaxIterations$ **do**
    $LNS_{Picking}(S_P, MaxRunTimePicking)$;
    Update order release times VRP based on picking completion times;
    $LNS_{Routing}(S_R, MaxRunTimeRouting)$;
    Update order due times in picking problem based on vehicle departure times;
    $LNS_{Integrated}(S_P, S_R, MaxRunTimeIntegrated)$;
    NoIterations = NoIterations + 1;

5 Computational experiments

This section gives an overview of the computational experiments, in which the algorithms from Section 4 are tested. For every algorthm, the computation time for the first optimisation step, i.e., at the start of the planning period, is fixed at 300 seconds. In all subsequent reoptimisation steps, only 60 seconds of computation time are given as the algorithms can start from a solution from a previous reoptimisation step. For the sequential algorithm, the computation time is evenly divided over the LNS for picking and the LNS for routing. In the iterative algorithm the total time is evenly divided over the two subproblems. For the integrated iterative algorithm, the total time is evenly divided over the picking problem, the routing problem and the integrated problem. However, for the iterative and integrated iterative algorithms, this assigned time is then once more divided by the number of times the algorithm iterates over the different search strategies. All algorithms are implemented in C++. The experiments are performed on an Intel Xeon Processor Gold 6140 at 2.3 gigahertz.

First, Section 5.1 introduces the problem instances on which the algorithms are tested. Next, Section 5.2 gives an overview of the parameter tuning for the different algorithms. Finally, the results of the different algorithms are shown and discussed in Section 5.3.

5.1 Problem instances

To test the algorithms, a series of new problem instances is developed as, to the best of our knowledge, no instances exist for the o-IOPVRP. The developed instances are made publicly available at **insert website**. An overview of the parameters regarding the layout of the warehouse and general VRP characteristics, is given in Table 4. Note that the resources for picking (i.e., the number of order pickers) and routing (i.e., the number of vehicles) were set such that the time required to perform both processes is somewhat balanced on average. Since, on average, delivery requires much more time per order than picking, the number of vehicles available is also higher than the number of order pickers.

The delivery locations of customers are randomly spread over the delivery area, with the distance between two locations computed by the Euclidean distance. The depot is located in the middle of the delivery area. The time window at every location is randomly set, rounded to the nearest minute. The upper bound of the time window is chosen between two hours after the start of the planning period and the end of the planning period, i.e., eight hours after the start of the planning period. The lower bound of a time window is set based on the upper bound and the time window width (which is discussed later). The delivery vehicles are capacitated, however, capacity is set sufficiently high in the experiments, making this constraint in fact non-binding in all instances. This setting was adopted since e-commerce orders are usually small, making capacity in general not an issue.

The warehouse parameters are based on those of D'Haen et al. [2023]. The number of order lines per order is randomly generated per order, based on an exponential distribution with mean 2, which gives an average of 2.6 order lines per order after rounding. The item locations in the warehouse are randomly chosen, based on an across-aisle storage policy, where three storage classes are used. Classes A, B and C are assigned 1/6, 1/3 and 1/2 of the SKUs, and 60%, 30% and 10% of the picks, respectively.

The problem instances are developed based on a factorial experimental design, to allow studying the performance of the algorithms in different operating contexts. The factorial design consists of two factor levels for every factor. The factors considered are the number of orders, the size of the delivery area, the order's urgency and the time window width. The factor levels for every factor are shown in Table 5.

To test the impact of operating on a different scale, unique order lists are generated for instances with an average (μ) of 300 and 600 orders. All other factors are variations on these basic lists of orders, allowing for a fair comparison between the factor levels,

Table 4: Warehouse and routing parameter values.

Warehouse parameters	Parameter value
Number of warehouse blocks	2
Number of aisles	12
Number of sub-aisles	24
Locations per aisle	240
Storage policy	Across-aisle
Storage location length	1.3 m
Storage location width	0.9 m
Pick aisle width	3.0 m
Cross-aisle width	6.0 m
Picker travel velocity	1 m/s
Batch setup time	180 seconds
Search and pick time	10 seconds
Batch capacity in number of orders	10
Avg. number of order lines per order	2.6
Number of order pickers	1 per 150 orders
Routing parameters	**Parameter value**
Vehicle velocity	50 km/h
Vehicle capacity	Unlimited
Number of vehicles	1 per 25 orders

Table 5: Factors and factor levels of the experimental design.

Factor	Factor levels	
Average number of orders	300	600
Size of delivery area	Small	Large
Order urgency (h)	[2,3]	[3,4]
Time window width (h)	1	2

since all differences are attributable to changing the factor levels, not changes in the list of orders (e.g., the number of orders in an instance and the coordinates of the customer locations are identical). Although the average number of orders is set at 300 and 600 for small and large order lists, respectively, randomness is introduced to mimic a real-life situation, in which an estimate on the number of orders may be available, but the exact number of orders is unknown. The exact number of orders is randomly generated based on a triangular distribution with a minimum, mode and maximum of 0.9μ, μ and 1.1μ, respectively. The size of the delivery area can be small or large, with large instances requiring double the time to travel between two customer locations compared to small instances (i.e., the distances between locations are doubled). The factor order urgency determines the available time for the picking and routing operations. It is the time interval between the latest delivery time (i.e., the upper bound of the time window) and the order's arrival time in the system (following the definition of urgency of van Lon et al. [2016]). Given the latest delivery time of an order, the order's arrival time is randomly generated to conform to an order urgency within the time interval shown in Table 5. Finally, the time window width sets the possible delivery times at the customer's location. Early delivery, i.e., before the start of the time window, is never allowed, late delivery is allowed but minimised by the algorithms. Combining all factors results in 2 $\times$ 2 $\times$ 2 $\times$ 2, or 16 factor combinations. For every factor combination, ten instances are generated and solved four times, once by every algorithm, i.e., the sequential, iterative, integrated and integrated iterative algorithm.

5.2 Parameter tuning

In the developed algorithms, a few parameters are tuned to optimise the algorithm performance. For the LNS on the order picking problem, the settings of D'Haen et al. [2023] are used, since the algorithm is similar. The LNS on the VRP is newly developed. Here, preliminary experiments have shown that all destroy and repair operators are often able to find improving solutions. All of the operators are thus included in the algorithm. In order to avoid confusion, the parameters of the simulated annealing and the destroy percentage for the VRP are also set as in the order picking algorithm. The initial temperature in the simulated annealing is therefore equal to 10% of the initial distance and every 1% of the computation time, the temperature is reduced to 90% of the current value. The destroy percentage is set at 10% for the LNS algorithms on the subproblems as well as on the overall problem.

Two parameters for which tuning seems required on a larger scale are the number of times the iterative and integrated iterative algorithms should iterate over all different search strategies, and the number of cutoff times that are tested for order insertion in the integrated and integrated iterative algorithms. To test the algorithm performance for the different parameter combinations, two order lists were selected, one with 300 and one with 600 orders. For each of these, the eight combinations of the other factors from

Table 6: Parameter tuning results. The best parameter combinations are highlighted in bold.

Algorithm	Iterations	cutoffs	Tardiness	Distance
Iterative	10	-	36,031	543,386
	20	-	**34,650**	538,785
	30	-	35,207	540,381
Integrated	-	3	29,833	538,068
	-	**5**	**28,414**	535,378
	-	7	32,703	542,656
Integrated iterative	10	3	27,984	534,351
	10	5	27,238	534,600
	10	7	27,951	533,191
	20	3	26,908	533,598
	20	**5**	**25,387**	532,916
	20	7	27,509	532,975
	30	3	27,044	532,929
	30	5	27,832	530,811
	30	7	28,645	533,933

the experimental design were considered, for a total of 32 instances. Every instance is solved three times by every algorithm and for every parameter combination. For the number of times the algorithms should switch search strategy, 10, 20 or 30 iterations for the iterative and integrated iterative algorithm are considered. Moreover, 3, 5 and 7 cutoff times are considered for the integrated and integrated iterative algorithm. Table 6 shows the average results of the parameter tuning, with the best result highlighted in bold. Based on these results, iterating 20 times over the different algorithms shows the best performance for both the iterative and integrated iterative algorithm. The best number of cutoff times is 5 for both the integrated and integrated iterative algorithm. These values are used in all further experiments.

5.3 Results

This section gives an overview of the results. All instances have been solved by the sequential, iterative, integrated and integrated iterative algorithms, using the same computation time per optimisation step to obtain a fair comparison between all algorithms. The assigned computation time is 300 seconds in the first optimisation step, and 60 seconds for all subsequent reoptimisation steps. Note that when solving an instance, a full run is considered, i.e., not just a single optimisation step but the optimisation at the start of the planning period and all further reoptimisation steps until the end of the

planning period are performed. The reported objective values are the actual tardiness and distance of the executed vehicle trips. First, Section 5.3.1 gives an overview of the performance of the different optimisation algorithms. Next, for the best performing algorithm, the impact of the factors of the factorial design are discussed in Section 5.3.2. Detailed results per instance are available at **insert website**.

To analyse whether there are differences between the algorithms and between the factor levels, a mixed ANOVA on both tardiness and routing distance is used. A mixed ANOVA is used because there are both independent, i.e., between groups variables (i.e., the average number of orders in an instance) and dependent observations, i.e., repeated measures variables (i.e., the different optimisation algorithms, the size of the delivery area, the order urgency and the time window width). To get reliable results from a mixed ANOVA, three assumptions should hold: normality, homogeneity of variance and sphericity of the covariance matrix. Normality and homogeneity of variance should not pose any problems due to the balanced experimental design used in the experiments [Field, 2013]. Sphericity is tested with Mauchley's test and is not violated for most variables. In the occasions where it is violated, the Greenhouse-Geisser correction is used to correct the inflated F-test type I error rate [Field, 2013]. The results of the ANOVA are shown in Appendix A. In case Mauchley's test of sphericity was violated, the corrected results are reported. Based on these results, conclusions on whether the impact of different algorithms and instance characteristics is statistically significant can be drawn, i.e., if the p value in the last column of the tables is smaller than 0.05, we conclude that the characteristic has a statistically significant impact on tardiness or distance, read in Table 9 and Table 10, respectively. For both tardiness and distance, the impact of the selected algorithm as well as all instance characteristics is statistically significant (all have p-values of 0.000). In the next sections, the impact of the algorithms and instance characteristics and their statistical significance (based on the ANOVA results) is discussed in more detail.

5.3.1 Performance of optimisation algorithms

The results of the different algorithms are shown on Figure 5 and in Table 7. The figure shows the average tardiness per order and average distance per order over all instances for every algorithm. In the figure, the bars show the tardiness and the dots indicate the distance, read on the left and right axis, respectively. The results show that solving the o-IOPVRP with the sequential algorithm leads to the worst performance. Clearly, the interaction between order picking and vehicle routing should be addressed. By using an iterative approach, highly improved solutions to the problem can be obtained, with the average tardiness per order reducing by more than 67% compared to a sequential approach. However, the iterative approach is again statistically significantly outperformed by an integrated approach, with a tardiness reduction of almost 15% when using the integrated instead of the iterative algorithm. Finally, the graph shows that using the in-

Table 7: Algorithm performance over all instances (i.e., 160 in total). The second column shows the number of times an algorithm obtained the best solution including ties with other algorithms. The third column shows the number of times an algorithm obtained a strictly better solution than any other algorithm.

Algorithm	Best found solutions (#)	Best found solutions excl. ties (#)
Sequential	1	0
Iterative	99	3
Integrated	122	25
Integrated iterative	132	35

tegrated iterative algorithm leads to the best results on average, with a further tardiness reduction of 3.8%, though the difference with the integrated algorithm is not statistically significant. Table 7 shows the number of times each algorithm obtained the best found solution, as well as the number of times it obtained a strictly better solution than the other solutions. Again, it is clear that addressing the interaction between picking and routing is required, since the sequential algorithm performs much worse than all other algorithms. The integrated algorithms once again do better than the iterative one, with the integrated iterative algorithm performing best.

Next to a remarkable drop in tardiness, the average routing distance per order is reduced considerably by the algorithms considering the interaction between picking and routing. More specifically, the distance is reduced by almost 20% by the iterative, integrated or integrated iterative algorithms compared to the sequential algorithm, with the same ordering of the algorithms as for tardiness, i.e., integrated iterative performing better than integrated, which does better than iterative. While the difference in distance between the integrated and integrated iterative algorithm is not statistically significant at the 5% level (but it is significant at the 10% level), the integrated iterative algorithm is significantly better than both the iterative and sequential algorithms.

The results indicate that using an integrated approach is very beneficial, leading to highly improved results by creating a better fit between the picking and routing schedules. Furthermore, there is an indication that a cycle of setting good cutoff times between picking and delivery by using the integrated algorithm, and next optimising the order picking and vehicle routing problems individually based on these cutoff times, leads to the best results. Preliminary experiments showed that performing this cycle multiple times, i.e., an iterative approach, leads to better results than a single run of the integrated algorithm followed by a single order picking and vehicle routing optimisation. Combined, it can be seen that using an algorithm that is better tailored to the problem at hand results in a highly improved service level accompanied by large efficiency improvements, with the best performance obtained by the integrated iterative algorithm.

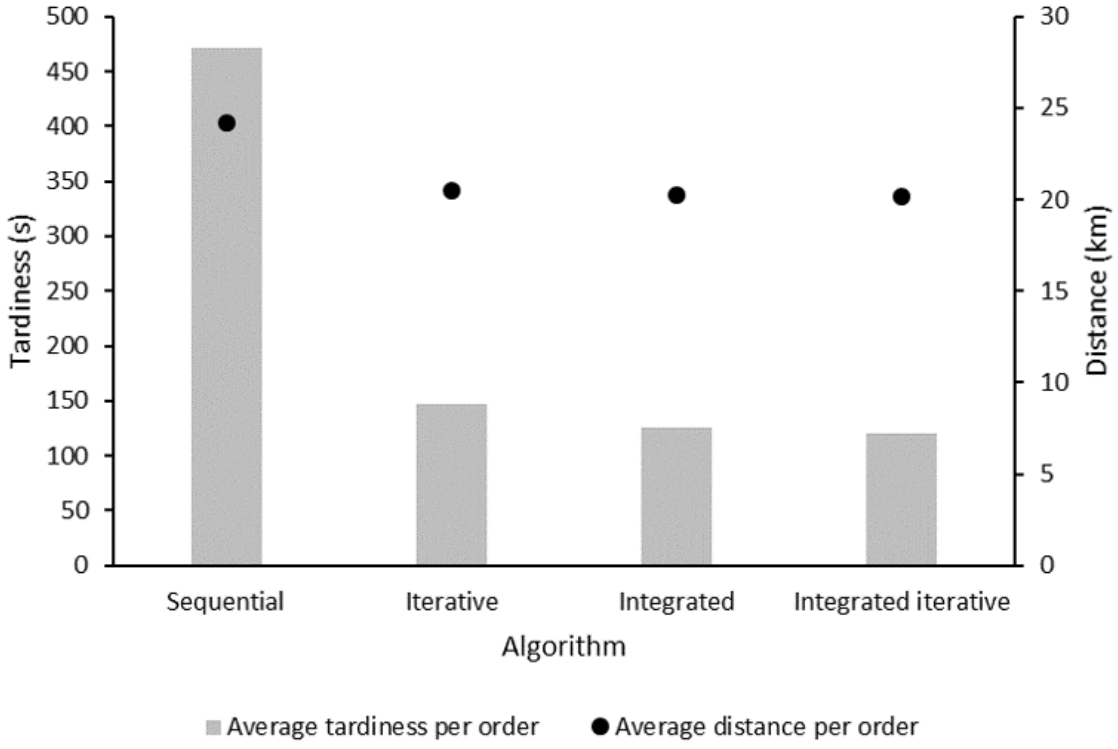


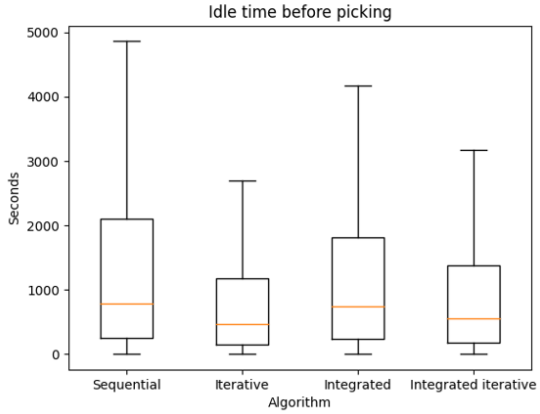
Figure 5: Average tardiness and travel distance per order over all instances for the different optimisation algorithms. Tardiness is read on the left axis, travel distance on the right axis.

Table 8: Average tardiness, distance and order pick time (OPT) per order and the number of batches and trips for the different solution algorithms.

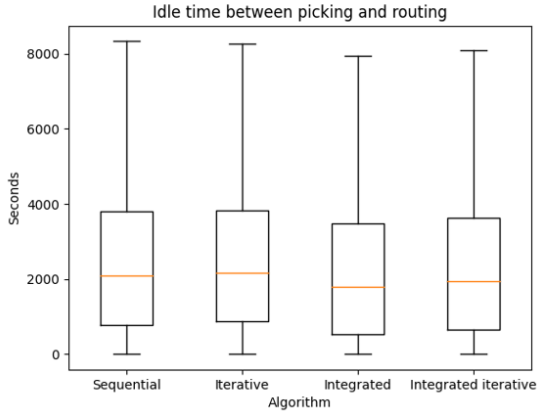
Algorithm	Tardiness (s)	Distance (km)	OPT (s)	Batches	Trips
Sequential	471.61	24.22	168.29	108.79	56.80
Iterative	146.75	20.47	132.13	62.43	39.02
Integrated	125.12	20.26	133.88	60.90	38.05
Integrated iterative	120.36	20.16	132.60	63.03	37.91

Based on the results, differences in solution structure between the different algorithms can be explored. Taking a look at the average number of batches and trips per algorithm, as shown in Table 8, leads to a first interesting observation. The number of batches and trips for the sequential algorithm is much larger than for the other algorithms. This indicates that both for the order picking and vehicle routing subproblems, the cutoff times are, in general, not well set. In the order picking problem, for example, a too early cutoff time results in order due times that are too stringent. This results in less efficient batches, since some combinations of orders are rejected due to their resulting tardiness. Depending on the due times of the orders, this may lead to the creation of two separate, partially filled batches, instead of one full batch, leading to an increase in the number of batches per instance. On the other hand, order due dates that are too late, may result in very efficient picking operations, but may lead to issues in the routing process, since there is too little time to efficiently route all orders, creating many trips with only few orders. Furthermore, since in the sequential approach no information about the vehicle routing problem is taken into account while optimising the order picking operations, there is a complete lack of fit between both problems. Orders of a single batch may belong to destinations very far apart, with non-matching time windows. This prohibits the creation of efficient routes, e.g., by resulting in two separate routes instead of one larger route, leading to an increase in the number of trips and travelled distance per order.

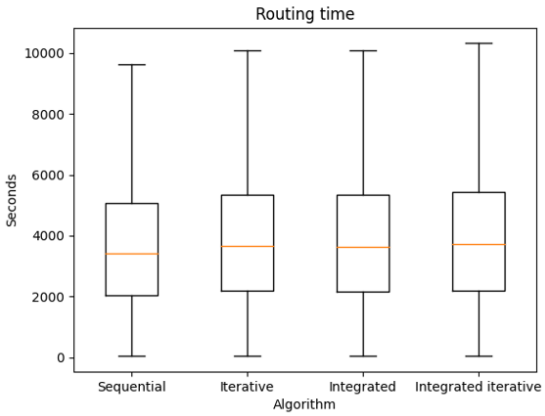
Second, the lack of fit between the picking and routing schedules can be illustrated by looking at the timestamps of different steps in the order handling process, graphically shown in Figure 6. When looking at the time between the release time of an order in the system and the time a picker starts picking the batch containing this order, an interesting insight emerges. The sequential algorithm is hard to interpret, since its tardiness is much larger than for the other algorithms and it has a much larger number of batches and trips. The integrated algorithm clearly results in different solutions compared to the iterative approach. On average, orders have to wait a little longer before a picker starts working on them, and the variation in this waiting time is much larger. Nevertheless, the average order pick time per order is barely higher than for the iterative approach,



(a)



(b)



(c)

Figure 6: The difference in solution structure between the algorithms is visible in the boxplots.

as shown in Table 8. This indicates that the integrated approach really improves the fit between picking and routing, by postponing less urgent orders in the picking process, to first pick orders that allow for improved routing solutions. The integrated iterative algorithm is situated between the iterative and integrated algorithms, exploiting the best of both algorithms.

For the idle time between picking and routing, i.e., the difference between picking completion time of an order's batch and the starting time of this order's trip, using the integrated or integrated iterative algorithm clearly results in a reduction in idle time. Although somewhat counter-intuitively, this idle time does not seem to explain the difference in performance between the algorithms well. The sequential and iterative algorithm have an almost equal idle time between picking and routing, although the outliers to the upside are somewhat smaller for the iterative algorithm. Nevertheless, the iterative algorithm performs much better regarding the objective function values. The reduced idle times for the integrated and integrated iterative algorithms can be explained by the used destroy operators. One of the operators of the integrated algorithm is removing orders with the largest idle time between picking and routing, resulting in an overall lower idle time in the final solution. Interestingly, despite not being a strong indicator of good performance when looking at the idle time in the final solutions, this operator was selected during preliminary testing since it did have an effect in finding better solutions.

The integrated iterative algorithm leading to the best performance can be explained by the average routing time of an order, computed as the time between starting the trip at the depot and the order being delivered at the customer's location. On average, this time is largest for the integrated iterative algorithm, despite this algorithm having the lowest tardiness and distance, which indicates an increased efficiency in the delivery routes. More specifically, by exploiting the interaction between picking and routing, it is possible to have orders in a vehicle for a longer time without violating the order due times, i.e., an order can be given priority in the picking process if this order is best loaded into a trip leaving early for efficient delivery, even if this order's time window is far in the future. The integrated algorithm seems to result in the best fit between picking and routing, e.g., when looking at the idle time before picking, the integrated algorithm has a larger average idle time compared to the other algorithms because of the consideration of the delivery process. However, the integrated iterative algorithm has the additional feature of focusing on optimising each of the subproblems individually, i.e., it balances the use of global neighbourhoods which focus on setting good cutoff times and local neighbourhoods which are efficient in optimising each of the subproblems. This way, the integration between picking and routing can be achieved by optimisation with the integrated algorithm, which leads to a good indication of the cutoff times to be used for every order. Next, the subproblems are optimised with a stronger algorithm for the picking and routing problem separately, while taking the cutoff times into account.

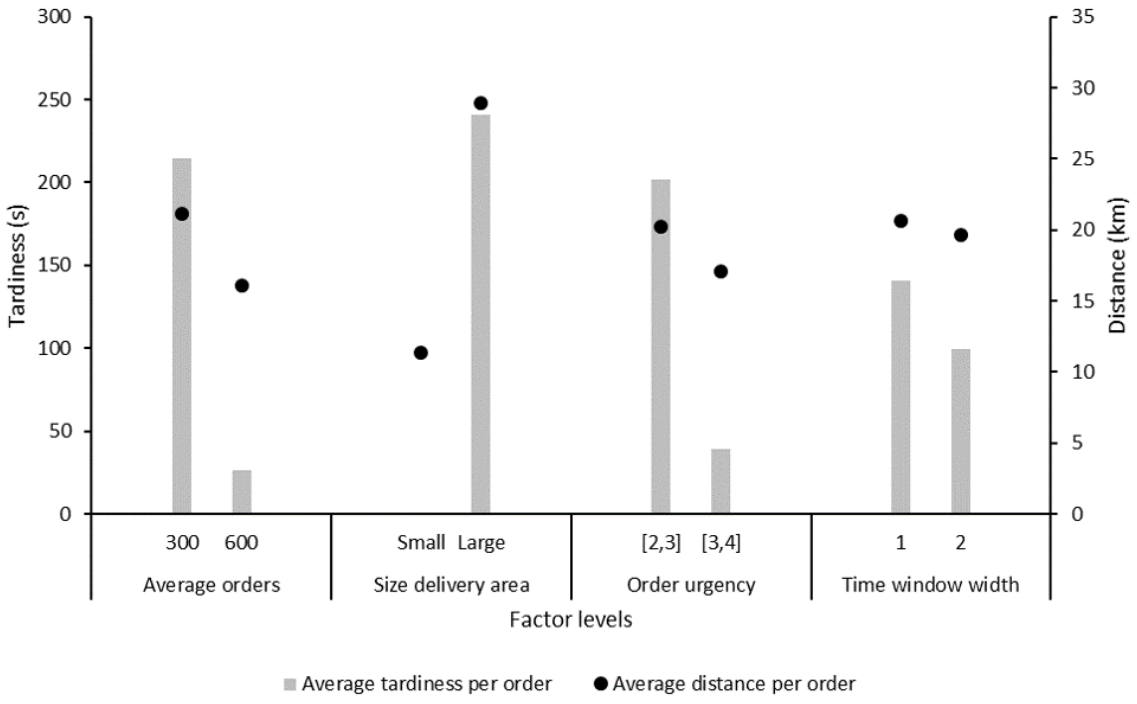


Figure 7: Tardiness and travel distance for the factor levels of the factorial design. Tardiness is read on the left axis, travel distance on the right axis.

5.3.2 Impact of operating context

In this section, the performance of the integrated iterative algorithm is discussed for different operating contexts, tested on the experimental design outlined in Table 5. The interpretation of the results for the other algorithms is very similar, but for clarity only the integrated iterative algorithm is reported. Figure 7 shows the average tardiness and distance per order for the different settings. All differences in tardiness and distance between the factor levels are statistically significant.

First, operating on a larger scale, i.e., handling more customer orders within a certain time period, clearly leads to economies of scale. Both the average tardiness per order and the average distance per order can be reduced by handling more orders per day. This result is not surprising, since the chance of receiving two very similar orders for picking or routing increases. More specifically, two orders with item locations in the warehouse close to each other can be picked more efficiently by picking them in the same batch, and two orders with delivery locations close to each other, if having somewhat related time windows, can be delivered more efficiently by a single vehicle trip.

Next, when looking at the size of the delivery area, it is immediately clear that operating in a larger area is more difficult if the same operating resources are available,

i.e., the same number of order pickers and delivery vehicles. In the experiments, there was never tardiness when operating in a small delivery area, but quite some tardiness in the large delivery area. Interestingly, while the travel distances were doubled in a large delivery area compared to a small area, the average distance per order increases by over 150% instead of doubling. This effect is most likely caused by the difficulty of reaching every customer in time, since the distances have doubled but the time windows remained identical. Additionally, doubling the distance between two locations leads to less customers being reachable in a certain period of time when starting from a certain location. This results in less efficient delivery routes, leading to an increase in the distance per order along with the tardiness increase.

When the time interval between the latest delivery time and the order arrival time (i.e., the customer ordering online) increases, it becomes significantly easier to deliver orders in time. Tardiness can be reduced by 80% if on average 3 to 4 instead of 2 to 3 hours are available, with an accompanying distance reduction of 15%. The tardiness reduction is expected since the available order handling time increases. The distance reduction can be explained by an increased number of orders available in the system, and these orders are known more in advance. By taking these orders into account earlier during the planning period, a more efficient schedule can be constructed for both the order picking and order delivery processes.

Finally, the time window width also impacts the operating performance. Tardiness is reduced by almost 30% when the delivery time window is two instead of one hour wide. Furthermore, there is an associated distance reduction of 4.7%. These results are explained as follows. By offering a longer accepted delivery interval, it is easier to deliver two orders whose delivery locations are closely located to one another in time with a single vehicle, leading to a shorter distance to be travelled. This in turn makes the operations more efficient, which also positively impacts the tardiness.

Combined, it is clear that the customer expectations and the provided service level by the company dramatically influence the operating efficiency. Expecting deliveries within a time window of one hour, while only ordering on average 2.5 hours before the latest delivery time, results in a much more difficult situation for the retailer compared to time windows of two hours and an average of 3.5 hours of order handling time. If no additional resources are available, but customers accept longer time windows and a bit longer delivery delay, the tardiness can be reduced by more than 90%, while the distance is reduced by over 18%. Furthermore, this efficiency improvement even leads to benefits external to the company, as the large reduction in distance will also lead to highly reduced externalities, e.g., regarding greenhouse gas emissions, noise pollution and traffic congestion. If customers are a little less demanding, the environmental impact of their purchases can be dramatically reduced. Based on these results, retailers could highlight the beneficial environmental effect of selecting a longer time window or longer delivery delay during the ordering process, to motivate customers to select the environmentally friendly option.

6 Conclusion

With the increasing importance of e-commerce sales for retailers, efficiently organising the order handling process becomes indispensable. The ordered items should be picked in the warehouse first, followed by delivery to the customer. The order picking and order delivery processes are traditionally handled separately. However, improved operating performance can be achieved by taking the interaction between order picking and order delivery into account. In this paper, several contributions are achieved. We studied different metaheuristic algorithms to take the interaction between picking and delivery into account, each algorithm integrating picking and routing to a different extent. Furthermore, a dynamic context is considered, in which new orders arrive during the planning period, which has not been studied before for the integrated order picking and vehicle routing problem. Based on a factorial design, with different factor levels to mimic different real-life operating contexts, the algorithms have been tested. The results showed that optimising the order picking and delivery operations in an integrated manner results in highly improved performance compared to a sequential optimisation approach, with an analysis of the solution structure indicating a better fit between the picking and routing operations when using integrated decision-making. The integrated iterative algorithm showed the best performance, indicating that iterating over setting the order cutoff times between picking and routing using global neighbourhoods, and optimising picking and routing decisions individually with local neighbourhoods, results in the best solutions.

Next, the integrated iterative algorithm was tested for different operating contexts. Results showed that there are significant economies of scale, with notable improvements in operating efficiency if more orders are handled. A valuable insight from the experiments is the large cost of offering high customer service levels. The company, but also the customers should think twice about their environmental impact as well, since a rather small reduction in customer service level, i.e., accepting delivery within 3.5 instead of 2.5 hours on average and allowing delivery within a two instead of one hour time window, reduces the routing distance and thus the associated environmental impact by more than 18%. This result can be used to propose multiple options to the customers, with a reduction in delivery charges for less demanding customers.

In future research, the impact of using stochastic information can be studied. If companies can use their historical order data, they may be able to make predictions about future orders. This information could then be used to develop more advanced dispatching strategies that consider order pickers (or even vehicles) to wait for future orders to improve the efficiency of the operations. In this study, all items are available in a single warehouse. Future studies could look into the effect of having multiple warehouses to perform the order picking operations, possibly with order consolidation to only have one delivery at the customer's location. Finally, the use of more environmentally friendly options could be explored, e.g., using electric vehicles or cargo-bikes in an urban setting,

adding constraints to the vehicle routing problem.

Acknowledgement

Ruben D'Haen is funded by the Research Foundation Flanders [grant number FWO-11J1721N]. This work is also supported by an FWO junior research project [G021422N]. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government.

Declarations of Interest

None.

References

A. Alnaggar, F. Gzara, and J. H. Bookbinder. Crowdsourced delivery: A review of platforms and academic literature. *Omega*, 98:102139, Jan. 2021. ISSN 0305-0483. doi: 10.1016/j.omega.2019.102139.

C. Archetti and L. Bertazzi. Recent challenges in Routing and Inventory Routing: E-commerce and last-mile delivery. *Networks*, 77(2):255–268, 2021. ISSN 1097-0037. doi: 10.1002/net.21995.

C. Archetti, D. Feillet, and M. G. Speranza. Complexity of routing problems with release dates. *European Journal of Operational Research*, 247(3):797–803, Dec. 2015. ISSN 0377-2217. doi: 10.1016/j.ejor.2015.06.057.

C. Archetti, D. Feillet, A. Mor, and M. G. Speranza. An iterated local search for the Traveling Salesman Problem with release dates and completion time minimization. *Computers & Operations Research*, 98:24–37, 2018. ISSN 0305-0548. doi: 10.1016/j.cor.2018.05.001.

C. Archetti, D. Feillet, A. Mor, and M. G. Speranza. Dynamic traveling salesman problem with stochastic release dates. *European Journal of Operational Research*, 280 (3):832–844, Feb. 2020. ISSN 0377-2217. doi: 10.1016/j.ejor.2019.07.062.

K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99: 300–313, Sept. 2016. ISSN 0360-8352. doi: 10.1016/j.cie.2015.12.007.

D. Cattaruzza, N. Absi, and D. Feillet. The Multi-Trip Vehicle Routing Problem with Time Windows and Release Dates. *Transportation Science*, 50(2):676–693, Jan. 2016. ISSN 0041-1655. doi: 10.1287/trsc.2015.0608. Publisher: INFORMS.

M.-C. Chen and H.-P. Wu. An association-based clustering approach to order batching considering customer demand patterns. *Omega*, 33(4):333–343, Aug. 2005. ISSN 0305-0483. doi: 10.1016/j.omega.2004.05.003.

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, Jan. 2015. ISSN 0925-5273. doi: 10.1016/j.ijpe.2014.09.029.

R. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2): 481–501, Oct. 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2006.07.009.

R. D'Haen, K. Braekers, and K. Ramaekers. Integrated scheduling of order picking operations under dynamic order arrivals. *International Journal of Production Research*, 61(10):3205–3226, May 2023. ISSN 0020-7543. doi: 10.1080/00207543.2022.2078747. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207543.2022.2078747.

M. Drexl. Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316, 2012. ISSN 0041-1655. Publisher: INFORMS.

A. Field. *Discovering Statistics using IBM SPSS Statistics*. SAGE, 2013.

M. A. Klapp, A. L. Erera, and A. Toriello. The One-Dimensional Dynamic Dispatch Waves Problem. *Transportation Science*, 52(2):402–415, Mar. 2018. ISSN 0041-1655. doi: 10.1287/trsc.2016.0682. Publisher: INFORMS.

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, Nov. 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.03.075.

W. Li, Y. Wu, P. N. R. Kumar, and K. Li. Multi-trip vehicle routing problem with order release time. *Engineering Optimization*, 52(8):1279–1294, Aug. 2020. ISSN 0305-215X. doi: 10.1080/0305215X.2019.1642880. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/0305215X.2019.1642880.

W. Li, K. Li, P. N. R. Kumar, and Q. Tian. Simultaneous product and service delivery vehicle routing problem with time windows and order release dates.

Applied Mathematical Modelling, 89:669–687, Jan. 2021. ISSN 0307-904X. doi: 10.1016/j.apm.2020.07.045.

S. Moons, K. Ramaekers, A. Caris, and Y. Arda. Integration of order picking and vehicle routing in a B2C e-commerce context. *Flexible Services and Manufacturing Journal*, 30(4):813–843, Dec. 2018. ISSN 1936-6590. doi: 10.1007/s10696-017-9287-5.

S. Moons, K. Braekers, K. Ramaekers, A. Caris, and Y. Arda. The value of integrating order picking and vehicle routing decisions in a B2C e-commerce environment. *International Journal of Production Research*, 57(20):6405–6423, Oct. 2019. ISSN 0020-7543. doi: 10.1080/00207543.2019.1566668.

B. H. Ojeda Rios, E. C. Xavier, F. K. Miyazawa, P. Amorim, E. Curcio, and M. J. Santos. Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering*, 160:107604, Oct. 2021. ISSN 0360-8352. doi: 10.1016/j.cie.2021.107604.

M. Ostermeier, A. Holzapfel, H. Kuhn, and D. Schubert. Integrated zone picking and vehicle routing operations with restricted intermediate storage. *OR Spectrum*, 44(3): 795–832, Sept. 2022. ISSN 1436-6304. doi: 10.1007/s00291-021-00664-7.

V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, Feb. 2013. ISSN 0377-2217. doi: 10.1016/j.ejor.2012.08.015.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, Aug. 2007. ISSN 0305-0548. doi: 10.1016/j.cor.2005.09.012.

H. N. Psaraftis, M. Wen, and C. A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016. ISSN 1097-0037. doi: 10.1002/net.21628.

M. Reimann, K. Doerner, and R. F. Hartl. D-Ants: Savings Based Ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591, Apr. 2004. ISSN 0305-0548. doi: 10.1016/S0305-0548(03)00014-5.

D. Reyes, A. L. Erera, and M. W. P. Savelsbergh. Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research*, 266(1):29–34, Apr. 2018. ISSN 0377-2217.

A. Rijal, M. Bijvank, and R. de Koster. Dynamics between warehouse operations and vehicle routing. *Production and Operations Management*, n/a (n/a), Aug. 2023. ISSN 1937-5956. doi: 10.1111/poms.14051. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/poms.14051.

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, Feb. 2011. ISSN 0736-5845. doi: 10.1016/j.rcim.2010.06.011.

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, Dec. 2017. ISSN 0377-2217. doi: 10.1016/j.ejor.2017.04.038.

D. Schubert, A. Scholz, and G. Wäscher. Integrated order picking and vehicle routing with due dates. *OR Spectrum*, 40(4):1109–1139, Oct. 2018. ISSN 1436-6304. doi: 10.1007/s00291-018-0517-3.

D. Schubert, H. Kuhn, and A. Holzapfel. Same-day deliveries in omnichannel retail: Integrated order picking and vehicle routing with vehicle-site dependencies. *Naval Research Logistics (NRL)*, 68(6):721–744, 2021. ISSN 1520-6750. doi: 10.1002/nav.21954. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.21954.

P. Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, Lecture Notes in Computer Science, pages 417–431, Berlin, Heidelberg, 1998. Springer. ISBN 978-3-540-49481-2. doi: 10.1007/3-540-49481-2_30.

B. C. Shelbourne, M. Battarra, and C. N. Potts. The Vehicle Routing Problem with Release and Due Dates. *INFORMS Journal on Computing*, 29(4):705–723, Nov. 2017. ISSN 1091-9856. doi: 10.1287/ijoc.2017.0756. Publisher: INFORMS.

N. Sluijk, A. M. Florio, J. Kinable, N. Dellaert, and T. Van Woensel. Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*, 304 (3):865–886, Feb. 2023. ISSN 0377-2217. doi: 10.1016/j.ejor.2022.02.022.

R. Soares, A. Marques, P. Amorim, and S. N. Parragh. Synchronisation in vehicle routing: Classification schema, modelling framework and literature review. *European Journal of Operational Research*, Apr. 2023. ISSN 0377-2217. doi: 10.1016/j.ejor.2023.04.007.

J. T. Soman and R. J. Patil. A scatter search method for heterogeneous fleet vehicle routing problem with release dates under lateness dependent tardiness costs. *Expert Systems with Applications*, 150:113302, July 2020. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.113302.

P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

M. Ulmer. Delivery deadlines in same-day delivery. *Logistics Research*, 10(3):1–15, 2017. ISSN 1865-0368. doi: 10.23773/2017_3.

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, Nov. 2017. ISSN 0377-2217. doi: 10.1016/j.ejor.2017.03.069.

T. van Gils, K. Ramaekers, A. Caris, and R. B. M. de Koster. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, 267(1):1–15, May 2018. ISSN 0377-2217. doi: 10.1016/j.ejor.2017.09.002.

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, Sept. 2019. ISSN 0377-2217. doi: 10.1016/j.ejor.2019.03.012.

W. J. A. van Heeswijk, M. R. K. Mes, and J. M. J. Schutten. The Delivery Dispatching Problem with Time Windows for Urban Consolidation Centers. *Transportation Science*, 53(1):203–221, Feb. 2019. ISSN 0041-1655. doi: 10.1287/trsc.2017.0773. Publisher: INFORMS.

R. R. S. van Lon, E. Ferrante, A. E. Turgut, T. Wenseleers, G. Vanden Berghe, and T. Holvoet. Measures of dynamism and urgency in logistics. *European Journal of Operational Research*, 253(3):614–624, Sept. 2016. ISSN 0377-2217. doi: 10.1016/j.ejor.2016.03.021.

S. A. Voccia, A. M. Campbell, and B. W. Thomas. The Same-Day Delivery Problem for Online Purchases. *Transportation Science*, 53(1):167–184, May 2017. ISSN 0041-1655. doi: 10.1287/trsc.2016.0732.

W. Yang, L. Ke, D. Z. W. Wang, and J. S. L. Lam. A branch-price-and-cut algorithm for the vehicle routing problem with release and due dates. *Transportation Research Part E: Logistics and Transportation Review*, 145:102167, Jan. 2021. ISSN 1366-5545. doi: 10.1016/j.tre.2020.102167.

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, Apr. 2016. ISSN 0360-8352. doi: 10.1016/j.cie.2016.02.001.

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, Sept. 2018. ISSN 0305-0483. doi: 10.1016/j.omega.2017.08.004.

J. Zhang, F. Liu, J. Tang, and Y. Li. The online integrated order picking and delivery considering Pickers' learning effects for an O2O community supermarket. *Transportation Research Part E: Logistics and Transportation Review*, 123:180–199, Mar. 2019. ISSN 1366-5545. doi: 10.1016/j.tre.2019.01.013.

Appendix

A ANOVA

Table 9: Results mixed factorial ANOVA on tardiness.

	Sum of squares	df	Mean square	F	p value
Main effects					
AverageOrders	211238.2	1	211238.2	337.744	0.000
Algorithm	14006420	1.685	8311978	1603.911	0.000
SizeArea	18497833	1	18497833	965.523	0.000
OrderUrgency	12596523	1	12596523	6418.32	0.000
TimeWindow	275598.8	1	275598.8	192.134	0.000
Two-way interaction					
AverageOrders × Algorithm	39469.51	1.685	23422.81	4.520	0.024
AverageOrders × SizeArea	6744303	1	6744303	352.029	0.000
AverageOrders × OrderUrgency	1255139	1	1255139	639.532	0.000
AverageOrders × TimeWindow	122039.9	1	122039.9	85.080	0.000
Algorithm × SizeArea	3023581	2.452	1232897	717.249	0.000
Algorithm × OrderUrgency	5855566	2.315	2529149	986.636	0.000
Algorithm × TimeWindow	4984.37	2.620	1902.186	1.545	0.219
SizeArea × OrderUrgency	6725464	1	6725464	3043.416	0.000
SizeArea × TimeWindow	319617.4	1	319617.4	222.548	0.000
OrderUrgency × TimeWindow	122.896	1	122.896	0.041	0.841
Residuals					
Between subjects	11257.89	18	625.439		
Within Algorithm	157188	30.332	5182.319		
Within SizeArea	344850.4	18	19158.36		
Within OrderUrgency	35326.6	18	1962.589		
Within TimeWindow	25819.39	18	1434.411		
Within Algorithm × SizeArea	75879.44	44.144	1718.924		
Within Algorithm × OrderUrgency	106827.8	41.674	2563.407		
Within Algorithm × TimeWindow	58076.95	47.166	1231.329		
Within SizeArea × OrderUrgency	39777.14	18	2209.841		
Within SizeArea × TimeWindow	25851.08	18	1436.171		
Within OrderUrgency × TimeWindow	53522.25	18	2973.458		

Table 10: Results mixed factorial ANOVA on distance.

	Sum of squares	df	Mean square	F	p value
Main effects					
AverageOrders	93.305	1	93.305	248.891	0.000
Algorithm	1854.61	1.883	984.928	1505.2	0.000
SizeArea	47613.38	1	47613.38	15815.67	0.000
OrderUrgency	2417.185	1	2417.185	1212.005	0.000
TimeWindow	106.447	1	106.447	309.187	0.000
Two-way interaction					
AverageOrders × Algorithm	26.202	1.883	13.915	21.266	0.000
AverageOrders × SizeArea	99.054	1	99.054	32.903	0.000
AverageOrders × OrderUrgency	72.736	1	72.736	36.471	0.000
AverageOrders × TimeWindow	12.787	1	12.787	37.142	0.000
Algorithm × SizeArea	67.635	2.315	29.219	99.709	0.000
Algorithm × OrderUrgency	338.585	2.686	126.063	564.186	0.000
Algorithm × TimeWindow	3.784	2.218	1.706	5.866	0.005
SizeArea × OrderUrgency	14.067	1	14.067	7.612	0.013
SizeArea x TimeWindow	9.463	1	9.463	30.730	0.000
OrderUrgency x TimeWindow	37.206	1	37.206	88.906	0.000
Residuals					
Between subjects					
Within Algorithm	6748	18	0.375		
Within SizeArea	54.189	18	3.011		
Within OrderUrgency	35.899	18	1.994		
Within TimeWindow	6.197	18	0.344		
Within Algorithm x SizeArea	12.210	41.666	0.293		
Within Algorithm x OrderUrgency	10.802	48.345	0.223		
Within Algorithm x TimeWindow	11.612	39.918	0.291		
Within SizeArea x OrderUrgency	33.266	18	1.848		
Within SizeArea x TimeWindow	5.543	18	0.308		
Within OrderUrgency x TimeWindow	7.533	18	0.418		