Integrated scheduling of order picking operations under dynamic order arrivals

DOI: 10.1080/00207543.2022.2078747
Handle: http://hdl.handle.net/1942/37567

Integrated scheduling of order picking operations under dynamic order arrivals

Ruben D'Haen[a,b], Kris Braekers[a], Katrien Ramaekers[a]

[a]Research Group Logistics, UHasselt, Agoralaan Building D, 3590 Diepenbeek, Belgium; [b]Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussel, Belgium

ARTICLE HISTORY
Compiled June 10, 2022

ABSTRACT
To remain competitive in the current e-commerce environment, warehouses are expected to handle customer orders as efficiently and quickly as possible. Previous research on order picking in a static context has shown that integrating batching, routing and scheduling decisions leads to better results than addressing these planning problems individually. In this study we propose an integrated solution approach that is able to deal with dynamic order arrivals, a problem often encountered in practice. Furthermore, we demonstrate the need to anticipate on future order arrivals to keep customer service levels high. We develop a new large neighbourhood search algorithm to solve the online, integrated batching, routing and scheduling problem. First, the algorithm is shown to outperform the current state-of-the-art static solution algorithm. Next, we develop an experimental design based on real-life data, to test the applicability of the model in different settings. The results of this experimental design are used to obtain insights on the particularity of this online, integrated problem. The effect of several real-life characteristics is demonstrated by using an ANOVA, leading to several managerial insights that may help companies to operate efficiently without jeopardising customer satisfaction.

KEYWORDS
Order batching; Order picking; Picker routing; Metaheuristics; Warehouse operations management

1. Introduction

In many supply chains, warehouse operations are a very important factor to consider. Efficient warehouse operations are not only indispensable for timely order delivery at the customer's location, but may also reduce the associated costs. Optimising the order picking operations is considered to be the best way to improve warehousing performance (de Koster, Le-Duc, and Roodbergen 2007).

This study focuses on operational decisions in manual, picker-to-parts order picking systems, as these systems are still used most often in practice (de Koster, Le-Duc, and

CONTACT Ruben D'Haen. Email: ruben.dhaen@uhasselt.be

Roodbergen 2007; Grosse et al. 2015; Kumar, Narkhede, and Jain 2021). In this context, operational decisions can be divided into multiple subproblems. Key subproblems include the order batching problem, which considers combining multiple customer orders into a single picking tour (Bozer and Kile 2008; Henn and Wäscher 2012), the picker routing problem, in which the visiting sequence of locations in a picking tour is determined (Theys et al. 2010), and the batch scheduling problem, in which batches are assigned to order pickers and the sequence of batches for every picker is determined, such that all orders are picked with minimum delay with respect to their due dates (Henn 2015). These individual subproblems have been studied extensively in the past. However, as these subproblems are in fact interrelated planning problems (van Gils et al. 2018a), better solutions are obtained by taking decisions in an integrated manner. Currently, research is moving towards solution methods that allow solving these integrated problems. In this context, the integrated problem of order batching, picker routing and batch assignment and sequencing is the most elaborate problem studied so far (Chen et al. 2015; Scholz, Schubert, and Wäscher 2017; van Gils et al. 2019). Optimisation of this integrated problem means that the picker routing problem cannot be solved by simple heuristics (e.g., S-shape). Instead, a real optimisation of the routes is required, whereby this routing problem is integrated with the batching decisions, as considered by the joint order batching and picker routing problem in the literature (Won and Olafsson 2005; Li, Huang, and Dai 2017). However, the additional complexity of time restrictions is integrated into this optimisation problem as well. Since order due dates are considered, the joint order batching and picker routing problem is extended to also include the batch scheduling problem. This integrated problem is NP-hard and solving it to optimality seems infeasible for problem instances of realistic size in reasonable computation times (van Gils et al. 2019). Therefore, all these studies propose (meta)heuristic solution approaches. Furthermore, up to now, only a static problem context has been considered, i.e., all orders are assumed to be known at the start of the planning horizon.

The rising importance of e-commerce requires very fast order picking operations. Same-day delivery operations are becoming more common, and companies like Amazon are currently even experimenting with one- or two-hour delivery options (Ulmer 2017). Additionally, our discussions with practitioners highlighted the importance of quickly responding to urgent orders in a spare parts warehouse setting. Here, orders could arrive until an hour before the shipping deadline. In order to be able to satisfy customer expectations in these contexts, and thus, in order to remain competitive, companies should be able to pick new orders very quickly. This requires the nearly instantaneous adaptation of picking schedules upon receiving customer orders, and thus leads to planning problems in which orders arrive dynamically throughout the planning horizon.

Literature on dynamic order picking problems is scarce. Most existing studies look at static problems with wave picking, in which orders are assigned to picking waves according to for example their due dates (Ardjmand et al. 2018; van Gils et al. 2019). After all orders of the current wave are completed, a next wave of orders is released in the system. Within a wave, all orders are batched and picked as efficiently as possible. For newly arriving urgent orders, waiting for the previous wave to be completed may lead to unacceptable delay. While a solution for this delay is picking these urgent orders individually, this will reduce the order picking efficiency because the benefits of order batching, a highly reduced travel time per order, are lost (de Koster, Le-Duc, and Roodbergen 2007).

Obtaining the benefits of order batching in a setting with dynamic order arrivals

requires an optimisation approach in which current order picking schedules are repeatedly updated through the day and even within a wave. Unlike in a static approach, the customer orders are not all known at the start of the planning period. Accounting for these dynamic order arrivals (i.e., arrivals during the planning period while operations are being executed), is possible with a dynamic or online solution approach. In a dynamic approach, every part of the current schedule can be modified, while an online approach fixes some parts of the current schedule (Vanheusden et al. 2022).

In this study, we propose an online solution method to cope with dynamic order arrivals while solving the integrated order batching, picker routing and batch scheduling problem (IBRSP). To the best of our knowledge, this study is the first one to solve the IBRSP while accounting for dynamic order arrivals. The contributions of this paper are multiple:

- The IBRSP in an online setting is formally introduced.
- A metaheuristic algorithm is developed to solve the IBRSP in the online setting. This large neighbourhood search algorithm is benchmarked on the static IBRSP to an existing state-of-the-art solution algorithm (van Gils et al. 2019) and shows excellent performance.
- An extensive numerical study, including a statistical analysis, is performed on a large set of real-life-based problem instances. This leads to insights on the impact of e.g., the number of orders and different settings to release orders in the picking system.
- As orders arrive dynamically during the planning horizon, we show the need to anticipate on future order arrivals to keep the customer service level high.

The remainder of this paper is structured as follows. First, in Section 2, an overview of related literature is given. Section 3 discusses the problem setting of the online IBRSP. Next, the developed metaheuristic algorithm is explained in Section 4, followed by the computational results in Section 5. Finally, managerial insights, a conclusion and future research opportunities are given in Section 6.

2. Literature review

This section discusses the literature specifically related to the online IBRSP. For more general discussions of literature on order picking operations and the importance of integrated decision making, we refer to de Koster, Le-Duc, and Roodbergen (2007) and van Gils et al. (2018b), respectively. In Section 2.1 an overview of research on the integration between the order batching, picker routing and batch scheduling problems in a static context is given. Next, Section 2.2 presents an overview of the literature on dynamic and online order picking problems.

2.1. *IBRSP*

Because the different order picking planning problems, e.g., order batching, picker routing and batch scheduling, are interrelated, better solutions can be obtained by solving these problems in an integrated way. The individual picker planning problems have been studied extensively before (an overview is given in the literature review of van Gils et al. (2018b)), whereas the integration of these problems is a more recent phenomenon (van Gils et al. 2019). For example, several studies focus on the integration of batching and routing decisions (e.g., Won and Olafsson (2005); Tsai, Liou,

and Huang (2008); Li, Huang, and Dai (2017); Attari et al. (2020)). However, only a few papers study the IBRSP in which batching, routing and scheduling decisions are addressed jointly. These papers are shown in Table 1. For a more elaborate overview of integrated approaches, including papers that are not integrating all three subproblems, we refer to van Gils et al. (2019).

Chen et al. (2015) were the first to solve this integrated problem. As they study a setting with a single picker, batches only have to be sequenced for that picker, i.e., the batches do not have to be assigned to a picker. The sequencing is important, however, as orders have due dates before which they have to be picked.

Other studies focus on problems with multiple order pickers. Ardjmand et al. (2018) study this problem in the context of a wave picking warehouse. In this setting orders do not have an individual due date, so batches need not be scheduled for a picker. Contrary to the other papers in Table 1, minimising tardiness is not relevant. Instead, the objective is to minimise the makespan.

In practice, warehouses often need to pick orders for different customers. These orders may have different due dates. Scholz, Schubert, and Wäscher (2017) study the IBRSP in this context while employing multiple order pickers. Both the batch assignment and batch sequencing problem are thus included. The objective function minimises tardiness. A similar setting is studied by van Gils et al. (2019). However, next to minimising tardiness, the minimisation of order pick time is considered as a secondary objective. Moreover, the possibility of high-level picking is introduced.

2.2. *Online and dynamic order picking problems*

If new information about customer orders arrives throughout the order picking operations, as is the case in a dynamic context such as e-commerce retail or spare parts warehouses, a static optimisation approach is unable to quickly respond to this information. Therefore, an online or dynamic order picking algorithm may lead to better results (Giannikas et al. 2017).

To account for new customer orders, the previous schedule has to be adapted. Research looking at dynamic order picking algorithms can be classified according to the decision on when to update the previous schedule. A first strategy is fixed time window batching where the optimisation algorithm is used after a fixed amount of time (Van Nieuwenhuyse and de Koster 2009). A second strategy is event driven batching, where optimisation happens after a prespecified event occurs. This event is usually seen as a certain number of new order arrivals (Rubrico et al. 2011; Li, Huang, and Dai 2017; Chen, Wei, and Wang 2018), sometimes also called variable time window batching (Xu et al. 2014; Giannikas et al. 2017). However, another type of event is the completion of an order picker's current task, for example an order picker returning to the depot after picking all items assigned to his batch (Henn 2012).

When optimisation occurs during the planning horizon, new orders may be added to the order pickers' schedules. While some studies allow the modification of batches that are currently being picked (i.e., a dynamic approach) (Chen, Wei, and Wang 2018), most studies only accept new orders in future batches (i.e., an online approach) (Rubrico et al. 2011; Li, Huang, and Dai 2017; Zhang et al. 2017). Although adapting batches that are already being picked allows for even faster customer order response times, implementing such a system requires a warehouse with continuous communication between order pickers and the warehouse management system to know exactly where each order picker is located in the warehouse. Moreover, in practice, 60% of

Table 1.: Overview of papers studying static integrated order picking problems.

	Batching	**Routing**	**Scheduling**		**Due dates**
			1 picker	>1 picker	
Chen et al. (2015)	x	x	x		x
Scholz, Schubert, and Wäscher (2017)	x	x		x	x
Ardjmand et al. (2018)	x	x		x	
van Gils et al. (2019)	x	x		x	x

the warehouses still use paper order pick lists (Giannikas et al. 2017), which makes changing pick lists during a picking tour impossible.

Table 2 gives an overview of the different order picking planning problems considered in the literature about both dynamic and online order picking. A distinction is made between simple heuristics, e.g., S-shape or largest gap order picker routing and seed or first-come-first-served batching, and advanced (meta)heuristics which truly optimise routing or batching. This distinction is important because using a simple heuristic avoids complete integration of this order picking problem with other sub-problems, and considerably reduces the complexity of the optimisation problem. Note that although tractable exact solutions for the routing problem exist under specific circumstances (Lu et al. 2016), to our knowledge they have not been applied in studies on dynamic and online order batching. Based on Table 2, there is only one paper studying the integration between order batching and picker routing (Rubrico et al. 2011). Scheduling is never integrated with batching or routing, as all studies use simple scheduling heuristics.

While a clear trend towards more integration between order picking planning problems is identified in a static context (van Gils et al. 2018b), this is clearly not the case for the online context. For example, recent research by Gil-Borrás et al. (2021) looks at batching, routing and scheduling in an online context, but not in an integrated manner. A simple S-shape heuristic is used for the picker routing, and a sequential instead of an integrated solution algorithm is proposed, as highlighted by the authors. This study moves beyond the existing research by integrating decisions on order batching, picker routing and batch scheduling for orders having individual due dates, while new orders arrive during the planning period. A solution algorithm for this problem is developed and tested on realistic instances. Furthermore, the effect of anticipating on future order arrivals is studied. Order anticipation has been studied for other combinatorial optimisation problems, e.g., inventory routing problems (Coelho, Cordeau, and Laporte 2014) and dynamic vehicle routing problems (Ferrucci, Bock, and Gendreau 2013). However, to the best of our knowledge, this is new in an order picking context.

3. Online IBRSP

In this section we introduce the online IBRSP. It is based on the operations of a global spare parts warehouse in the automotive industry and can be considered an extension of the static IBRSP introduced by van Gils et al. (2019). The integrated batching, routing and scheduling problem solves the optimisation problem of how a set of customer orders is retrieved from the warehouse as efficiently as possible, by grouping orders into batches (order batching), computing the most efficient path through the warehouse to pass the storage location of all items in a batch (picker routing) and deciding which picker is assigned which batch and in which sequence (batch scheduling) (van Gils et al. 2019).

All order pickers are assumed to be identical, with the number of order pickers based on the expected workload. The warehouse considered is a two-block warehouse, where pickers move between the storage racks to pick the items on their pick list (picker-to-parts). Orders consisting of one or more order lines, are grouped in batches, with each order being assigned to a single batch (i.e., order splitting is not allowed). Batches have a limited capacity, expressed as a number of orders, to simulate the available space in the picking device. In this picking device, orders are stored separately, so sorting is done during the pick rounds (sort-while-pick).

Table 2.: Overview of papers studying online or dynamic order picking.

	Batching		**Routing**		**Scheduling**		**Due dates**
	Simple Heur.	Adv. Heur.	Simple Heur.	Adv. Heur.	1 picker	>1 picker	
Rubrico et al. (2011)		x		x		x	
Henn (2012)		x	x		x		
Pérez-Rodríguez, Hernández-Aguirre, and Jöns (2015)		x	x		x		
Chen et al. (2016)				x		x	
Li, Huang, and Dai (2017)	x			x			
Lu et al. (2016)				x	x		
Zhang, Wang, and Huang (2016)	x		x			x	x
Zhang et al. (2017)	x		x			x	
Chen, Wei, and Wang (2018)	x		x			x	
Alipour, Mehrjedrdi, and Mostafaeipour (2020)	x		x			x	
Gil-Borrás et al. (2020)		x	x		x		
Gil-Borrás et al. (2021)		x	x			x	

After the picking operations, customer orders are shipped to their destinations. Based on a customer order's destination, it is assigned to a specific truck, with the due date of the order being equal to the departure time of its truck. A hierarchical objective function with two objectives is considered. The primary objective is retrieving all orders with minimum tardiness. The secondary objective, only applied to compare solutions with equal tardiness, is to minimise the order pick time, a measure of efficiency. The order pick time is assumed to consist of travel time, search and pick time, and batch setup time (van Gils et al. 2018a).

As truck departures are spread out during the day and customers can place urgent orders up to 30 minutes before the picking deadline of their truck, not all customer orders are known at the start of the planning period. Instead, at least some customer orders are only known during the planning period. This dynamic context requires schedules to be updated and reoptimised multiple times during the planning period as, in general, appending new orders at the end of the schedule without any optimisation will not lead to good solutions. Picker rerouting is not allowed, so only the part of the schedule (i.e., batches) currently being picked is fixed, everything else is allowed to change in every optimisation step. To avoid optimising continuously, the approach of Henn (2012) is followed: every time an order picker finishes his current batch and returns to the depot to request a new pick list, optimisation occurs and the order picker receives a new pick list. The order picker then starts with his new picking tour immediately without waiting for possible future order arrivals. In this paper, we use the term 'optimisation step' to describe this optimisation procedure every time a picker returns to the depot.

In the context of this problem, we study two settings that influence the optimisation algorithm. Both of these settings are decisions that the warehouse manager can make in order to respond to environmental factors. The first setting is based on discussions with the spare parts warehouse. Under the current operations, some urgent customer orders may arrive very late. Since the customer service level is deemed of utmost importance, the company wants to make sure to deliver every order in time, certainly these urgent orders. To maintain a high service level, the company restricts the number of trucks for which orders can be picked (only the ones with upcoming departures). By following this policy, operating efficiency may be lost, since very efficient batches may be constructed by combining orders of multiple trucks in a single batch. To assess the impact of this policy, we will study the effect of batching orders of multiple trucks.

Secondly, the possible benefits and drawbacks of anticipating on future customer orders are studied, as anticipating on future, urgent orders might help to pick these orders in time while still allowing the company to benefit from increased operating efficiency by batching over multiple trucks.

A mixed integer linear programming model for this problem can be obtained by slightly adapting the model of van Gils et al. (2019) for the static problem. Appendix A shows the required changes to the model in order to include tardiness as the primary objective and allow for an online context. As shown by van Gils et al. (2019), the model is not useful for problem instances of a realistic size if only limited computational time is available. The sole purpose of Appendix A is to clearly define the online IBRSP.

4. Algorithm description

In this section the optimisation algorithm for the online IBRSP is discussed. First, the online solution procedure is clarified in Section 4.1. Next, the large neighbourhood

search algorithm to solve the online IBRSP is explained in Section 4.2, followed by an explanation of how future order arrivals are taken into account in Section 4.3.

4.1. *Online solution procedure*

In the online IBRSP, new orders arrive over time and reoptimisation happens repeatedly. An overview of the solution procedure used in this study is given in Algorithm 1. The algorithm begins with an initialisation phase, wherein the orders to be picked are grouped into batches and assigned to the order pickers. To assess the effect of restricting the number of trucks that may be picked at the same time, called the number of active trucks, a limit is imposed on this number of active trucks. Only orders of active trucks can be batched. All order pickers are assumed to be at the depot at the start and can thus immediately start with their first picking tour. As long as there are unpicked orders left, the algorithm keeps running. Since rerouting is not allowed, batches already assigned to order pickers cannot be changed. This means that it suffices to optimise the schedule every time an order picker returns to the depot. In this optimisation step, the system checks whether new orders arrived since the last optimisation step. If new orders arrived, the orders are inserted in the existing schedule by using the 2-regret insertion, explained in Section 4.2, and the large neighbourhood search algorithm is then used to optimise the solution. The order picker who arrived at the depot is assigned the batch scheduled at his first position. This batch is then removed from the schedule, the associated orders are marked as completed and the current time of the system is updated to the next arrival time of an order picker at the depot. In case all orders currently known are picked, at most one extra truck can be activated to avoid pickers remaining idle.

Once all orders of a truck are picked and the current time passed the last allowed arrival time for orders in that truck, the active trucks are updated and orders of a new truck can be picked. The pool of pickable orders at the next optimisation step is then updated, and this process continues until all trucks have been completed.

Algorithm 1: Online solution procedure

```
Initialise algorithm;
while Not all trucks completed do
    Every time a picker returns to depot do
        Check for new orders;
        if new orders then
            Insert orders in previous solution;
            Run LNS on solution;
        Assign batch to picker;
        Remove batch from solution;
        Update time to first completion of a pick round or first order arrival if
          no orders left;
    while no active truck completely picked;
    Update active trucks;
```

4.2. *Online large neighbourhood search*

Solving the online IBRSP to optimality seems not possible within reasonable computational time, as the problem is NP-hard. Therefore, a metaheuristic is developed to solve this problem. The metaheuristic used in this study is a large neighbourhood search (LNS), first proposed by Shaw (1998). Although LNS was originally developed for the vehicle routing problem, it has been applied in the area of order picking before in the form of adaptive large neighbourhood search (Žulj, Kramer, and Schneider 2018; Kuhn, Schubert, and Holzapfel 2020).

An LNS operates by continuous relaxation (destroy) and reoptimisation (repair) (Shaw 1998). Part of the solution is relaxed and then reoptimised by use of different heuristics. In order to find good solutions, the heuristics within the LNS framework should help in diversifying as well as intensifying the search (Pisinger and Ropke 2007). In each iteration of the algorithm, a single destroy and a single repair operator are randomly selected and applied.

A series of destroy operators was implemented and tested. Operators that are part of the final algorithm are discussed next, while a full list of tested operators is given in Appendix B. The operators can be divided into two categories: those working on the order level, and those working on the batch level. The used operators are the following:

- Random order removal (order level): randomly select orders and remove them from the solution.
- Smallest distance savings (batch level): for every batch, calculate the difference between the total picker travel distance to pick every order line individually and the travel distance of the batch. Destroy the batches with the smallest difference in travel distance.
- Largest number of additional aisles (batch level): for every batch, find the difference between the number of unique subaisles to visit when picking this batch and the number of unique subaisles to visit for this batch's order with the most unique subaisles. Destroy the batches with the largest difference in subaisles.
- Largest additional covering area (batch level): for every batch, calculate the difference between the rectangular covering area of the whole batch and the covering area of this batch's order with the largest covering area. Destroy the batches with the largest difference in covering area.

The random order removal operator diversifies the search, while the other operators intensify the search by removing a part of the solution that looks inefficient. Note that the intensification operators are deterministic: it is thus possible to get stuck in a local optimum. To avoid deterministic operators trying the same destroy operations multiple times, a random component can be introduced (Pisinger and Ropke 2007). Therefore, the deterministic operators are used for half of the wanted destruction, followed by random order removals for the other half.

Once part of the solution is destroyed by the selected destroy operator, the removed orders are reinserted in the partial solution. The repair operators used are greedy insertion as well as k-regret insertion. Greedy insertion selects the removed orders one by one ordered by increasing due dates, and inserts them in the cheapest position, where cheap is defined as causing the lowest increase in tardiness or, in case the order can be inserted without a tardiness increase, where the increase in order pick time is the lowest. For the k-regret insertion, all orders are tested in every possible position and the order with the largest difference in insertion cost between its best and k-best position is selected to be inserted in its best position. Insertion costs of the remaining

orders are then updated (note that only the insertion costs for the last changed picker have to be updated, saving a lot of computational time). Then, the next order to be inserted is selected in the same way. This continues until a new feasible solution is constructed. Costs are once again defined as the increase in tardiness and, at the second level, the increase in order pick time. Note that greedy insertion is not the same as 1-regret: the former takes a single order and inserts it in the cheapest position, while the latter would compute insertion costs for all orders and then select the cheapest position over all orders. This implies a computational complexity of $O(n)$ for greedy insertion, and $O(n^2)$ for k-regret, with n the number of orders.

Calculating the insertion and removal costs for every order is very computationally expensive, because the routing of the associated batch has to be updated. van Gils et al. (2019) found that enumerating all possibilities is preferable when at most eight order lines (i.e., locations) have to be picked in one batch. If more order lines need to be sequenced, the routing is optimised using the Lin-Kernighan-Helsgaun (LKH) heuristic (Helsgaun 2000). Even when using this heuristic, computational times quickly become an issue. Therefore, during the LNS a simple heuristic, greedy insertion, is used to calculate the distance of a batch. When removing an order from a batch, all its order lines are simply removed, with the remaining locations still being visited in their previous sequence. When inserting an order in a batch, all order lines of this order are selected one by one in a random sequence, and are inserted at the position with the smallest increase in travelling distance. Although this way of computing order insertion and removal costs may not lead to the best routing sequence, it allows very fast computational times. This simple heuristic is used in every iteration of the LNS.

The full LNS algorithm is shown in Algorithm 2. The initial solution is constructed from the final solution of the previous optimisation step by removing the picked batches and inserting any new orders with the 2-regret insertion operator. If no previous optimisation step exists at the start of the planning period, the initial solution is constructed with earliest due date batching. This initial solution is then optimised with a local search, which proved very useful in optimising a problem in this setting (van Gils et al. 2019). The local search of van Gils et al. (2019) is used and consists of four operators: order shift, order swap, batch shift and batch swap. The resulting solution is then used as input for the LNS. During the local search, the routing of a batch is optimised by the LKH-heuristic.

The LNS itself consists of repeatedly destroying and repairing, which is done until the maximum number of iterations is reached. Although the LNS alone shows very good performance on order pick time, preliminary experiments showed that it is less efficient in reducing tardiness. A combination of local search and LNS is therefore used: on every new solution for the LNS, a local search is performed. Since the local search is computationally expensive compared to the LNS, it is only used for a limited number of iterations and if no solution without tardiness was found yet. The local search is then used once more on the final solution from the LNS.

In every iteration of the LNS, a decision regarding the acceptance of the new solution is made. Recall that the objective function is hierarchical, with tardiness as the primary, and order pick time as the secondary objective. Based on whether the local search is still active, as discussed in the previous paragraph, two phases of the algorithm can be identified: LNS with local search, or LNS alone (both phases are mentioned in the comments in Algorithm 2). These phases influence the decision to accept a new solution. In the first phase, when the local search is still being used, a best, second best and four random solutions are used, with the parameter setting of van Gils et al. (2019). In the second phase, where pure LNS without local search

Algorithm 2: LNS algorithm

Create BestSolution S^*, SecondBestSolution S^{**}, PreviousSolution S^0, CurrentSolution S^1, RandomSolutions $S^{r1}, S^{r2}, S^{r3}, S^{r4}$;
Create initial solution S^i;
Local search on S^i;
S^*, S^{**}, S^0, S^1, $S^{r1}, S^{r2}, S^{r3}, S^{r4} \leftarrow S^i$;
while *NoIterations* $\leq$ *MaxIterations* **do**
    Select destroy operator randomly;
    Destroy S^1 with selected destroy operator;
    Select repair operator randomly;
    Repair S^1 with selected repair operator;
    if $f^t(S^*) > 0$ **and** *NoIterations* $\leq$ *MaxLocalSearchIterations* **then**
     `// Phase 1 of the algorithm`
        Local search on S^1;
        if $f^t(S^1) < f^t(S^*)$ **or** $(f^t(S^1) = f^t(S^*)$ **and** $f^p(S^1) < f^p(S^*))$ **then**
            $S^{**} = S^*$;
            $S^* = S^1$;
            $S^0 = S^1$;
        end
        else if $f^t(S^1) < f^t(S^{**})$ **or** $(f^t(S^1) = f^t(S^{**})$ **and** $f^p(S^1) < f^p(S^{**}))$ **then**
            $S^{**} = S^1$;
            $S^0 = S^1$;
        end
        else
            Save S^1 as one of four random solutions;
            Randomly select solution as S^0 and S^1;
        end
    end
    else `// Phase 2 of the algorithm`
        if $(f^t(S^1) < f^t(S^*))$ **or** $(f^t(S^1) = f^t(S^*)$ **and** $f^p(S^1) < f^p(S^*))$ **then**
            $S^* = S^1$;
            $S^0 = S^1$;
        end
        else if $(f^t(S^1) < f^t(S^0))$ **or** $(f^t(S^1) = f^t(S^0)$ **and** $f^p(S^1) < f^p(S^0))$ **then**
            $S^0 = S^1$;
        end
        else if $(f^t(S^1) = f^t(S^0))$ **and** $(f^p(S^1) > f^p(S^0))$ **then**
            if $RandNum < e^{((f^p(S^0) - f^p(S^1))/Temperature)}$ **then**
                $S^0 = S^1$;
            end
        end
    end
    NoIterations = NoIterations + 1;
    Update Temperature;
end
Local search on S^*;

is operating, simulated annealing is used as acceptance criterion. The simulated annealing only works on the order pick time and never accepts solutions with worse tardiness. A new best solution is always accepted, while a worse solution can also be accepted with a diminishing probability. A slowly decreasing temperature throughout the LNS iterations will reduce the probability of accepting a worse solution, to move from diversifying to intensifying the search.

4.3. *Anticipating order arrivals*

In the online setting, new orders arrive during the planning period. It is not known if, when or how many new customer orders will arrive for every truck. Nevertheless, it may be important to anticipate on future order arrivals from the start of the planning period, to make sure sufficient order picking capacity is available for these dynamic orders. Although exact information about future orders is not available, companies are able to make very accurate forecasts based on previous order arrival data (Leung et al. 2020; van Gils et al. 2017).

Under the assumption that previous ordering data is available, managers can estimate an average number of orders per truck as well as the variability on this number of orders. This information can then be included during the optimisation. In order to reserve sufficient time for future orders, we propose to include dummy orders in the schedule. Dummy orders are given a due date like normal orders, depending on the truck to which they belong. Every dummy order is inserted in a separate batch and cannot be batched with normal or other dummy orders. Dummy orders occupy space in the schedule, with an estimated pick time equal to the average pick time for an order in the warehouse.

The number of dummy orders for a certain truck will change throughout the planning horizon, based on two forecasts of the number of orders to expect for this truck. A first forecast, F_0, is constant and based on the average expected number of orders, as given by the warehouse manager's data. A second forecast, F_1, is updated in every optimisation step, based on the current order arrival speed and the expected demand pattern.

Orders are assumed to arrive between an earliest and latest arrival time, denoted by t_e and t_l respectively. Suppose that n_t is the number of orders that arrived up to the current system time t, for a specific truck with due date d. Then, if the order arrival distribution is known, the cumulative density function can be used to estimate the fraction of orders that has arrived at time t. This fraction is denoted by c. Using these parameters, F_1 is computed by equation (1) if $t > t_e$ (and thus, $c > 0$), otherwise F_1 is equal to zero.

$$F_1 = \frac{n_t}{c} \tag{1}$$

Next, F_0 and F_1 are used for the combined forecast F. When $t = t_e$, F is based completely on F_0. For $t = t_l$ only F_1 is relevant. If $t_e < t < t_l$, F is the weighted average of F_0 and F_1, where the weights are based linearly on the proportion of the arrival interval that passed, following equation (2). This equation shows that the weight of F_0 decreases, while the weight of F_1 increases with t, since the accuracy of F_1 is

assumed to increase over time.

$$F = \frac{t_l - t}{t_l - t_e} \times F_0 + \frac{t - t_e}{t_l - t_e} \times F_1 \tag{2}$$

The number of dummy orders to insert in the schedule, n_f, is then given by subtracting the orders already known from the expected number of orders, as shown in equation (3). Note that a forecast should be made for every active truck, to estimate the number of dummy orders that should be inserted with this truck's due date.

$$n_f = max(0, F - n_t) \tag{3}$$

5. Experimental study and insights

In this section, results and insights of a large-scale numerical study are presented. The algorithms are implemented in C++. All experiments are performed on an Intel Xeon Processor Gold 6140 at 2.3 gigahertz. First, in Section 5.1, the performance of the LNS algorithm is tested in a static problem setting and compared to the state-of-the-art solution algorithm from the literature. In Section 5.2 the design of the online problem instances is discussed, followed by parameter tuning in Section 5.3. These parameters are then used in Section 5.4, where the algorithm is tested on the online problem instances and insights are discussed.

5.1. *Static benchmark*

In order to test the performance of the new LNS heuristic, the problem instances of van Gils et al. (2019) are used, as well as the results of their iterated local search (ILS) algorithm. We had access to the programming code of their optimisation algorithm, which made a thorough comparison possible. Note that the algorithm of van Gils et al. (2019) was developed for the static IBRSP and does not take release dates into account. Nevertheless, for the online algorithm, we need instances with order release dates larger than zero to obtain insights into the online problem. Therefore, different instances are used for the algorithm comparison in this section, compared to the instances in the following sections.

To make the calculation of distances between the warehouse locations consistent between both studies, a small adaptation of their distance calculations is necessary. Therefore, the results reported in van Gils et al. (2019) cannot immediately be used. All large problem instances are solved again with all settings identical to those reported in their paper, except for the correction of this distance calculation, to allow for a fair comparison between both algorithms. Detailed results of these solutions are available at `doi.org/10.17605/osf.io/7r4gn`.

The same instances, with the same number of order pickers, are solved by the LNS algorithm. After some preliminary testing on some test instances, the following algorithm setup is applied. The LNS uses 10,000 iterations on every instance. If the best solution found so far contains tardiness, the combination of local search and LNS is used during the first 5000 iterations (phase 1 of the algorithm). After 5000 iterations or once there is no longer tardiness in the best solution, the remaining iterations are performed with only the LNS operators (phase 2). In every LNS iteration, 10% of the solution is destroyed. During phase 1, the destroy operators smallest distance savings

Table 3.: Comparison of ILS and LNS on the static IBRSP.

	ILS	LNS	Difference
Tardiness instances (#)	20	18	-10%
Total tardiness (s)	21007	18230	-13.22%
Average order pick time (s)	41508	41295	-0.52%
Average CPU time (s)	127.85	64.08	-49.88%

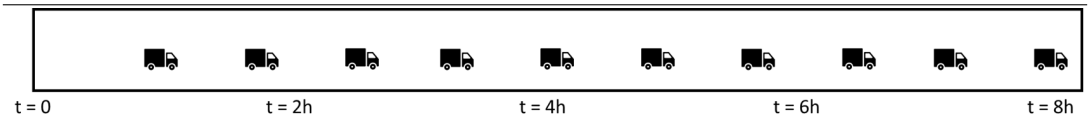
(batch), largest number of additional aisles (batch) and largest additional covering area (batch) are used. In phase 2, only random order removal is used. To repair the solution in both phase 1 and phase 2, greedy insertion as well as 2, 3, 4 and 5-regret are used.

The results of both algorithms are reported in Table 3. Recall that the primary objective is minimising tardiness, followed by minimising the order pick time. Each of the 7290 instances of van Gils et al. (2019) is solved by a single run of each algorithm. Twenty instances have some tardiness remaining in the final solution after using the ILS algorithm. When using the LNS, eighteen instances have tardiness remaining after 10,000 iterations (all of which contain tardiness with the ILS as well). The sum of the tardiness over all instances is reduced by over 13%. The average order pick time is also reduced by 0.52% when using the LNS. The largest improvement, however, is obtained regarding the computational time. It is almost cut in half when using the LNS compared to the ILS. Based on these results, the LNS seems very well suited to tackle the IBRSP.

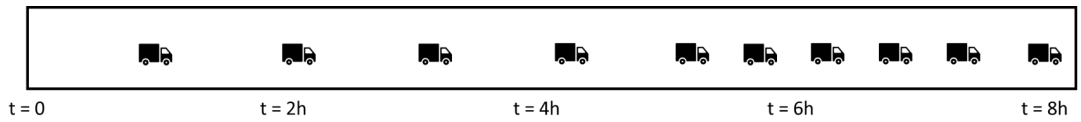
5.2. *Instances*

To test the developed metaheuristic algorithm, new problem instances for the online problem are constructed. These instances are made publicly available at `doi.org/10.17605/osf.io/7r4gn`. General parameters of the warehouse are given in Table 4 and are based on van Gils et al. (2019) and real-life data. For the across-aisle storage policy, three classes are used. Class A, B and C contain 1/6, 1/3 and 1/2 of the SKUs, while accounting for 60%, 30% and 10% of the picks, respectively. The number of order lines per order is randomly generated following an exponential distribution with mean 3.5. As the resulting number is rounded up, the average number of order lines is approximately 4.

Next to these general warehouse parameters, an experimental design is used to generate the problem instances. This design consists of five factors, each having multiple factor levels, as shown in Table 5. A planning period of eight hours is considered. The first factor, the average number of orders (λ), is either 600 or 1000. These orders belong to several trucks, with the number of trucks being the second factor. The expected number of orders per truck (μ) is equal to the average number of orders divided by the number of trucks available (Ω), calculated by equation (4). Therefore, μ is constant within an instance. However, to mimic a real-life context, randomness is involved in the order generation. The level of randomness is the third factor in the experimental design and is expressed as a percentage of variation. This percentage (p) is used to decide on the number of orders in an instance: for each truck, the number of orders is



(a) Uniform truck distribution for ten trucks.



(b) Peak truck distribution for ten trucks.

Figure 1.: Due time distribution for an instance with ten trucks.

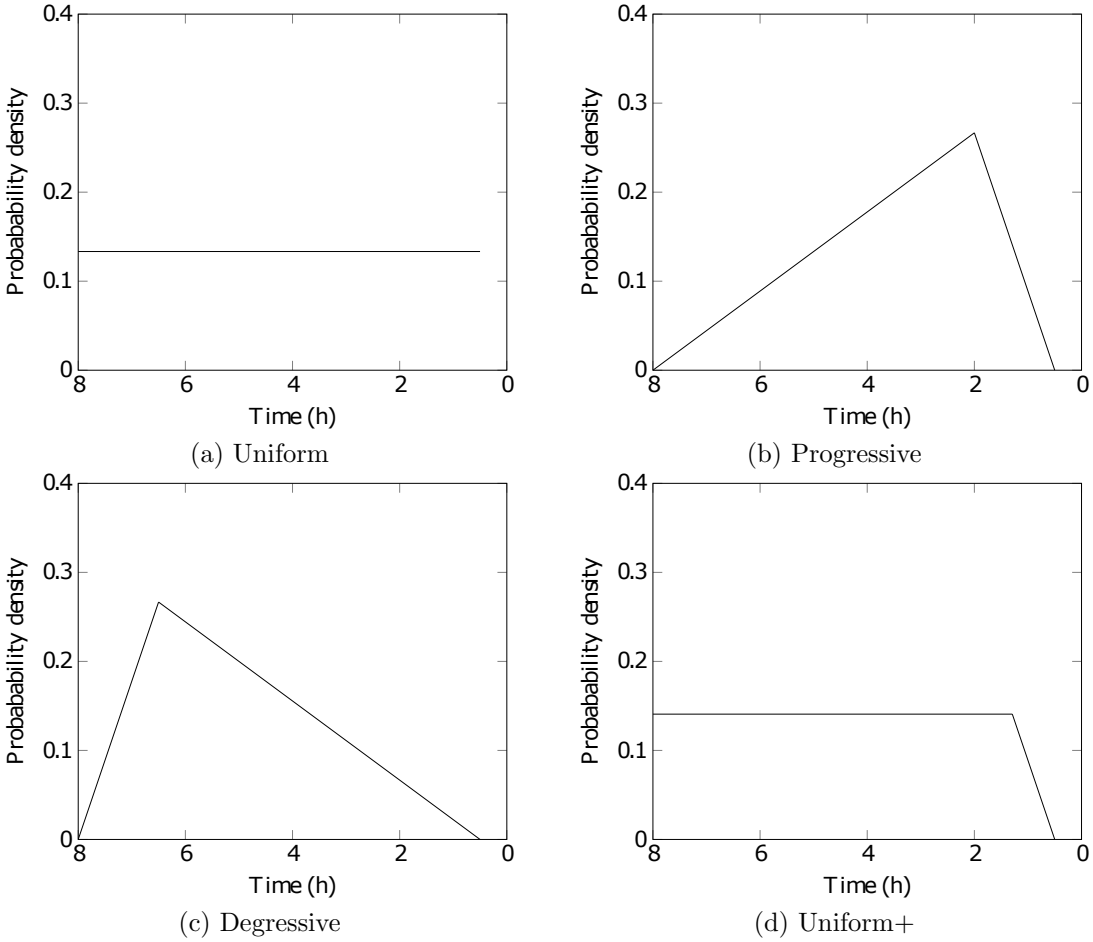
randomly drawn from a triangular distribution with μ, $(1-p)\times\mu$ and $(1+p)\times\mu$ as mode, minimum and maximum, respectively.

$$\mu = \frac{\lambda}{\Omega} \tag{4}$$

As a result, the total number of orders is not exactly equal to the average of 600 or 1000 orders. The number of order pickers available in an instance is based on the expected number of orders. For every 200 orders, one order picker is scheduled. For instances with an average of 600 and 1000 orders, this means three and five order pickers are available, respectively.

The fourth factor is the truck distribution, to gauge the effect of different truck departure patterns. Every truck departs at a different time in the planning horizon. In every instance, the first truck departs after one hour, while the last truck departs after eight hours, i.e., the end of the planning period. The other trucks are scheduled in between, with either a uniform or a peak distribution. Under the uniform distribution, the remaining trucks are distributed evenly between the first and last truck. In the peak distribution, the planning period is divided into a busy and calm part. The busy part lasts 160 minutes (i.e., one third of the planning period) and starts after 280 minutes. Half of the trucks depart during the peak period. In both the busy and the calm part, the trucks are evenly distributed, resulting in double the time between trucks during the calm period compared to the busy period. The difference between both distributions is visualised in Figure 1.

Finally, the fifth factor is the order arrival pattern, looking at the effect of different order arrival patterns. The order arrival pattern can be uniform, progressive or degressive. Under every pattern, orders arrive at most eight hours and at least half an hour before their due time (the departure of the truck the order belongs to). The different patterns are shown in Figures 2a - 2c (2d is referred to in Section 5.4.1). Under the uniform pattern, an order is equally likely to arrive at every point in time. With the progressive pattern, order arrivals are skewed towards their due times, leaving less time for picking. Under the degressive pattern a rather early arrival is more likely. Arrival times are drawn randomly from a triangular distribution in both the progressive and degressive case. For the progressive pattern, arrival times are drawn from a distribution with 0.5, 2, 8 hours before the due time as minimum, mode and maximum respectively. For the degressive pattern, these numbers are 0.5, 6.5 and 8, respectively.



(a) Uniform (b) Progressive

(c) Degressive (d) Uniform+

Figure 2.: Probability density function of the order arrival patterns. The horizontal axis shows the time before the truck departure, with 0 equal to the truck departure, 8 and 0.5 the earliest and latest possible arrival time of an order, respectively.

Table 4.: Warehouse parameter values.

Warehouse parameters	Parameter value
Number of warehouse blocks	2
Number of levels	1
Number of aisles	12
Number of sub-aisles	24
Locations per aisle	240
Storage policy	Across-aisle
Storage location length	1.3 meters
Storage location width	0.9 meters
Pick aisle width	3.0 meters
Cross-aisle width	6.0 meters
Picker travel velocity	1 meter per second
Setup time	180 seconds
Search and pick time	10 seconds
Batch capacity in number of orders	10
Avg. number of order lines per order	4
Number of order pickers	1 per 200 orders
Duration planning period	8 hours

Combined, these factors lead to a factorial design with $2 \times 3 \times 2 \times 3 \times 2$, or 72 factor combinations. For each factor combination ten instances are generated. Every instance will be solved six times: for 20%, 40% and 60% of the trucks activated, to assess the impact of batching orders over a larger number of trucks as discussed in Section 3, and both with and without order anticipation, to test whether anticipating on future orders is beneficial.

5.3. *Parameter tuning*

Because the online problem instances are quite different from the static benchmark in Section 5.1, a few parameter modifications to the algorithm are necessary. For example, the maximum number of orders in an optimisation step increases from at

Table 5.: Experimental design online problem.

Factor	Factor levels		
Average number of orders (λ)	600	1000	
Number of trucks (Ω)	10	15	20
Truck distribution	Uniform	Peak	
Order arrival distribution	Uniform	Progressive	Degressive
Variation (p)	10%	30%	

Table 6.: Factors and factor levels in parameter tuning.

Factor	Factor levels			
LNS iterations	500	1000		
Destroy percentage	5	10	15	
Initial temperature percentage	5	10	15	20

Table 7.: Results of parameter tuning with the selected factor combination highlighted in bold.

Destroy %	LNS it.	Tardiness (s)	OPT (s)	Longest opt. step (s)	
				Average	P95
5	500	5184	110,024	26	90
	1000	5813	109,992	31	95
10	**500**	**4935**	**109,785**	**38**	**124**
	1000	4598	109,679	54	176
15	500	4272	109,727	61	193
	1000	3959	109,545	99	306

most 300 under the static problem instances to more than 500 in some of the online problem instances. This increase has a large effect on the computational times of the LNS, mostly on the local search iterations. In order to keep the computational times under control, at most 5000 LNS iterations, of which 100 local search iterations, are used in the initialisation phase of the online algorithm. Note that the local search was only used if the best solution contained tardiness during the static problems, while it is always performed for 100 iterations in the online problem to allow for a fair comparison and better interpretation of the results.

Afterwards, every time an order picker returns to the depot, a restricted version of the LNS is used. Since a good solution is already available from a previous optimisation step, less iterations are needed to obtain a new good solution. A shorter computational time is also required to not postpone the departure of the order picker. In the reduced LNS algorithm, 10 local search iterations are used in every optimisation step. In the online optimisation, the same selection of operators is used as under the static problem.

To optimise the algorithm for the online IBRSP, three parameters are tuned on 30 online test instances, solved for every parameter combination and with 20%, 40% and 60% of the trucks activated. The parameters to be tuned are the number of LNS iterations every time an order picker returns to the depot, the percentage of destruction in every LNS iteration and the percentage of the initial order pick time to use as the initial temperature during the simulated annealing (this temperature is afterwards reduced to 90% of the previous temperature after every 1% of the total iterations of the LNS). The parameters and values tested in the parameter tuning are shown in Table 6. Note that simulated annealing is only used once the local search iterations are completed and only works on the order pick time. Solutions with a higher tardiness are never accepted.

Table 8.: Results of tuning the initial temperature percentage, with the selected setting highlighted in bold.

Initial temperature percentage	OPT (s)
5%	109,771
10%	**109,751**
15%	109,804
20%	109,842

Table 7 shows the results of the parameter tuning for the destroy percentage and number of LNS iterations (the initial temperature is discussed later, as it only influences the order pick time). Destroying a larger part of the solution clearly results in a lower tardiness and order pick time per order (we look at tardiness and order pick time per order, to make a fair comparison between instances with a different number of orders). Increasing the number of LNS iterations has a variable effect on tardiness, but always decreases the order pick time. This is as expected as the LNS operators are very effective in reducing order pick time, whereas the local search is more efficient in reducing tardiness. Although a large destroy percentage combined with many LNS iterations is more effective in reducing tardiness and order pick time, the computational times quickly become prohibitively large for an online setting. The fifth column shows the average duration of the longest optimisation step to solve an instance, where the first optimisation step is not considered because more time to optimise may be used before the start of the shift. During the shift, optimisation starts from a good solution, which allows using less computational time in every optimisation step. An observation in the spare parts warehouse showed a batch setup time of approximately 120 seconds, in line with batch setup times of 180 seconds in the literature (van Gils et al. 2019). Because a few outliers with very high computational times were observed, the 95th percentile of the longest optimisation step is also reported. To obtain acceptable results while making sure not to exceed a computational time of 180 seconds, the parameters are set at 10% destruction with 500 LNS iterations.

The simulated annealing only works on order pick time and does not have a markable impact on tardiness or computational time. The effect of the initial temperature on the order pick time is shown in Table 8. Although the differences between the parameter values are small, a value of 10% is selected.

Finally, the stability of the selected algorithm was tested (i.e., does it provide similar results over multiple runs?). Eight online problem instances were randomly selected. On these instances, every optimisation step (i.e., each time a picker returns to the depot) is solved 30 times by the algorithm. Each time, the solution of the first algorithm run is implemented to generate the input for the next optimisation steps, thereby ensuring that all 30 runs have the same input at every optimisation step. This process is repeated until the end of the instance. The results of this test run are shown in Table 9. In the table, it can be seen that including all optimisation steps in all instances results in a very low absolute standard deviation regarding tardiness (9.31 seconds), but compared to the average tardiness this value is still high (10.33%). However, the results are skewed because of some optimisation steps with very low tardiness. In these steps, most algorithm runs resulted in a solution without tardiness, but a few

Table 9.: Results of stability tests on the selected algorithm.

Optimisation steps	Tardiness		Order pick time	
	Avg STD (s)	Avg STD (%)	Avg STD (s)	Avg STD (%)
All steps	9.31	10.33	96.23	0.72
$0 <$ tardiness < 90s	5.93	471.42		
Tardiness > 90s	145.41	1.91		

steps had a small amount of tardiness (always less than 90 seconds). This results in a very low average tardiness (sometimes close to zero), which makes the average standard deviation in percentages very large. When looking at optimisation steps with considerable tardiness in all algorithm runs (always more than 90 seconds of tardiness in every run), the average standard deviation is 145.41 seconds, which is equal to 1.91% of the average tardiness in that optimisation step. When looking at the order pick time over all optimisation steps and all instances, a distinction in separate cases is not required since there is always a sufficiently large order pick time. The results indicate an average standard deviation of only 96.23 seconds, which is 0.72% of the average. Based on these results, we believe that the algorithm is sufficiently robust to make meaningful conclusions.

5.4. *Experimental design*

To assess the performance of the proposed algorithm, ten instances per factor combination of the factorial design are solved. For each of these instances, the LNS algorithm is run once for 20%, 40% and 60% of the trucks active, both with and without order anticipation. Detailed results on an instance level are available at `doi.org/10.17605/osf.io/7r4gn`. Appendix C reports the results of a mixed ANOVA regarding tardiness and order pick time. The main insights are discussed in the next sections, where the 5% significance level is used to test statistical significance. To obtain reliable conclusions from the ANOVA, three assumptions should be satisfied: normality, homogeneity of variance and sphericity of the covariance matrix. Normality is ensured by using a balanced experimental design, which makes the F-statistic quite robust to normality violations. Although Levene's test for homogeneity of variance is violated, this is not a problem due to the large sample size in every group, and the equal sample sizes between all groups. Mauchly's test for sphericity is also violated, so the conservative Greenhouse-Geisser corrections are used to correct the inflated F-test type I error rate (Field 2013).

A mixed ANOVA is used to analyse the results since the data contain both between groups variables (the factors of the experimental design) and repeated measures variables (percentage of active trucks and with or without order anticipation) (Field 2013). The between groups variables do not directly influence the optimisation algorithm itself, but influence the construction of the instances. These different instance characteristics allow to make some interesting conclusions, discussed in Section 5.4.1. Note that in practice these instance characteristics are mostly fixed by the environment in which the company operates. The repeated measures variables, on the other hand, are set by decisions made by the warehouse manager and directly influence the

optimisation algorithm. These variables lead to interesting insights into the effect of two operating practices. In Section 5.4.2 the effect of the percentage of active trucks is discussed, followed by a discussion of order anticipation in Section 5.4.3.

5.4.1. Discussion instance characteristics

When looking at the results of the different factors of the factorial design, shown in Figure 3, the following conclusions can be made regarding tardiness and order pick time per order. Increasing the average number of orders from 600 to 1000 results in a statistically significant decrease in both tardiness and order pick time. This can be explained by the increased possibility of having similar orders to be batched in an optimisation step, which allows making more efficient batches.

Spreading the orders over more trucks shows a mixed effect, albeit not statistically significant. For the different truck distributions, an interesting divergence between tardiness and order pick time can be observed, with both effects being statistically significant. Under a uniform truck distribution, tardiness is higher while order pick time is lower, compared with the peak truck distribution. The higher tardiness for a uniform distribution may seem counter-intuitive. However, since the peak period is situated towards the end of the planning period, under the uniform distribution a larger part of the orders has to be picked at every point during the planning period. This makes the planning problem harder to solve, as there is less time at the beginning of the planning period to pick some orders very efficiently in advance. The increased order pick time for the peak distribution may be caused by less flexibility in batching orders, because of the increased workload during the peak period.

The order arrival pattern impacts tardiness and order pick time in a statistically significant way. Under the degressive case, orders arrive rather early during the planning period and batching the orders efficiently is thus easier because more orders are available from the start. In the progressive case, orders arrive rather late, which results in higher tardiness and order pick time. The uniform pattern leads to the highest tardiness and order pick time, which may sound counter-intuitive because it seems to fit between the progressive and degressive case. However, this is not completely correct. Under the uniform pattern, more orders are arriving very late and close to the due date of their truck. As the progressive pattern is modelled with a triangular distribution, more orders arrive close to their due date under the uniform pattern than under the progressive pattern. To mitigate this issue, a new release time pattern is created, called uniform+, shown in Figure 2d. Under this pattern, order arrivals are uniformly distributed over the first part of the planning period, up to the intersection of the uniform and the progressive pattern. After this intersection point, order arrival times follow the progressive pattern. This way, there is no large group of orders arriving at the end of the planning period. Figure 3 shows that the tardiness reduces considerably by implementing this feature, although it is still approximately equal to the progressive pattern. Note that only the difference in tardiness between the uniform and degressive pattern is statistically significant. Because of high standard deviations regarding the tardiness, proving statistical significance is difficult. Nevertheless, these results may indicate that the late arrival of some orders is indeed very complex, and some more margin between the order arrivals and order due dates may be required. However, a trade-off can be identified between offering a higher customer service level by allowing late order arrivals, and offering a higher customer service level by fulfilling every order in time.

Finally, regarding the factor variation, it is clear that a higher variation in the

Table 10.: Computational times for the longest optimisation step and the average optimisation step (excluding initialisation), as well as the initialisation, for both the standard setting and order anticipation.

Percentile	Longest opt. step (s)		Average opt. step (s)		Initialisation (s)	
	Standard	Anticipate	Standard	Anticipate	Standard	Anticipate
Average	35.68	39.85	15.18	16.59	329.22	326.54
90%	82.21	92.80	35.38	82.21	739.50	738.48
95%	105.43	127.47	47.27	105.43	955.09	981.20
99%	161.38	226.16	66.59	161.38	1258.65	1275.06

number of orders per truck leads to a statistically significant increase in tardiness and order pick time. This can be explained as follows. With a higher variation a truck may contain much more or much less orders than the average, which increases or decreases the workload for that individual truck, and sometimes for the instance as well if other trucks do not balance this workload. Firstly, although having less orders is never an issue, the instances with an increase in the number of orders are characterised by an increased size of the planning problem and thus the solution space, which makes the optimisation problem harder to solve. Secondly, a very high workload for even a single truck may lead to issues in picking everything for that truck in time. Thirdly, as the number of order pickers is based on the average number of orders per instance, a much larger number of orders increases the workload of order pickers significantly. Therefore, companies that experience high variation in order levels and do not want to compromise their high service level, are required to employ enough order pickers to make sure fluctuations can be handled adequately.

Next to tardiness and order pick time, the solution time is also important for an online optimisation problem. We aimed for a solution time less than the setup time of three minutes in every optimisation step (except for the initialisation). Looking at Table 10, it is clear that the computational times are acceptable, with even the 99% percentile being below three minutes, except for the case with order anticipation. There are only few outlier optimisation steps with a higher computational time, so in practice a time limit can be applied to the algorithm to avoid excessive run times. These computational times indicate that the LNS algorithm is reasonably fast and usable in a real-life setting.

5.4.2. Discussion percentage active trucks

The spare parts warehouse under study currently restricts the number of trucks that can be picked at the same time. To test the effect of relaxing this restriction, every instance is solved with 20%, 40% and 60% of the trucks activated. The percentage of active trucks has a statistically significant effect on both tardiness and order pick time (shown by the ANOVA in Appendix C). Figure 4 shows that increasing the percentage of active trucks from 20% to 40% reduces the tardiness and order pick time per order considerably (indicated by the black bars and line, respectively). When going from 40% to 60%, however, tardiness increases slightly while the order pick time can still be reduced. This result is not as expected, as batching orders of a larger number of trucks leads to more orders in every optimisation step, offering more

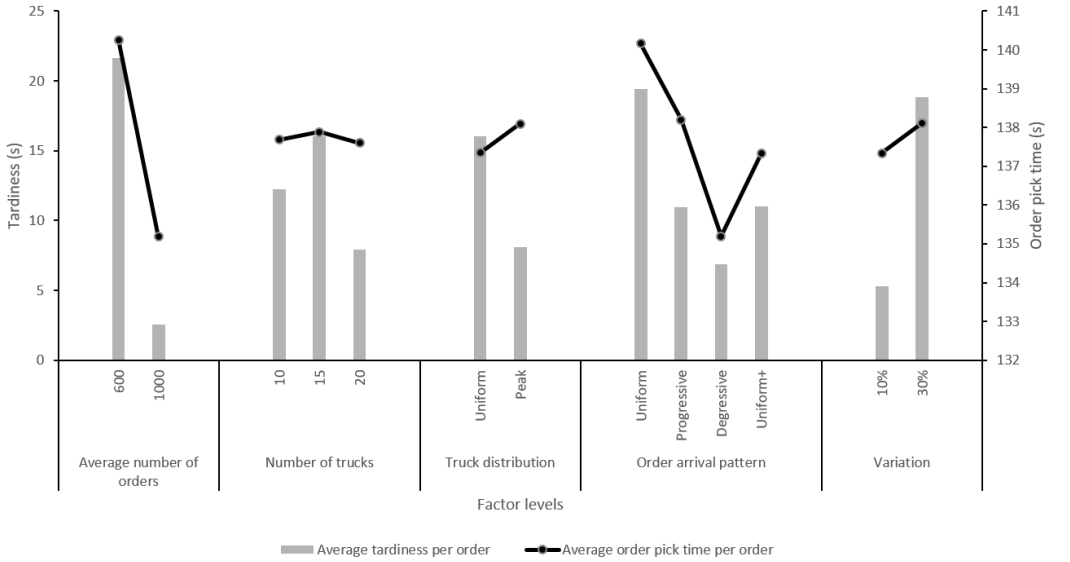


Figure 3.: Tardiness and order pick time for the online LNS algorithm, averaged over all factor combinations, with 20%, 40% and 60% of the trucks active, and with and without order anticipation.

batching opportunities and thus allowing for more efficient batches. Indeed, this seems to happen since the order pick time still decreases when going from 40% to 60%. Nevertheless, the tardiness does increase a bit. For practitioners, these results are very interesting. Batching over too few trucks clearly leads to a bad customer service level and inefficient operations. Yet, taking more trucks into account has quickly diminishing returns, as the complexity of the problem increases, leading to longer computational times, with only small benefits regarding operational efficiency, and even a decrease in customer service level.

Still, this counter-intuitive result is remarkable. Batching orders with due dates further into the future should never lead to worse solutions, as a solution for 60% of the trucks can be identical to the case with 40% with the additional orders appended behind all previous batches. The increasing tardiness may be caused by the following reason. At the start of the planning period, very efficient batches with orders from multiple due dates are made. Some orders with early due dates can still be postponed without any tardiness in the solution. While these batches with orders with a later due date are being picked, new orders of the early due dates may arrive. Suddenly, the combination of postponed and new orders may prohibit picking everything in time. Therefore, anticipating on these future order arrivals may be required to avoid the increase in tardiness.

5.4.3. Discussion order anticipation

The same instances were solved while the algorithm anticipates on future order arrivals by implementing dummy batches for every expected order, as introduced in Section 4.3. The results are visible in Figure 4, shown by the grey bars and line. Anticipating on future order arrivals reduces tardiness in a statistically significant way, with the effect becoming larger for a higher percentage of active trucks (shown by the ANOVA in Appendix C). The unexpected effect of an increasing tardiness when going from

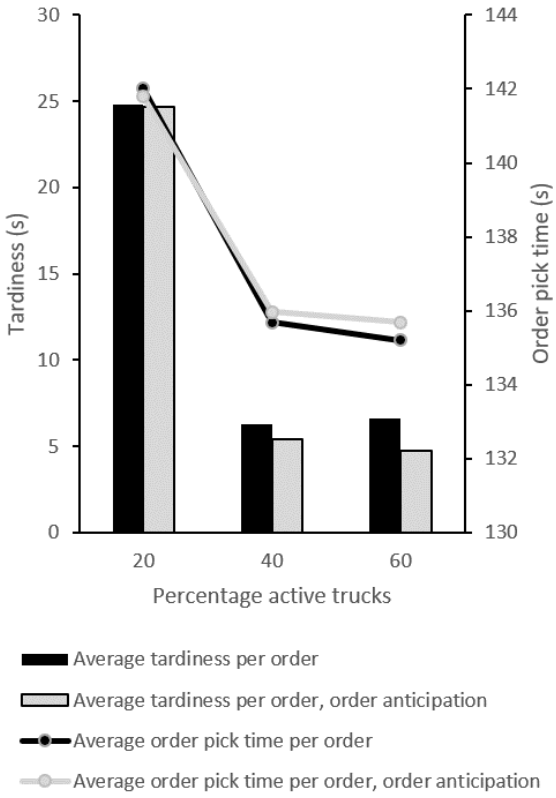


Figure 4.: Effect of order anticipation on tardiness and order pick time.

40% to 60% active trucks is also removed. Note that order anticipation was expected to be relevant mostly for a higher percentage of active trucks. Nevertheless, contrary to the effect of order anticipation alone, the interaction between the percentage of active trucks and order anticipation is not statistically significant, possibly caused by the high variance regarding tardiness. Although the service level clearly improves, the order pick time is statistically significantly higher when anticipating on future order arrivals. This indicates that companies should make a trade-off between customer service level and operational efficiency.

Next to the increasing order pick time, using dummy orders has another disadvantage. Since the problem size gets larger by including dummy orders, there is a non-negligible increase in computational times, as shown in Table 10.

6. Managerial insights and conclusions

Warehouses should handle a large number of customer orders as efficiently as possible. The interaction between order batching, picker routing and batch scheduling asks for an integrated solution approach. In practice, customers request ever shorter delivery times, so faster order picking operations and a quick response to new customer orders is indispensable. To offer a high customer service level, new orders should be accounted for in a dynamic fashion. Therefore, in this paper, a new metaheuristic optimisation algorithm for the integrated batching, routing and scheduling problem is proposed, that is able to handle dynamic order arrivals. It is first shown that this algorithm outperforms a state-of-the-art algorithm in the static problem setting in which all orders are known at the start of the planning period. Next, a large-scale numerical study for an online problem setting is performed.

This leads to several interesting managerial insights. In the online setting, it was shown that batching orders of multiple trucks is beneficial when looking at the customer service level and order picking efficiency. However, if orders of trucks far into the future may be batched as well, we show the need to anticipate on future order arrivals to keep customer service levels high. To the best of our knowledge, this is the first time order anticipation is implemented in this field of study. Order anticipation was found to improve the results. However, a trade-off was identified between service level and operating efficiency when anticipating on future order arrivals. For the spare parts warehouse under study, these results are very valuable. Large efficiency improvements are possible if they start batching over multiple trucks, compared to the current situation of only picking the most urgent orders. Although they were hesitant to implement large scale batching, dreading a decrease in their customer service levels, our study shows that this concern is resolved by anticipating on future order arrivals.

Nevertheless, in practice, other constraints may be involved that are not yet taken into account. Warehouses may use large sorting and packing installations, imposing restrictions on the number of different trucks that can be handled at the same time. Exploring the impact of these and other real-life constraints is worth investigating in the future, to make the model resemble reality more closely. Furthermore, upgrading the system to a fully dynamic order picking system, where picker rerouting is allowed, should allow even faster customer response times, but may lead to more stressful situations for the order pickers. Yet, this trade-off could lead to interesting research opportunities, bringing human factors into account as well. Finally, although order anticipation clearly leads to an improved solution quality, it is worth looking into how to best implement this setting. Right now, the algorithm anticipates on the average

expected number of orders and updates this number during the planning period based on the current order arrival pattern. Studying the impact of other order arrival patterns and other strategies to update the expected number of orders, may lead to even better results and seems thus worthwhile.

Funding

Ruben D'Haen is funded by the Research Foundation Flanders (FWO-11J1721N). This work is also supported by the project Data-driven logistics (FWO Strategic Basic Research, S007318N). 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.

Disclosure of interest

The authors report no conflict of interest.

Data availability statement

The data that support the findings of this study are openly available in OSF at doi.org/10.17605/osf.io/7r4gn.

References

Alipour, Mehrdad, Yahia Zare Mehrjedrdi, and Ali Mostafaeipour. 2020. "A rule-based heuristic algorithm for on-line order batching and scheduling in an order picking warehouse with multiple pickers." *RAIRO - Operations Research* 54 (1): 101–107. Number: 1 Publisher: EDP Sciences.

Ardjmand, Ehsan, Heman Shakeri, Manjeet Singh, and Omid Sanei Bajgiran. 2018. "Minimizing order picking makespan with multiple pickers in a wave picking warehouse." *International Journal of Production Economics* 206: 169–183.

Attari, Mahdi Yousefi Nejad, Ali Ebadi Torkayesh, Behnam Malmir, and Ensiyeh Neyshabouri Jami. 2020. "Robust possibilistic programming for joint order batching and picker routing problem in warehouse management." *International Journal of Production Research* 0 (0): 1–19. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207543.2020.1766712.

Bozer, Y. A., and J. W. Kile. 2008. "Order batching in walk-and-pick order picking systems." *International Journal of Production Research* 46 (7): 1887–1909.

Chen, Fangyu, Hongwei Wang, Yong Xie, and Chao Qi. 2016. "An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse." *Journal of Intelligent Manufacturing* 27 (2): 389–408.

Chen, Fangyu, Yongchang Wei, and Hongwei Wang. 2018. "A heuristic based batching and assigning method for online customer orders." *Flexible Services and Manufacturing Journal* 30 (4): 640–685.

Chen, Tzu-Li, Chen-Yang Cheng, Yin-Yann Chen, and Li-Kai Chan. 2015. "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem." *International Journal of Production Economics* 159: 158–167.

Coelho, Leandro C., Jean-François Cordeau, and Gilbert Laporte. 2014. "Heuristics for dynamic and stochastic inventory-routing." *Computers & Operations Research* 52: 55–67.

de Koster, René, Tho Le-Duc, and Kees Jan Roodbergen. 2007. "Design and control of warehouse order picking: A literature review." *European Journal of Operational Research* 182 (2): 481–501.

Ferrucci, Francesco, Stefan Bock, and Michel Gendreau. 2013. "A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods." *European Journal of Operational Research* 225 (1): 130–141.

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

Giannikas, Vaggelis, Wenrong Lu, Brian Robertson, and Duncan McFarlane. 2017. "An interventionist strategy for warehouse order picking: Evidence from two case studies." *International Journal of Production Economics* 189: 63–76.

Gil-Borrás, Sergio, Eduardo G. Pardo, Antonio Alonso-Ayuso, and Abraham Duarte. 2020. "GRASP with Variable Neighborhood Descent for the online order batching problem." *Journal of Global Optimization* 78 (2): 295–325.

Gil-Borrás, Sergio, Eduardo G. Pardo, Antonio Alonso-Ayuso, and Abraham Duarte. 2021. "A heuristic approach for the online order batching problem with multiple pickers." *Computers & Industrial Engineering* 160: 107517.

Grosse, Eric H., Christoph H. Glock, Mohamad Y. Jaber, and W. Patrick Neumann. 2015. "Incorporating human factors in order picking planning models: framework and research opportunities." *International Journal of Production Research* 53 (3): 695–717. Publisher: Taylor & Francis.

Helsgaun, Keld. 2000. "An effective implementation of the Lin–Kernighan traveling salesman heuristic." *European Journal of Operational Research* 126 (1): 106–130.

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

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

Henn, Sebastian, and Gerhard Wäscher. 2012. "Tabu search heuristics for the order batching problem in manual order picking systems." *European Journal of Operational Research* 222 (3): 484–494.

Kuhn, Heinrich, Daniel Schubert, and Andreas Holzapfel. 2020. "Integrated Order Batching and Vehicle Routing Operations in Grocery Retail – A General Adaptive Large Neighborhood Search Algorithm." *European Journal of Operational Research* .

Kumar, Shashank, Balkrishna E. Narkhede, and Karuna Jain. 2021. "Revisiting the warehouse research through an evolutionary lens: a review from 1990 to 2019." *International Journal of Production Research* 59 (11): 3470–3492. Publisher: Taylor & Francis.

Leung, K.H., Daniel Y. Mo, G.T.S. Ho, C.H. Wu, and G.Q. Huang. 2020. "Modelling near-real-time order arrival demand in e-commerce context: a machine learning predictive methodology." *Industrial Management & Data Systems* 120 (6): 1149–1174. Publisher: Emerald Publishing Limited.

Li, Jianbin, Rihuan Huang, and James B. Dai. 2017. "Joint optimisation of order batching and picker routing in the online retailer's warehouse in China." *International Journal of Production Research* 55 (2): 447–461.

Lu, Wenrong, Duncan McFarlane, Vaggelis Giannikas, and Quan Zhang. 2016. "An algorithm for dynamic order-picking in warehouse operations." *European Journal of Operational Research* 248 (1): 107–122.

Pisinger, David, and Stefan Ropke. 2007. "A general heuristic for vehicle routing problems." *Computers & Operations Research* 34 (8): 2403–2435.

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

Rubrico, Jose I. U., Toshimitu Higashi, Hirofumi Tamura, and Jun Ota. 2011. "Online rescheduling of multiple picking agents for warehouse management." *Robotics and Computer-Integrated Manufacturing* 27 (1): 62–71.

Scholz, André, Daniel Schubert, and Gerhard Wäscher. 2017. "Order picking with multiple pickers and due dates – Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems." *European Journal of Operational Research* 263 (2): 461–478.

Shaw, Paul. 1998. "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems." In *Principles and Practice of Constraint Programming — CP98*, edited by Michael Maher and Jean-Francois Puget, Lecture Notes in Computer Science, Berlin, Heidelberg, 417–431. Springer.

Theys, Christophe, Olli Bräysy, Wout Dullaert, and Birger Raa. 2010. "Using a TSP heuristic for routing order pickers in warehouses." *European Journal of Operational Research* 200 (3): 755–763.

Tsai, C.-Y., J. J. H. Liou, and T.-M. Huang. 2008. "Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time." *International Journal of Production Research* 46 (22): 6533–6555.

Ulmer, Marlin. 2017. "Delivery deadlines in same-day delivery." *Logistics Research* 10 (3): 1–15.

van Gils, Teun, An Caris, Katrien Ramaekers, and Kris Braekers. 2019. "Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse." *European Journal of Operational Research* 277 (3): 814–830.

van Gils, Teun, Katrien Ramaekers, Kris Braekers, Benoît Depaire, and An Caris. 2018a. "Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions." *International Journal of Production Economics* 197: 243–261.

van Gils, Teun, Katrien Ramaekers, An Caris, and Mario Cools. 2017. "The use of time series forecasting in zone order picking systems to predict order pickers' workload." *International Journal of Production Research* 55 (21): 6380–6393. Publisher: Taylor & Francis.

van Gils, Teun, Katrien Ramaekers, An Caris, and René B. M. de Koster. 2018b. "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.

Van Nieuwenhuyse, Inneke, and René B. M. de Koster. 2009. "Evaluating order throughput time in 2-block warehouses with time window batching." *International Journal of Production Economics* 121 (2): 654–664.

Vanheusden, Sarah, Teun van Gils, Katrien Ramaekers, Trijntje Cornelissens, and An Caris. 2022. "Practical factors in order picking planning: state-of-the-art classification and review." *International Journal of Production Research* 0 (0): 1–25. Publisher: Taylor & Francis.

Won, J., and S. Olafsson. 2005. "Joint order batching and order picking in warehouse operations." *International Journal of Production Research* 43 (7): 1427–1442.

Xu, Xianhao, Tian Liu, Kunpeng Li, and Weihong Dong. 2014. "Evaluating order throughput time with variable time window batching." *International Journal of Production Research* 52 (8): 2232–2242.

Zhang, Jun, Xuping Wang, Felix T. S. Chan, and Junhu Ruan. 2017. "On-line order batching and sequencing problem with multiple pickers: A hybrid rule-based algorithm." *Applied Mathematical Modelling* 45: 271–284.

Zhang, Jun, Xuping Wang, and Kai Huang. 2016. "Integrated on-line scheduling of order batching and delivery under B2C e-commerce." *Computers & Industrial Engineering* 94: 280–289.

Žulj, Ivan, Sergej Kramer, and Michael Schneider. 2018. "A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem." *European Journal of Operational Research* 264 (2): 653–664.

Appendix

Appendix A. MIP model

In this appendix the mixed integer programming model for the online IBRSP is shown. Note that this formulation describes the reoptimisation problem in every optimisation step. This reoptimisation problem is solved at the start of the planning period and every time a picker returns to the depot. Since most of the constraints are similar to the formulation of van Gils et al. (2019), only the required changes to their model are discussed here. These adaptations are related to the objective function as well as the online context.

An additional decision variable is introduced:
τ_k tardiness of order k.

The objective function is hierarchical and tries to minimise tardiness and order pick time as the primary and secondary objective, respectively.

$$\text{Primary objective:} \qquad \min \sum_{k \in \kappa} \tau_k \tag{A1}$$

$$\text{Secondary objective:} \qquad \min \sum_{q \in \sigma} T_{qP} \tag{A2}$$

Subject to all constraints mentioned in van Gils et al. (2019), except for their constraints (15) and (16). Constraint (15) is changed since pickers may still be performing a previous pick round, and should thus not get a new batch assigned before their current batch is completed. Moreover, the completion time of a batch should be equal to the current time of the system plus the pick time of that, and possible previous, batch(es). If we define T_{q0} as the completion time of the current batch of order picker q, constraint (15) of van Gils et al. (2019) is changed to:

$$T_{q(p-1)} + t_{setup} Z_{qp0} + t_{search} \sum_{k \in \kappa} o_k R_{qpk} + \sum_{a \in \alpha} t_a X_{qpa} = T_{qp} \quad \forall q \in \sigma \quad \forall p \in \pi \tag{A3}$$

In case a picker is waiting at the depot (which is always the case in the initialisation but may also happen in later optimisation steps), his T_{q0} is equal to the current time of the system. If he is performing a picking tour that was assigned to him in a previous optimisation step, T_{q0} is equal to that batch's completion time.

Constraint (16) of van Gils et al. (2019) is slightly modified to include tardiness as follows:

$$T_{qp} \leq t_k + \tau_k + \mathcal{M}(1 - R_{qpk}) \quad \forall q \in \sigma \quad \forall p \in \pi \quad \forall k \in \kappa \tag{A4}$$

Additionally, the following non-negativity constraint regarding tardiness is added:

$$\tau_k \geq 0 \quad \forall k \in \kappa \tag{A5}$$

Note that the set of customer orders, κ, contains only those orders that are not yet picked, are part of the currently active trucks and have a release time less than or equal to the current time.

Appendix B. LNS Operators

This appendix presents an overview of the destroy operators that were tested during the algorithm development. Most of them did not provide significant improvements and were therefore not selected to be part of the final algorithm. After the discussion of the destroy operators, the process of finding the final algorithm is discussed in this appendix as well. The destroy operators can be divided into two large groups: those working on the order level and those working on the batch level.

Destroy operators working on the order level:

(1) Random order removal: randomly select orders and remove them from the solution.
(2) Remove order high distance: for every order, find the reduction in picking travel distance if it is removed from the solution. Remove the orders with the highest distance reduction per order line.
(3) Remove order largest due date difference: for every order look at the difference between its due time and the average due time of orders in its batch. Remove the orders with the largest difference.
(4) Remove order largest minimal pairwise distance: for every order, find the largest minimal pairwise distance and remove orders with the largest values. The minimal pairwise distance is the smallest distance of an order line to an order line of another order in the same batch. First calculate this distance for every order line in a batch. Next, for every order, find the largest minimal pairwise distance between its order lines and save this value. Orders with the largest values are removed.
(5) Remove order largest average pairwise distance: calculate the minimal pairwise distance for all order lines. Save the average value of all order lines of an order and remove the order with the largest average distance.
(6) Largest number of additional aisles order: for every order, look at the difference between the number of unique subaisles to visit when picking the batch with this order and the number of unique subaisles to visit for that batch when the order is removed. Remove the orders with the largest difference in unique subaisles to visit.
(7) Largest additional covering area order: for every order, calculate the rectangular covering area of all order lines in the batch with this order and the rectangular covering area of all order lines in that batch if the order is removed. Remove the orders with the largest difference in covering area.

Destroy operators working on the batch level:

(8) Random batch destroy: a random batch is selected and removed from the solution.
(9) Destroy batch high distance: destroy batches with the largest picking travel distance per order line.
(10) Largest difference due time batch: for every batch, find the difference between the average due time of orders in the batch and the earliest due time of an order in this batch. Destroy batches with the largest difference between these two values.
(11) Smallest distance savings batch: for every batch, calculate the difference between the total picker travel distance to pick every order line individually and the travel distance of the batch. Destroy the batches with the smallest difference in travel

distance.

(12) Largest total minimal pairwise distance: for every batch, calculate the sum of the minimal pairwise distance of all its order lines. Destroy the batches with the largest sum of these distances.

(13) Largest number of aisles batch: for every batch, calculate the number of unique subaisles to be visited. Remove the batches with the largest number of unique subaisles.

(14) Largest number of additional aisles batch: for every batch, find the difference between the number of unique subaisles to visit when picking this batch and the number of unique subaisles to visit for this batch's order with the most unique subaisles. Destroy the batches with the largest difference in subaisles.

(15) Largest covering area batch: for every batch, compute the rectangular covering area for all its order lines and destroy the batches with the largest areas.

(16) Largest additional covering area batch: for every batch, calculate the difference between the rectangular covering area of the whole batch and the covering area of this batch's order with the largest covering area. Destroy the batches with the largest difference in covering area.

With the above destroy operators, several algorithm configurations could be constructed. Testing every possible configuration is not feasible in reasonable time (there are 2^{16} possible combinations with these destroy operators), so we used the following approach. First, some small-scale experiments were run to identify potentially good operator combinations, starting with every operator being applied in the algorithm. In every iteration of the algorithm, a single destroy operator is randomly selected. By counting how often a certain operator led to a new best solution, insight in which operators were most likely leading to improvements was obtained. This number of improvements formed the basis to select which combinations of operators looked promising. If two algorithms had very similar performance, the one with the least operators was preferred, to keep things simple.

Second, the most promising algorithms were tested on 30 test instances, with 3 runs on every instance. As discussed in Section 4.2, the algorithm is split in two phases, i.e., phase 1 when the best found solution contains tardiness and the total number of iterations is smaller than the maximum number of local search iterations, and phase 2 when there is no longer tardiness in the best found solution or the current iteration number is larger than the maximum number of local search iterations. The tested algorithms with their results are shown in Table B1.

In general, the results suggest that the operators working on the batch level are the most important ones to reduce tardiness, while the operators working on the order level are better suited to reduce the order pick time. The performance of the destroy operator random order removal is remarkable, and suggests that reducing order pick time is possible by using this operator only. Looking at more problem specific operators seems unnecessary to reduce order pick time. Of the tested algorithms, algorithm 6, consisting of three batch operators (11, 14, 16) in phase 1, combined with random order removal (1) in phase 2, was considered best due to having the lowest tardiness and still good order pick time.

Table B1.: Tests of the six most promising algorithm configurations. The destroy operators used in phase 1 and phase 2 of the algorithm are given with the resulting tardiness and order pick time of the test runs. The selected algorithm is highlighted in bold.

Algorithm	Destroy operators		Tardiness (s)	Order pick time (s)
	Phase 1	Phase 2		
Algorithm 1	1 - 16	1 - 16	1439.83	48,937.39
Algorithm 2	1 - 16	1	1429.34	48,642.56
Algorithm 3	8 - 16	8 - 16	1410.96	49,458.24
Algorithm 4	11, 14, 16	1 - 16	1485.72	48,869.68
Algorithm 5	11, 14, 16	1, 3, 6	1482.78	48,659.88
Algorithm 6	11, 14, 16	1	**1402.06**	**48,677.21**

Appendix C. ANOVA

Table C1.: Results mixed factorial ANOVA on tardiness.

	Sum of squares	df	Mean square	F	p value
Main effects					
AverageNumberOfOrders	524420.7	1	524420.7	41.863	0.000
NumberOfTrucks	63226.94	2	31613.47	2.524	0.081
TruckDistribution	90636.66	1	90636.66	7.235	0.007
OrderArrivalPattern	119661.8	3	39887.26	3.184	0.023
Variation	262695.2	1	262695.2	20.970	0.000
PercentageActiveTrucks	462569.8	1.008	458792.6	71.581	0.000
OrderAnticipation	1293.65	1	1293.65	7.228	0.007
Two-way interaction					
AverageNumberOfOrders × NumberOfTrucks	49574.7	2	24787.35	1.979	0.139
AverageNumberOfOrders × TruckDistribution	67281.65	1	67281.65	5.371	0.021
AverageNumberOfOrders × OrderArrivalPattern	144592.9	3	48197.63	3.848	0.009
AverageNumberOfOrders × Variation	138426.8	1	138426.8	11.050	0.001
NumberOfTrucks × TruckDistribution	17263.71	2	8631.853	0.689	0.502
NumberOfTrucks × OrderArrivalPattern	68492.2	6	11415.37	0.911	0.486
NumberOfTrucks × Variation	52296.94	2	26148.47	2.087	0.125
TruckDistribution × OrderArrivalPattern	7024.685	3	2341.562	0.187	0.905
TruckDistribution × Variation	111678.8	1	111678.8	8.915	0.003
OrderArrivalPattern × Variation	31898.39	3	10632.8	0.849	0.467
PercentageActiveTrucks × OrderAnticipation	718.422	1.288	557.859	2.475	0.107
PercentageActiveTrucks × AverageNumberOfOrders	287148.9	1.008	284804.1	44.435	0.000
PercentageActiveTrucks × NumberOfTrucks	66665	2.016	33060.32	5.158	0.006
PercentageActiveTrucks × TruckDistribution	1773.124	1.008	1758.645	0.274	0.602
PercentageActiveTrucks × OrderArrivalPattern	49934.96	3.025	16509.07	2.576	0.052
PercentageActiveTrucks × Variation	127927.2	1.008	126882.6	19.796	0.000
OrderAnticipation × AverageNumberOfOrders	521.432	1	521.432	2.913	0.088
OrderAnticipation × NumberOfTrucks	578.915	2	289.458	1.617	0.199
OrderAnticipation × TruckDistribution	41.582	1	41.582	0.232	0.630
OrderAnticipation × OrderArrivalPattern	780.592	3	260.197	1.454	0.226
OrderAnticipation × Variation	160.624	1	160.624	0.897	0.344
Residuals					
Between subjects	10823281	864	12526.95		
Within PercentageActiveTrucks	5583310	871.113	6409.396		
Within OrderAnticipation	154635.8	864	178.977		
Within PercentageActiveTrucks × OrderAnticipation	250750.3	1112.675	225.358		

Table C2.: Results mixed factorial ANOVA on order pick time.

	Sum of squares	df	Mean square	F	p value
Main effects					
AverageNumberOfOrders	37150.74	1	37150.74	428.279	0.000
NumberOfTrucks	78.066	2	39.033	0.450	0.638
TruckDistribution	792.556	1	792.556	9.137	0.003
OrderArrivalPattern	18360.32	3	6120.105	70.553	0.000
Variation	872.563	1	872.563	10.059	0.002
PercentageActiveTrucks	50407.49	1.222	41256.81	5439.448	0.000
OrderAnticipation	50.327	1	50.327	40.038	0.000
Two-way interaction					
AverageNumberOfOrders × NumberOfTrucks	144.443	2	72.221	0.833	0.435
AverageNumberOfOrders × TruckDistribution	376.925	1	376.925	4.345	0.037
AverageNumberOfOrders × OrderArrivalPattern	130.381	3	43.460	0.501	0.682
AverageNumberOfOrders × Variation	3.340	1	3.340	0.039	0.844
NumberOfTrucks × TruckDistribution	234.719	2	117.359	1.353	0.259
NumberOfTrucks × OrderArrivalPattern	321.965	6	53.661	0.619	0.716
NumberOfTrucks × Variation	380.902	2	190.451	2.196	0.112
TruckDistribution × OrderArrivalPattern	400.256	3	133.419	1.538	0.203
TruckDistribution × Variation	536.05	1	536.05	6.180	0.013
OrderArrivalPattern × Variation	408.886	3	136.295	1.571	0.195
PercentageActiveTrucks × OrderAnticipation	124.121	1.891	65.651	51.798	0.000
PercentageActiveTrucks × AverageNumberOfOrders	964.061	1.222	789.051	104.031	0.000
PercentageActiveTrucks × NumberOfTrucks	316.514	2.444	129.528	17.077	0.000
PercentageActiveTrucks × TruckDistribution	17908.01	1.222	14657.09	1932.444	0.000
PercentageActiveTrucks × OrderArrivalPattern	7.055	3.665	1.925	0.254	0.894
PercentageActiveTrucks × Variation	133.908	1.222	109.599	14.450	0.000
OrderAnticipation × AverageNumberOfOrders	6.356	1	6.356	5.056	0.025
OrderAnticipation × NumberOfTrucks	23.603	2	11.802	9.389	0.000
OrderAnticipation × TruckDistribution	7.405	1	7.405	5.891	0.015
OrderAnticipation × OrderArrivalPattern	12.318	3	4.106	3.266	0.021
OrderAnticipation × Variation	4.224	1	4.224	3.360	0.067
Residuals					
Between subjects	74947.02	864	86.744		
Within PercentageActiveTrucks	8006.708	1055.634	7.585		
Within OrderAnticipation	1086.042	864	1.257		
Within PercentageActiveTrucks × OrderAnticipation	2070.356	1633.49	1.267		

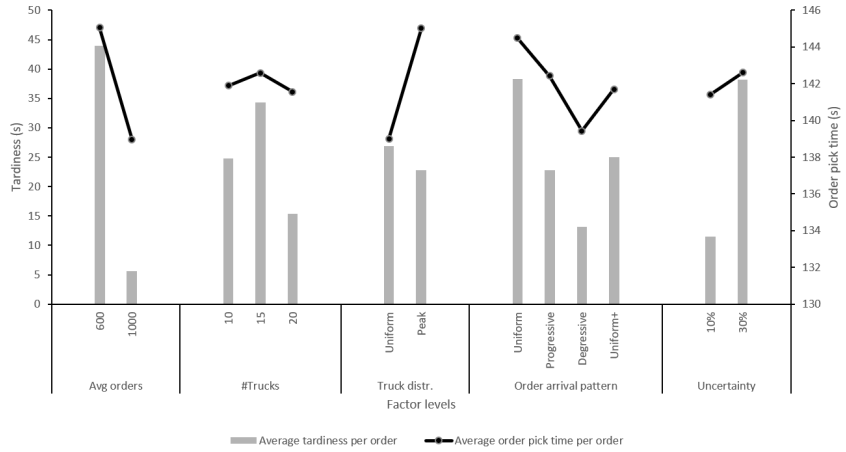
Appendix D. Figures factor levels

This appendix shows detailed results of the full factorial design. The results are split between 20%, 40% and 60% of the trucks active, as well as with and without order anticipation. The values on the axes are identical in every case to allow easy comparisons between the graphs.

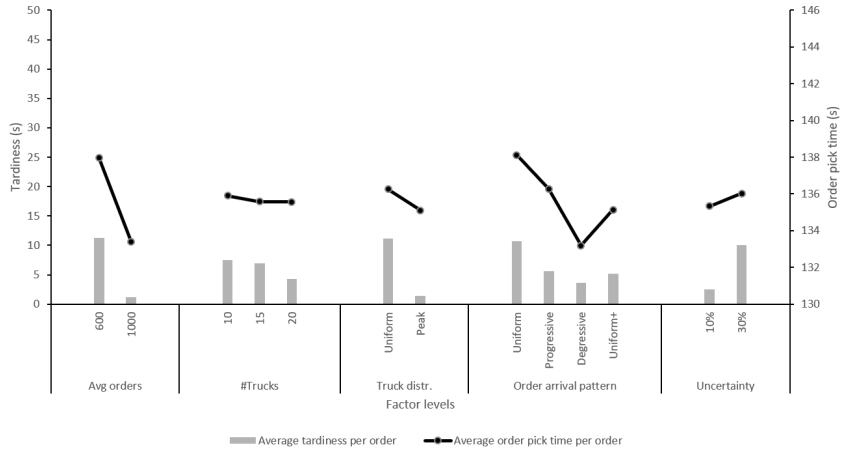
A first observation is the improved algorithm performance when using order anticipation. When the algorithm anticipates on future orders, the tardiness is a bit lower, with the effect getting larger for a larger percentage of active trucks. The ratio between the factors and the factor levels remains approximately equal between the with and without order anticipation cases.

Regarding the percentage of active trucks, there is one remarkable difference between 20% active trucks and 40% or 60%, situated at the factor truck distribution. Under the uniform distribution, a larger part of the orders should have already been picked at every point of the planning period. However, in the truck distribution with a peak, it is hard to schedule everything during the peak period. This effect is visible in the tardiness: while there is barely tardiness in the 40% and 60% cases for the peak distribution, the tardiness for the 20% case is almost as high as under the uniform distribution. In the 20% case, the algorithm cannot look far into the future. This makes anticipating on the peak impossible, and results in a higher tardiness. In the 40% or 60% case, the algorithm can anticipate on the busy peak period, resulting in better solutions.

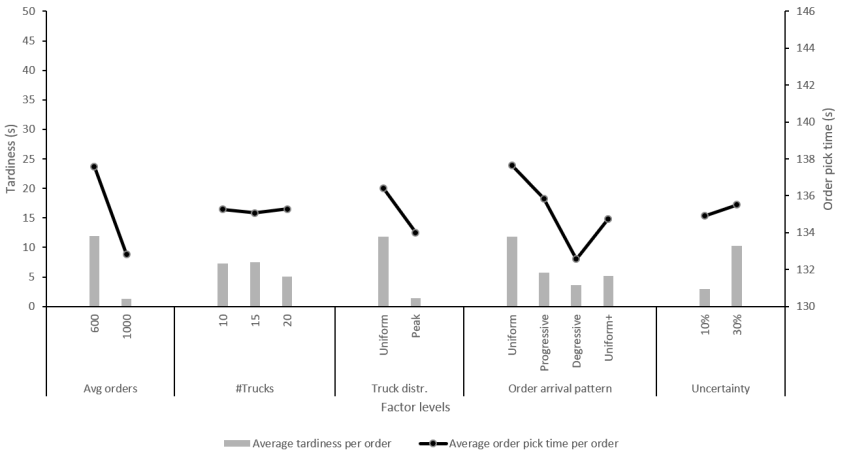
The difference in order pick time between the settings is even more remarkable. While the order pick time is higher for the peak truck distribution than for the uniform truck distribution in the 20% case, it is the other way around for the 40% or 60% case. A possible interpretation is the following. When only 20% of the trucks can be picked at the same time, anticipating on the peak period is impossible. Because it is so busy in this period, the algorithm has to focus on finding solutions with limited tardiness and is unable to find very efficient batches to reduce the order pick time. In the 40% or 60% case, it is easier to schedule the busy peak period as the algorithm can start planning this period earlier, which, combined with the larger pool of pickable orders, allows the creation of more efficient batches. This results in a lower order pick time.



(a) No order anticipation, 20% trucks

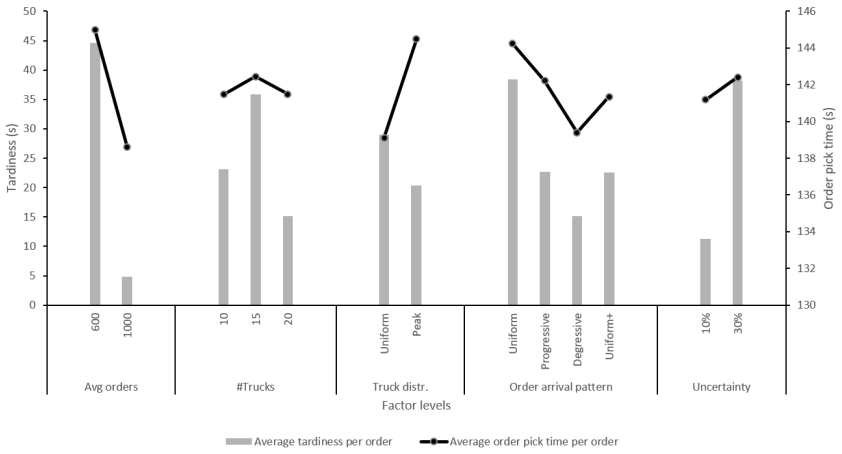


(b) No order anticipation, 40% trucks

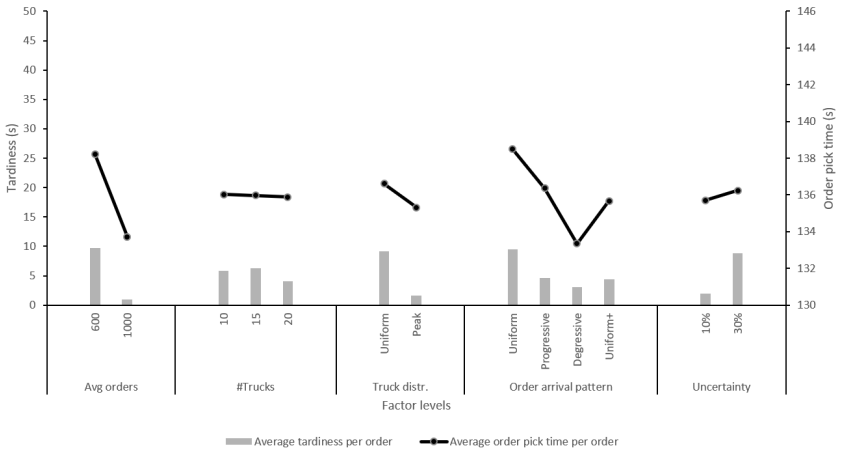


(c) No order anticipation, 60% trucks

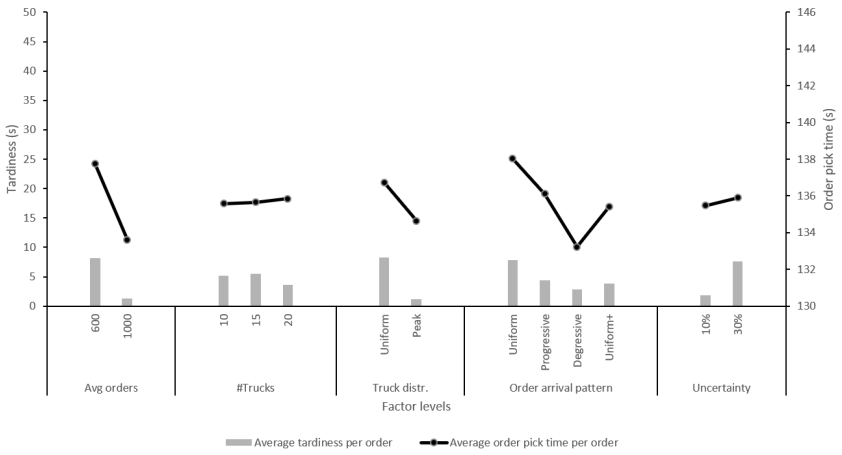
Figure D1.: Probability density function of the order arrival patterns. The horizontal axis shows the time before the truck departure, with 0 equal to the truck departure, 8 and 0.5 the earliest and latest possible arrival time of an order, respectively.



(d) Order anticipation, 20% trucks



(e) Order anticipation, 40% trucks



(f) Order anticipation, 60% trucks

Figure D1.: Probability density function of the order arrival patterns. The horizontal axis shows the time before the truck departure, with 0 equal to the truck departure, 8 and 0.5 the earliest and latest possible arrival time of an order, respectively.

Appendix E. Results algorithm

This appendix gives a short summary of the algorithm results in Table E1. The minimum, average and maximum number of batches per picker, as well as the standard deviation in the number of batches per picker, is shown in the first part of the table. These values are split between the cases with and without order anticipation, and for the instances with three and five order pickers.

Next, for all pickers combined, averaged over all instances with three and five pickers, respectively, the average number of batches and the average number of optimisation steps for a complete instance are given. The number of optimisation steps is a little less than the number of batches, because a batch is assigned to every picker after the initialisation. Afterwards, an optimisation step is performed every time an order picker returns to the depot, followed by assigning a single batch to this picker.

Table E1.: Composition of the solutions of the algorithm.

	Without order anticipation		With order anticipation	
Batches per picker	3 pickers	5 pickers	3 pickers	5 pickers
Minimum	15	14	15	14
Average	21.85	21.44	21.89	21.42
Maximum	34	35	33	33
Standard deviation	2.42	2.77	2.41	2.65
All pickers combined				
Average number of batches	65.54	107.21	65.33	106.51
Average number of opt. steps	63.68	104.04	63.44	103.24