Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

Algorithms for on-line order batching in an order picking warehouse

Sebastian Henn*

Faculty of Economics and Management, Otto-von-Guericke University Magdeburg, P.O. Box 4120, D-39016 Magdeburg, Germany

ARTICLE INFO

Available online 8 January 2012

Keywords:
Warehouse management
Order picking
Order batching
On-line optimization

ABSTRACT

In manual order picking systems, order pickers walk or ride through a distribution warehouse in order to collect items required by (internal or external) customers. Order batching consists of combining these – indivisible – customer orders into picking orders. With respect to order batching, two problem types can be distinguished: in off-line (static) batching, all customer orders are known in advance; in on-line (dynamic) batching, customer orders become available dynamically over time. This paper considers an on-line order batching problem in which the maximum completion time of the customer orders arriving within a certain time period has to be minimized. The author shows how heuristic approaches for off-line order batching can be modified in order to deal with the on-line situation. In a competitive analysis, lower and upper bounds for the competitive ratios of the proposed algorithms are presented. The proposed algorithms are evaluated in a series of extensive numerical experiments. It is demonstrated that the choice of an appropriate batching method can lead to a substantial reduction of the maximum completion time.

1. Introduction

Order picking is a warehouse function dealing with the retrieval of items (article units) from their storage locations in order to satisfy (internal or external) customer orders. It arises because incoming articles are received and stored in (large volume) unit loads while customers tend to order small volumes of different articles [1]. Order picking is critical to each supply chain since underperformance results in unsatisfactory customer service (long processing and delivery times) and high costs (labor costs, costs of additional and/or emergency shipments). Even though different attempts have been made to automate the picking process, systems involving human operators are still prevalent in practice. Such manual order picking systems can be differentiated into two categories [2]: picker-to-parts systems, where order pickers drive or walk through the warehouse and collect the required items; and parts-to-picker systems, where automated storage and retrieval systems deliver the items to stationary order pickers. In systems of the first kind, which are considered in this paper, three activities at the operative level can be distinguished [3]: the assignment of articles to storage locations (article location), the transformation of customer orders into picking orders (order batching) and the routing of pickers through the warehouse (picker routing). This paper stresses the second activity where different customer orders can be combined into picking orders (batches) and jointly released for picking. This activity has proven to be pivotal for the efficiency of warehouse operations [4].

With respect to the availability of customer orders, two situations for batching customer orders can occur [5]: in *off-line* (*static*) *batching* all customer orders are known at the beginning of the (short-term) planning period (shift or day). In *on-line* (*dynamic*) *batching* customer orders become available dynamically over time. Batches have to be formed based only on the known customer orders. The aim of this paper is to examine how solution approaches for the static batching problem can be modified for on-line situations in order to improve the warehouse efficiency. Moreover, decision rules are proposed that define which customer orders should be satisfied directly and which ones should be satisfied later, if on a specific point in time more than one batch can be released.

The remainder of this paper is organized as follows: In Section 2 the on-line order batching problem (OOBP) will be defined and an optimization model will be given. Section 3 contains an overview of the relevant literature. Algorithms for the OOBP will be presented in Section 4. To analyze the solution quality of these algorithms and to show the limitations for possible algorithms, a competitive analysis of the generated solutions will be performed in Section 5. Moreover, numerical experiments have been carried out to evaluate the performance of the proposed algorithms. The purpose and design of the numerical study will be described in Section 6. The performance of the algorithms will be compared for different problem classes in Section 7. The paper will conclude with a summary and an outlook on further research topics.

2. On-line order batching problem

2.1. Problem description

In a manual picker-to-parts system *order pickers* are guided by *pick lists*, which specify the sequence in which the storage

* Tel.: +49 391 67 11841; fax: +49 391 67 18223.
E-mail address: sebastian.henn@ovgu.de

locations should be visited as well as the number of items demanded of each article. A pick list may contain the items of a single customer order or of a combination of customer orders. The picking process can be described in the following way: the order picker starts at the *depot*, walks (or rides on an appropriate vehicle) through the warehouse and collects items from different storage locations. Afterwards, he/she returns to the depot and hands over the picked items. The corresponding route through the warehouse is typically determined by means of a so-called routing strategy. Despite the fact that an optimal, polynomial time algorithm for the picker routing problem exists [6], it is hardly ever used in practice. Order pickers do not seem to accept the optimal routes, since they are not always straightforward and sometimes even confusing [4]. Two well-known strategies are the *S-Shape* and the *Largest Gap* heuristic which provide the required non-confusing routing schemes. Fig. 1 demonstrates the straightforward character of both routing schemes for a set of items to be picked. The black rectangles symbolize the corresponding locations where items have to be picked (pick locations).

Order picking is usually done with the help of a picking device (e.g., cart, roll pallet, etc.). Consequently, customer orders can be combined until the capacity of the picking device is exhausted. This capacity is typically defined by a number of items. The splitting of a customer order into two or more batches is prohibited, since this would result in additional unacceptable sorting effort (*order integrity condition*). If an order picker has started a tour through the warehouse, an interruption of this process and a rearrangement of the orders is also prohibited.

The time period necessary to complete a batch is called (batch) *service time* (batch *processing time*). The service time of a customer order is defined as the service time of the batch the customer order is assigned to. The batch service time is composed of the *travel time*, i.e. the time period the order picker needs to travel from the depot to the first pick location, between the pick locations and from the last pick location to the depot; the *search time*, i.e. the time period needed for the identification of articles; the *pick time*, i.e. the time period needed for moving the items from the pick location on the picking device; and the *setup time*, i.e. the time period for administrative and set-up tasks at the beginning and the end of each tour [7].

In this paper an order picking system is considered in which customer requests are submitted to the warehouse only in a specific time period. Therefore, it is not necessary that the warehouse has to operate day and night, and the order pickers retrieve the customer orders only during a shift. During this shift, customer orders are not known in advance, but become available over time. The decision regarding which customer orders should be processed together in a batch has to be made without considering the information of future incoming orders. Due to the limited operating times in the warehouse it has to be decided how long customer orders have to be accepted. The decision that no later-arriving customer order will be processed in this shift is given with the last customer order which is sent to the order picking area including the information that no further customer order will arrive. (For the case that no 'real' customer order exists, the information that no further order will be processed during this shift can be submitted by an (virtual) empty order.)

The point in time when a customer order becomes available is called *arrival time*. The *start time* (*release time*) of a batch is the point in time when an order picker starts to process this batch. The start time of an order is identical to the start time of the batch the order is assigned to. The point in time when the order picker returns to the depot after having collected all items is called *completion time* of a batch or of a customer order, respectively.

The (customer order) *waiting time* can be determined as the length of the time period between the arrival time and the start time of a customer order. The *turnover time* (*response time*) is the time period for which a customer order stays in the system, i.e. the time period between the arrival and the completion time of a customer order.

If the number of arriving customer orders is too large for processing each customer order separately in an appropriate total time, customer orders must be combined to batches. The *On-line Order Batching Problem* (OOBP) consists of grouping customer orders into batches such that the maximum completion time (or makespan) – identical to the completion time of the last released batch – is minimized. In the following, we discuss the situation with a single order picker, i.e. all batches must be processed one after another.

2.2. Optimization model

In the following, an optimization model for the off-line version of the OOBP is formulated. The model, which requires the complete information of all incoming orders, is presented in order to analyze the structure of the problem. These constants are used:

- n: number of customer orders;
- m: upper bound on the number of required batches (a trivial upper bound can be $m=n$);
- r_i: arrival time of customer order i ($\forall i \in \{1,\dots,n\}$), where $0 \le r_i \le r_{i+1}$ holds;
- t_{setup}: setup time, i.e. time necessary for administrative and set-up tasks for each batch;
- v_{travel}: travel velocity, i.e. number of length units the order picker can cover in the warehouse per time unit;
- v_{pick}: pick velocity, i.e. number of items the order picker can search and pick per time unit;
- w_i: number of items of customer order i ($\forall i \in \{1,\dots,n\}$);
- W: maximal number of items which can be included in a batch.

The model uses the following variables:

s_j: start time of batch j ($\forall j \in \{1,\dots,m\}$);
$\mathbf{x}_j := (x_{j1},\dots,x_{jn})^T$ a vector in $\{0,1\}^n$ where

$$x_{ji} = \begin{cases} 1 & \text{if customer order } i \text{ is assigned to batch } j \\ 0 & \text{else} \end{cases} \quad \forall j \in \{1,\dots,m\}$$

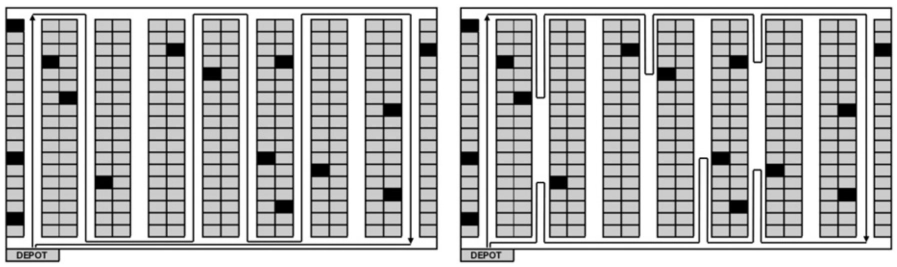

Fig. 1. Example of S-Shape (left) and Largest Gap (right) heuristic in a single-block warehouse.

Without loss of generality, it is assumed that batch j is started after batch $j-1$ has been completed, i.e. $s_j > s_{j-1}$. The total length of the picking tour for a particular batch involving a particular routing method is determined by the function $d : \{0,1\}^n \mapsto \mathbb{Q}$. For an empty batch, the function value is zero. The requested items and their pick locations are hidden in the function d. The problem can be modeled as follows:

$$\min \max_{j \in \{1,\ldots,m\}} \left\{ s_j + d(\mathbf{x}_j)/v_{\text{travel}} + \sum_{i=1}^{n} w_i x_{ji}/v_{\text{pick}} + t_{\text{setup}} \,\middle|\, \text{at least one } x_{ji} = 1 \right\} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ji} = 1, \quad \forall i \in \{1,\ldots,n\} \tag{2}$$

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

$$s_j \ge \max_{i \in \{1,\ldots,n\}} \{r_i \cdot x_{ji}\}, \quad \forall j \in \{1,\ldots,m\} \tag{4}$$

$$s_j \ge s_{j-1} + d(\mathbf{x}_{j-1})/v_{\text{travel}} + \sum_{i=1}^{n} w_i x_{j-1,i}/v_{\text{pick}} + t_{\text{setup}} \quad \forall j \in \{2,\ldots,m\} \tag{5}$$

$$s_j \ge 0, \quad \forall j \in \{1,\ldots,m\} \tag{6}$$

$$\mathbf{x}_j \in \{0,1\}^n, \quad \forall j \in \{1,\ldots,m\} \tag{7}$$

In the objective function (1) the expression $d(\mathbf{x}_j)/v_{\text{travel}} + \sum_{i=1}^{n} w_i x_{ji}/v_{\text{pick}} + t_{\text{setup}}$ represents the service time of a batch. This sum is composed of the time the order picker needs to travel through the warehouse, the time he/she needs to pick the items and the setup time. By addition of the start time s_j of the batch j, its completion time is obtained. In summary, (1) minimizes the maximum completion time of all batches with at least one assigned customer order. Eq. (2) ensures the assignment of each customer order to exactly one batch. Furthermore, inequalities (3) guarantee that the capacity of the picking device is not violated. Conditions (4) indicate that a batch is started when all customer orders assigned to this batch are known. From (5) it follows that a batch is started no earlier than the previous one is completed. Finally, the constraints of types (6) and (7) indicate that start times are non-negative and that variables $\mathbf{x}_j$ are binary vectors, respectively.

To verify the model, it remains to show that empty batches do not affect the objective function value. This is necessary since (5) causes that $s_j > s_{j-1}$ if t_{setup} is set greater than 0. This inequality also holds in case no customer order is assigned to batch $j-1$. Consider an instance in which only $\tilde{m}$ batches ($\tilde{m} < m$) are needed to process all customer orders. Without loss of generality, it is assumed that $s_1 < s_2 < \cdots < s_{\tilde{m}-1} < s_{\tilde{m}}$ holds for the start time of these batches. Then, the start times $s_{\tilde{m}+k}$ are set to $s_{\tilde{m}+k-1} + t_{\text{setup}}$ and $x_{i,\tilde{m}+k}$ are set to 0 ($\forall k \in \{1,\ldots,m-\tilde{m}\}, i \in \{1,\ldots,n\}$). This solution satisfies all constraints. Their objective function value is the completion time of batch $\tilde{m}$ which is equal to the maximum completion time.

It has to be noticed that the number of batches the customer order can be assigned to grows exponentially with the number of orders. Gademan and van de Velde [8] proved the $\mathcal{NP}$-hardness of the (off-line) Order Batching Problem (in which the total travel time for a given set of customer orders has to be minimized), if the number of orders which can be assigned to a batch is greater than two. Since the problem described here is a generalization of the (off-line) Order Batching Problem (all orders have identical arrival times) it is also $\mathcal{NP}$-hard.

3. Literature review

The minimization of picking times depends on the following three planning issues: article location, order batching and picker routing. The simultaneous solution of these problems, which would be representing a global optimum, is not a very realistic approach [2]. In practice, the decisions for the planning issues are made sequentially and existing order batching methods consider the routing policy as constant [1], whereas only one integrated approach solving the batching and routing problem simultaneously has been proposed so far.

For the off-line Order Batching Problem Gademan and van de Velde [8] presented a branch-and-price algorithm with column generation that is able to solve small instances to optimality in a reasonable amount of computing time. For the case of S-Shape routing, Bozer and Kile [9] presented a mixed integer programming approach that generates near-optimal solutions for small sets of orders (up to 25). For larger off-line problems the use of heuristics is still advisable. These heuristic approaches can be distinguished in four groups. The first ones are *priority rule-based algorithms*, where customer orders are ranked according to a priority value and are then assigned to batches following this rank [10]. The probably best-known and straightforward way is the application of the First-Come-First-Served rule (FCFS). The second group consists of *seed algorithms*, introduced by Elsayed [11], which generate batches sequentially. A customer order is selected as the start order for a batch. Additional customer orders are assigned to that batch according to an order-congruency rule. An overview of the various seed selection and order-congruency rules was given by Ho et al. [12]. Methods of the third group, *savings algorithms*, are based on the Clarke-and-Wright-Algorithm for the Vehicle Routing Problem [13] and have been adapted in several ways for the Order Batching Problem. For each pair of customer orders, the savings can be obtained by collecting all items of the two customer orders in one (large) tour instead of collecting them in two separate tours [14]. Starting with the pair of customer orders with the largest savings, pairs are considered for being assigned to a batch in a non-ascending order. Finally, the last group contains *metaheuristics*. Hsu et al. [15] presented a genetic algorithm for the Order Batching Problem. Their approach includes an aisle-metric for the determination of the tour lengths and is, therefore, limited to S-Shape routing, only. Tsai et al. [16] described an integrated approach, in which solutions to the batching problem as well as to the routing problem are determined by genetic algorithms. Iterated Local Search and Ant Colony Optimization were proposed by Henn et al. [17].

With respect to on-line batching Kamin [18] described a real-world problem, in which greeting cards have to be retrieved from a warehouse. Order pickers use automated guided vehicles on a fixed course to collect greeting cards according to customer orders. Besides a competitive analysis, the system is simulated and evaluated according to different objectives including the time needed to complete all customer orders of a day. The impacts of different batching strategies and problem parameters are investigated. In contrast to the problem here, Kamin selected the next batch which should be processed according to a due date, i.e. a point in time when the customer order is due to be completed.

A batching problem where customer orders arrive at different times was given by Won and Olafsson [19]. The authors observed a trade-off between total service time and turnover time of a customer order. They formulated an optimization model where the objective function is a weighted sum of service and waiting time of customer orders within a batch. For this approach the arrival times of the customer orders must be known in advance. A two-step procedure was proposed in which batches are constructed by means of the FCFS rule. The subsequent routing problem is solved by a 2-opt procedure.

Elsayed and Lee [20] described an automated storage and retrieval system where articles have to be picked from the warehouse. The arrivals of the orders are dynamic and due dates have been assigned to each order. The objective function is meant to form batches and sequence them in a way that the tardiness (maximum of the completion time minus the due date and zero) of the customer orders is minimized. The authors distinguished between a static and a dynamic case. In the static case, customer orders arriving in a particular time interval form a group. This time interval is determined by the time necessary to process the customer orders of the previous group. In the dynamic case, a customer order arriving while a group of customer orders is being processed is added to the set of non-processed customer orders. This results in a new set of batches. In their approach, customer orders are sequenced according to their due dates and the times needed to process them in single batches. According to this sequence, three decision rules are proposed and evaluated for the selection of batches: a nearest schedule rule, a shortest service time rule, and a most common locations rule.

The application of these approaches to the OOBP described above is limited since the proposed algorithms incorporate due dates in their objective functions.

Apart from these approaches, *time window batching* is prevalent in the on-line situation [21]. Time window batching can be carried out in two different variants: fixed and variable time window batching. In fixed time window batching all orders arriving during a particular time interval are assigned to one batch. In variable time window batching the order picker waits until a particular number of orders has arrived, and collects the items of these orders in a joint tour.

Chew and Tang [7] described an on-line problem in which the number of order pickers is limited. They carried out a theoretical analysis of travel and service times on the basis of S-Shape routing. To measure the quality of their estimation the authors simulated the picking system as a queuing network with two queues. In the first one, orders arrive according to a Poisson process and batches are generated by means of the FCFS rule. They use variable time window batching where each batch consists of exactly n_0 orders. If n_0 orders are in the first queue, these orders are assigned to a batch and move to the second queue. The batches in the second queue are released successively according to the availability of order pickers. In numerical experiments the authors focused on the optimal number of orders which should be assigned to a batch such that average turnover time is minimized. For a 2-block warehouse, Le-Duc and de Koster [22] proved an estimation for the average turnover time for a random order and observed similar results to Chew and Tang with an uniformly distributed demand frequency. They concluded that the average turnover time is a convex function of the number of orders per batch (batch size). On the one hand, a large batch size leads to a small average service time of each order, but to a large average waiting time. On the other hand, the average service time is large for a small batch size, whereas the average waiting time is small. Le-Duc and de Koster [22] also mentioned possible extensions to this model, i.e. multiple order pickers and multi-line orders. A queuing model to determine the average turnover time for an order in a 2-block warehouse for variable and fixed time window batching is given by van Nieuwenhuyse and de Koster [21].

The presented OOBP is related to the On-line Batch Processing Problem (OBPP) of minimizing the makespan and to the On-line Dial-a-Ride Problem (ODARP) in which the completion time has to be minimized. In the first problem, a machine can process dynamically arriving jobs together in a batch. The start times and completion times of the jobs in the same batch are identical. Each job has a (single) processing time, and the processing time of a batch is determined by the longest (single) processing time of any job in the batch. In the capacitated version the number of jobs, which can be processed in the same batch, is limited. According to the three field classification for scheduling problems this problem is classified as $1/r_j,B,\text{on}-\text{line}/C_{\max}$ [23]. The difference to the OOBP is that the batch service time is not just determined by the longest single service time of the customer orders assigned to the batch, but controlled by all customer orders which are assigned to the batch. The corresponding off-line problem, in which all jobs are available at the beginning, can be solved to optimality by application of the Full Batch Longest Processing Time (FBLPT) rule. In this rule the jobs with the longest single processing time are assigned to a batch until the capacity restriction is violated, then the jobs with the next-longest processing time form another batch and so on. Zhang et al. [23] proposed two on-line algorithms (called H^B and MH^B). If the machine becomes idle or the machine is idle and a new job arrives, a set of batches is determined by application of the FBLPT rule. If this results in more than one batch the batch with the shortest (H^B) or longest (MH^B) processing time is released. If only one batch is available it will be started immediately or postponed. This decision depends on the actual time, the release date and the processing time of the job with the longest processing time in the batch. Poon and Yu [24] proposed a similar algorithm which starts the batch with the second-longest processing time if more than one batch can be released. If only one batch can be started the decision of whether the start should be released or postponed depends on the actual time, a constant and the processing time of the actual batch.

In the ODARP objects are to be transported between several points. Each dynamically arriving request is characterized by a source and a destination. Each object has to be transported from its source to its destination. Due to capacity limitations only a predefined number of objects can be transported in parallel. The problem consists of the determination of a transportation schedule with minimal completion time of all requests [25,26]. The OOBP can be interpreted as an extension of the ODARP, i.e. the pick locations correspond to the sources and the depot to the destination of each request. One specific characteristic of the OOBP is that, due to the order integrity requirement, a request consists of a set of sources (pick locations) which have to be accessed during the same tour. Another characteristic is that while the order picker processes a batch, no additional order can be assigned to the batch, which is usually possible in the ODARP. For the ODARP several strategies can be identified. Following the IGNORE strategy an optimal schedule for the set of known orders is determined. The algorithm follows that schedule and ignores all new requests until this schedule is completed [25,26]. In the REPLAN strategy the actual schedule is completely discarded and a new schedule is computed each time a new request arrives [25]. Like in the OBPP, Ascheuer et al. [25] proposed a strategy (SMARTSTART) in which no request will be served until a specific point in time is reached. This point depends on the actual time, a constant parameter and the time required to complete the schedule for the actual known requests.

4. Algorithms

4.1. Basic principle

The difference between on-line and off-line problems consists of the availability of the input parameters. Formally, an instance of an on-line problem can be described as a (input) sequence of requests. In the OOBP, this is a sequence of customer orders $1,\dots,n$ with different arrival times $r_1,\dots,r_n$. An *on-line algorithm* applied to this sequence has to deal with each request i at time r_i, independent of the requests $i+1,\ldots,n$. An algorithm for the OOBP has to form and release batches without having complete information on the types and the arrival times of future customer orders.

The points in time when a decision of this kind has to be made are called *decision points*. These can be distinguished into three types:

Type A: A set of unprocessed (also called *open*) customer orders exists and an order picker becomes available. Decision points of this type appear at the beginning of the planning period or at a completion time of a batch. At this point either the next batch should be released directly or its start should be postponed to a later point in time.

Type B: An order picker is idle and a new customer order arrives. The algorithm can rearrange the set of batches and determine a start time for the next batch.

Type C: The last customer order arrives.

The basic principle of the proposed on-line algorithm combines ideas of Kamin [18] for a special warehouse type and of Zhang et al. [23] for the related OBPP. At each decision point t the algorithm determines a solution of the off-line Order Batching Problem for all customer orders which are known but have not been processed at that time. The set $P(t)$ of these customer orders is called *set of open orders*. Since the off-line Order Batching Problem is $\mathcal{NP}$-hard, an exact solution of the mathematical model derived in the previous section or the application of the branch-and-price algorithm described in [8] is not possible or requires a large amount of computing time, which is not reasonable in a real-time environment. Therefore, the application of a batching heuristic $\mathcal{H}_b$ is suggested in order to obtain a set of batches $B(t)$. If the solution for batching $P(t)$ is a single batch j with service time st_j, it has to be decided whether the batch should be started immediately or if the start should be postponed. It should be avoided that after the release of a batch a new customer order arrives which could have been added to the previous batch. The immediate start would result in a larger maximal completion time as in the case that the start would be postponed and the order would be added to this batch. Therefore, a waiting time is included which is chosen conceptually similar to [23] in case of the OBBP. Each customer order of batch j is assigned to a distinct batch and for each of these batches the service time is calculated. The time – during which the order is processed together with no other order – is called single service time of a customer order. Let i' be a customer order which would require the longest single service time $st_{i'}$. It is expected that the order with the longest single service time has the largest impact on the batch service time. Therefore, this order serves as reference order for the calculation of the waiting period. Also the arrival time of this order is compared to the actual time. If this difference is small (the order has arrived shortly before) a large waiting period exists in order to include similar customer orders which arrive in the following time units. If the difference is large, i.e. the order is known for a longer time period, it is intended that similar orders have arrived in the meantime and further waiting is not necessary. To sum up, the start of the batch is scheduled to $\max\{t,(1+\alpha)r_{i'}+\alpha st_{i'}-st_j\}$, unless a new customer order arrives in the meantime. This case would be a decision point of type B and a new set of batches $B(t)$ would be determined. α is a parameter in [0,1]. This choice of the waiting period ensures that the batch (if no further order arrives before the start of this batch) is completed at $(1+\alpha)r_{i'}+\alpha st_{i'}$, which is by definition less than or equal to $2t+st_j$. If α is set to 0 a single batch would be started immediately since $\max\{t,r_{i'}-st_j\}=t$, which represents the actual time. In this case the algorithm behaves like strategies in which no waiting period is included, like the REPLAN strategy developed for the ODARP. The largest waiting time occurs in case that α is 1. If more than one batch can be released, i.e. $|B(t)|>1$, waiting for a further customer order is not necessary. Therefore, a batch from $B(t)$ is selected and released according to a selection rule $\mathcal{H}_s$. Algorithm 1 ($\mathcal{A}$) summarizes this approach.

Algorithm 1. Basic principle of an on-line order picking algorithm.

```
Decision Point A and B (at time $t$):
  generate a set of batches $B(t)$ by means of batching heuristic
  $\mathcal{H}_b$ to $P(t)$;
  if $|B(t)|=1$ then
    let $j$ be the batch in $B(t)$;
    $i'=\arg\max\{st_i|i$ is assigned to $j\}$;
    the start of batch $j$ is scheduled to
    $\max\{t,(1+\alpha)r_{i'}+\alpha st_{i'}-st_j\}$;
  else
    select one batch $j$ of $B(t)$ according to selection rule $\mathcal{H}_s$;
    start batch $j$;
  end if
Decision Point C:
  generate a set of batches $B(t)$ by means of batching heuristic
  $\mathcal{H}_b$ to $P(t)$;
  release all batches sequentially;
```

4.2. Routing strategies

Service times have to be calculated in the basic principle, in the batching heuristics and in the selection rules. Since service times depend basically on the length of the picking tour, the specification of a routing strategy is necessary. For the routing strategy the S-Shape heuristic and the Largest Gap heuristic are suggested (cf. Fig. 1) which have already been introduced in Section 2.

The *S-Shape heuristic* provides solutions in which the order picker enters and traverses an aisle completely if at least one required item is located in that aisle (an exception would be the last aisle if the order picker is positioned on the front cross-aisle). Afterwards, the order picker moves to the next aisle which has to be visited [27].

The *Largest Gap heuristic* gives a solution in which the order picker completely traverses the first aisle and the last aisle containing a demanded item. All other aisles – containing at least one required item – are entered from the front and from the back in a way that the non-traversed distance between two adjacent pick locations or the end of the aisle is maximal.

4.3. Batching heuristics

For the batching heuristic $\mathcal{H}_b$ every mentioned batching heuristic for the off-line Order Batching Problem which are described in Section 3 can be used. Three options, namely the *First-Come-First-Served* (FCFS) rule, the *savings algorithm* C&W(ii) and *Iterated Local Search* (ILS) are considered. Seed algorithms are not presented here since they generate batches sequentially. A seed algorithm will serve in the numerical experiments as benchmark heuristic.

In *First-Come-First-Served* the customer orders are ranked according to their arrival time. With respect to the capacity constraint and this list the customer orders are assigned to batches.

At the beginning of the *savings algorithm* C&W(ii) each customer order is assigned to a different batch [14]. The savings are computed for each combination of batches. These values can be obtained by collecting the items of both batches in one (large) tour instead of collecting them in two separate tours. The pairs of batches are ranked according to their savings. The algorithm searches for the pair with the largest savings which can be combined with respect to the capacity restriction. In case two batches were combined, the savings are computed again, and one searches for a further combination of batches. If no combination is possible or no pair with positive savings exists the algorithm terminates.

Iterated Local Search is based on the Local Search principle, where one starts from a candidate solution and searches iteratively for better neighbor solutions. ILS consists of two alternating phases, an improvement and a perturbation phase. In the first phase one starts from an initial solution and terminates in a local optimum. The vicinity of this local optimum (used as an incumbent solution) will be explored in order to identify a solution with an improved objective function value. In the perturbation phase, the incumbent solution is partially destroyed and a further improvement search phase is applied to this solution. This new local optimum has to pass an acceptance criterion in order to become the new incumbent solution, otherwise the previous solution remains the incumbent solution for a further perturbation. These two phases are repeated until a termination condition is met. Henn et al. [17] modified ILS to the off-line Order Batching Problem in the following way: an initial solution is generated by means of the FCFS rule. Two different solutions (a solution can be described as a set of batches) are called neighbors if one solution can be achieved from the other by interchanging two orders from different batches (SWAP) or by assigning one order to a different batch (SHIFT). The local search phase uses a systematic first improvement strategy: an attempt is made to reduce the maximum completion time (which serves as objective function here) of the customer orders by a SWAP. If an improvement can be obtained, the algorithm proceeds with this new solution and searches for another SWAP which improves the maximum completion time. If no SWAP can be applied in order to reduce the maximum completion time, an attempt is made to achieve a local minimum with respect to the SHIFT neighborhood. After the identification of a local minimum, a search is again made for a SWAP that leads to an improved solution. This is repeated until no improvement by SWAPs and SHIFTs can be identified.

In the perturbation phase, two different batches k and l are selected randomly and the first q orders from batch k are moved to batch l, and in exchange the first q orders of l are moved to k (q is a random number, which is at most half of the number of orders in k and l, respectively). If this rearrangement is not possible due to capacity constraints, remaining orders will be assigned to a new batch. The number of rearrangements – a critical step in the algorithm – is set to $m^\star \cdot \theta + 1$, where $m^\star$ is the number of batches in the best-known solution and θ (rearrangement parameter) is a constant in [0,1]. Regarding the acceptance criterion, a new solution is accepted as an incumbent solution if its maximum completion time is shorter than the best currently known one. Also a few deteriorating steps are allowed if a sequence of perturbation phases and local search phases applied to a particular incumbent solution does not lead to a new global best solution within a certain time interval $t_{\text{incumbent}}$. The solution obtained in the last local search phase will be chosen as the new incumbent solution if the ratio between the maximum completion time of the last solution and the best-known maximum completion time is smaller than a threshold $1+\mu$, where the threshold parameter μ is in [0,1]. Otherwise the incumbent solution will be modified again.

4.4. Selection rules

For the selection rule $\mathcal{H}_s$ four different options (FIRST, SHORT, LONG, SAV) are suggested. The selection rule FIRST chooses the first batch of $B(t)$. If batching is carried out by means of the FCFS rule, this rule selects a batch whose customer orders have minimal arrival times among all open customer orders.

The other three selection rules are meant to control the choice of a specific batch in a way that influences the set of customer orders and subsequent decisions. SHORT determines a batch with the shortest service time. Batches with a short service time may not include customer orders which guide the order picker to aisles far from the depot. If the demand frequency of articles is low in those aisles it may be preferable to collect customer orders, demanding these items, and process them together at the end of the planning period.

As opposed to the previous strategy LONG selects the batch which requires the longest service time. With this rule the time interval between two batching steps is very large and while the batch is processed new customer orders may arrive which can be combined more favorably with the non-processed customer orders.

SAV computes for each batch a savings value, i.e. the sum of the single service times of the assigned customer orders and subtracts the batch service time. By selecting a batch with the largest sum, the intention is to release a batch with similar orders.

5. Competitive analysis

5.1. Scope

The *competitive analysis* is a common approach for the evaluation of the performance of on-line algorithms [28]. It is based on the considerations of Sleator and Tarjan [29] and analyzes the performance on each input sequence. This is done by comparison of an on-line algorithm with an optimal off-line algorithm, i.e. an algorithm that determines an optimal solution for the complete input sequence. More formally, an on-line algorithm $\mathcal{A}$ for a minimization problem is called c-competitive if a constant β exists such that for all possible input sequences $\mathcal{I}$ the inequality $\mathcal{A}(\mathcal{I}) \le c \cdot \text{OPT}(\mathcal{I}) + \beta$ holds, where $\mathcal{A}(\mathcal{I})$ is the objective function value provided by algorithm $\mathcal{A}$ for instance $\mathcal{I}$ and $\text{OPT}(\mathcal{I})$ is the objective function value of an optimal off-line solution for $\mathcal{I}$. The infimum over all c for which the inequality holds is called *competitive ratio of* $\mathcal{A}$.

5.2. Lower bound

The objective function value of an algorithm $\mathcal{A}$ (the maximal completion time) can be expressed as sum of the batch service times and possible idle times. As described in Section 2 the batch service time is composed of the travel time, the pick time and the setup time. $\mathcal{A}(\mathcal{I})$ is therefore composed of the travel times of all batches, the pick times of all batches, the setup times of all batches and possible idle times. Also, $\text{OPT}(\mathcal{I})$ consists of these four components. In both expressions the total pick time is identical.

A general lower bound for the competitive ratio has to be valid for each warehouse configuration. In order to show that a general lower bound different from 1 does not exist, a specific warehouse configuration is considered. In this specific situation the batch service time depends only on the pick time – the time needed for moving the items from the pick location on the picking device – (i.e. $v_{\text{travel}} \to \infty$ and $t_{\text{setup}} = 0$). For many warehouse configurations an optimal off-line algorithm can make use of idle times if unprocessed orders are known, e.g. in case that a new order will arrive immediately. Waiting for a new arriving order can reduce the total travel time or will result in a smaller number of batches reducing the number of setup tasks. However, benefits by combining a customer order with a later arriving customer order cannot be achieved for the pick time. In the described warehouse configuration, in which the batch service times only depend on the pick times, idle times will increase the maximal completion time if unprocessed orders exist. In these warehouse configurations an optimal off-line algorithm will not determine idle times for the order picker while unprocessed orders are known. Moreover, each on-line algorithm (which starts each batch immediately) behaves like an optimal off-line algorithm and the corresponding competitive ratio is 1.

This implies that it is not possible to derive a general lower bound greater than 1 for algorithms which start batches immediately, since a general lower bound has to be valid for each warehouse configuration.

Considering the total travel time only some simple lower bound instances stemming from lower bound examples for on-line vehicle routing problems [30] can be constructed for $\mathcal{A}$ (defined in Section 4.1). The following instances are just an exemplary illustration in order to show how lower bounds can be constructed in general. The examples do not depend on the used batching strategy and the used selection rule. For the analysis a single-block warehouse with two cross-aisles, one in the front and one in the back of the picking area is assumed. The depot is located in front of the leftmost (picking) aisle and all (picking) aisles are vertically orientated. The layout of the warehouse is depicted in Fig. 2. The warehouse consists of U aisles with C storage locations (cells) on each side of an aisle. Being positioned in the center of an aisle, the order picker can pick items from cells on the right, as well as from cells on the left without additional movements. Whenever the order picker leaves an aisle he/she has to move (in order to reach the cross-aisle) a distance equal to the width of a cell l_{cw} in vertical direction from the first storage location, or from the last storage location, respectively. Let l_c denote the center-to-center distance between two aisles.

Let two customer orders (i_1,i_2) be available at time 0. Both orders require $W/2+1$ items and cannot be included in the same batch. Order i_1 requires only items stored in the first aisle whereas i_2 requires the items in the first storage location of an aisle in the right part of the warehouse. The location of the items is chosen in a way that for both orders an identical travel distance (one in a horizontal direction, one in a vertical direction) has to be covered. This distance can be determined as two times the minimum of the distance from the depot to the last cell of the first aisle and the distance from the depot to the first cell of the last aisle, i.e. $2\cdot\min\{C\cdot l_{cw};(U-1)\cdot l_c+1\cdot l_{cw}\}$. If $C\cdot l_{cw}$ is the smaller distance an aisle u ($u\in\{1,\dots,U\}$) is chosen in a way that $(u-1)\cdot l_c+1\cdot l_{cw}$ is equal to $C\cdot l_{cw}$. Otherwise, a storage location c ($c\in\{1,\dots,C\}$) is selected so that $c\cdot l_{cw}=(U-1)\cdot l_c+1\cdot l_{cw}$. (It should be noted that equality cannot always be ensured due to the characteristics of the parameters and the integer values u and c. In this case an aisle or cell is selected so that the difference is minimal. A possible deviation between both distances will be neglected in the following since it has only a small impact on the presented bound.) Let the corresponding travel time be denoted by t_{travel}. The routing for both batches according to the S-Shape heuristic and the Largest Gap strategy, respectively, is shown in Fig. 2. $\mathcal{A}$ forms two batches, of which each batch has a service time

$$st=t_{\text{travel}}+\frac{W/2+1}{v_{\text{pick}}}+t_{\text{setup}}$$

$\mathcal{A}$ releases one of the batches directly. Let an order i_3 be available (at time 1) which requires $W/2-1$ items. If the batch including order i_1 is processed first the order i_3 requires the items in the last cell of the first aisle. In case that the batch including order i_2 is processed first the order i_3 requested identical items which are requested by order i_2. For picking the two remaining orders a batch is built which requires a travel time of $2t_{\text{travel}}$. Therefore

$$\mathcal{A}(\mathcal{I})=3t_{\text{travel}}+\frac{W+W/2+1}{v_{\text{pick}}}+2t_{\text{setup}}$$

An optimal algorithm will process first the order which is not identical to i_3 and than both similar orders in one batch

$$\text{OPT}(\mathcal{I})=2t_{\text{travel}}+\frac{W+W/2+1}{v_{\text{pick}}}+2t_{\text{setup}}$$

For the competitive ratio the following expression can be calculated

$$\frac{\mathcal{A}(\mathcal{I})}{\text{OPT}(\mathcal{I})}=\frac{3t_{\text{travel}}+\dfrac{W+W/2+1}{v_{\text{pick}}}+2t_{\text{setup}}}{2t_{\text{travel}}+\dfrac{W+W/2+1}{v_{\text{pick}}}+2t_{\text{setup}}}$$

A lower bound of 3/2 can be obtained if the pick time and the setup time can be neglected. Since the result only holds for these conditions the bound 3/2 is not valid for each warehouse configuration.

In the appendix (supplementary material) a special warehouse configuration is considered. For this (unrealistic) condition it is shown that the competitive ratio of $\mathcal{A}$ (combined with all proposed batching strategies and proposed selection rules) converges to 2. It should be noted that for this ratio a very special warehouse configuration is required and that the result is not valid in general.

5.3. Upper bound

In the following it is shown that an algorithm exists whose competitive ratio is at most 2, independent of the kind of warehouse. In the following, the proof of Zhang et al. [23] for the OBPP is adapted to the OOBP for which only some small modifications are necessary. For this proof it is required that an optimal batching strategy $\mathcal{H}_{\text{opt}}$ is used, i.e. the algorithm obtains an optimal solution for the off-line Order Batching Problem. Let $\mathcal{A}^\star$ be $\mathcal{A}$ using $\mathcal{H}_{\text{opt}}$ and an arbitrary selection rule. The arrival time of the last customer order is denoted by r_n (for the sake of simplicity it is assumed that exactly one customer order has the arrival time r_n). In order to prove that the competitive ratio of $\mathcal{A}^\star$ is less or

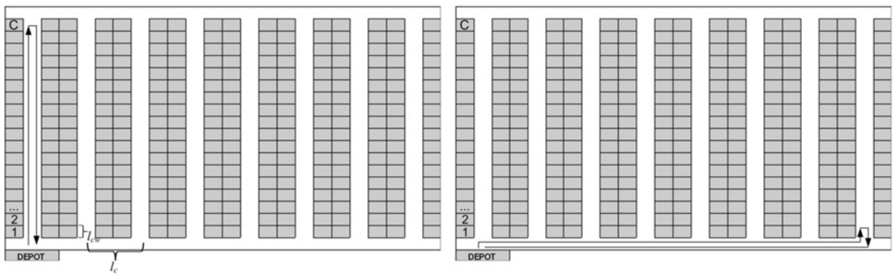

Fig. 2. Routing for the customer orders i_1 (left) and i_2 (right).

equal than 2, a distinction is made between the following three cases at time r_n:

1. *All customer orders, which have arrived before customer order n, are already completed before* r_n: $\mathcal{A}^\star$ obtains an optimal solution.
2. *There are unprocessed customer orders and the order picker is idle*: In this case the solution of the batching problem at r_{n-1} leads to a single batch. According to $\mathcal{A}^\star$, the last batch j includes all open customer orders without customer order n at time r_n and has service time st_j. The batch j would be started at $(1+\alpha)r_{i'}+\alpha st_{i'}-st_j$ where i' is a customer order assigned to batch j with the longest single service time $st_{i'}$. Since the order picker is idle, $(1+\alpha)r_{i'}+\alpha st_{i'}-st_j$ must be greater than r_n. Since the batching heuristic is optimal, $\mathcal{A}^\star$ will lead to a solution which is at least as good as a solution that will release j followed by a separated batch for order n. For $\mathcal{A}^\star(\mathcal{I})$ the following is obtained:
$$\begin{aligned}\mathcal{A}^\star(\mathcal{I}) &\le r_n+st_j+st_n < (1+\alpha)r_{i'}+\alpha st_{i'}-st_j+st_j+st_n\\ &= (1+\alpha)r_{i'}+\alpha st_{i'}+st_n\end{aligned}$$
If $st_{i'} \ge st_n$ holds, the upper bound for $\mathcal{A}^\star(\mathcal{I})$ can be estimated as follows:
$$\begin{aligned}\mathcal{A}^\star(\mathcal{I}) &< (1+\alpha)r_{i'}+\alpha st_{i'}+st_{i'}\\ &= (1+\alpha)(r_{i'}+st_{i'}) \le (1+\alpha)\mathrm{OPT}(\mathcal{I}) \le 2\mathrm{OPT}(\mathcal{I})\end{aligned}$$
Otherwise ($st_{i'} < st_n$) one obtains
$$\begin{aligned}\mathcal{A}^\star(\mathcal{I}) &< (1+\alpha)r_{i'}+\alpha st_n+st_n \le (1+\alpha)r_n+(1+\alpha)st_n\\ &= (1+\alpha)(r_n+st_n) \le 2\mathrm{OPT}(\mathcal{I})\end{aligned}$$
3. *There are unprocessed customer orders and the order picker is not idle*: Let $\tilde{j}$ be the batch which is being processed while customer order n arrives. Let $s_{\tilde{j}}$ be the start time and $f_{\tilde{j}}$ be the completion time of $\tilde{j}$, therefore $s_{\tilde{j}} < r_n \le f_{\tilde{j}}$. Let r be the earliest arrival time in the time interval $(s_{\tilde{j}}, f_{\tilde{j}}]$, i.e. $r=\min_i\{r_i | s_{\tilde{j}} < r_i \le f_{\tilde{j}}\}$ and $A(r)$ be the set of the customer orders arriving after r, more formally $A(r) := \{i | r_i > r\}$. Let $st^\star_{A(r)}$ and $st^\star_{P(s_{\tilde{j}})}$ be the total service time of $A(r)$ and the set of open customer orders $P(s_{\tilde{j}})$ at time $s_{\tilde{j}}$, respectively. Furthermore, the inequalities $\mathrm{OPT}(\mathcal{I}) \ge r+st^\star_{A(r)}$ and $\mathrm{OPT}(\mathcal{I}) \ge st^\star_{P(s)}$ hold. Let $\tilde{I}$ be the set of customer orders assigned to batch $\tilde{j}$. All customer orders of $\tilde{I}$ must be available before $\tilde{j}$ is started and, therefore, $\tilde{I} \subset P(s_{\tilde{j}})$. The solution of the on-line algorithm is $\mathcal{A}^\star(\mathcal{I}) = f_{\tilde{j}}+st^\star_{A(r)\cup\{P(s_{\tilde{j}})\setminus\tilde{I}\}}$, where $st^\star_{A(r)\cup\{P(s_{\tilde{j}})\setminus\tilde{I}\}}$ is the service time for a set of batches for the customer orders in $A(r)\cup\{P(s_{\tilde{j}})\setminus\tilde{I}\}$. Since the algorithm uses an optimal batching strategy, $st^\star_{A(r)\cup\{P(s_{\tilde{j}})\setminus\tilde{I}\}}$ is the optimal service time for the customer orders in $A(r)\cup\{P(s_{\tilde{j}})\setminus\tilde{I}\}$. Let further $st^\star_{P(s_{\tilde{j}})\setminus\tilde{I}}$ be the optimal service time for $P(s)\setminus\tilde{I}$. Then, $st^\star_{A(r)\cup\{P(s_{\tilde{j}})\setminus\tilde{I}\}} \le st^\star_{P(s_{\tilde{j}})\setminus\tilde{I}}+st^\star_{A(r)}$ is valid. For the estimation of the upper bound, the following is obtained:
$$\begin{aligned}\mathcal{A}^\star(\mathcal{I}) &= f_{\tilde{j}}+st^\star_{A(r)\cup\{P(s)\setminus\tilde{I}\}} \le f_{\tilde{j}}+st^\star_{P(s_{\tilde{j}})\setminus\tilde{I}}+st^\star_{A(r)}\\ &= s_{\tilde{j}}+st^\star_{P(s_{\tilde{j}})}+st^\star_{A(r)} \le r+st^\star_{P(s)}+st^\star_{A(r)} \le 2\mathrm{OPT}(\mathcal{I})\end{aligned}$$

To summarize, the competitive ratio for $\mathcal{A}^\star$ is less than or equal to 2. Combined with the consideration above, warehouse configurations exist for which this bound is tight.

5.4. Discussion

For the OOBP it is shown that the competitive ratios of the proposed algorithms depend on the warehouse characteristics. Moreover, warehouse configurations exist for which the proposed algorithms are at best 2-competitive. For the related OBPP on-line algorithms with a competitive ratio less than $1+(\sqrt{5}-1)/2$ do not exist if the number of jobs, which can be assigned to a batch, is greater than or equal to 2 [23]. Feuerstein and Stougie [26] proved by transferring results from the On-line Traveling Salesman Problem that every deterministic algorithm for the ODARP must have a competitive ratio of at least 2. Whereas the ODARP (usually) allows for updating the current tour at each point, the current tour of an order picker is unchangeable with the OOBP. In the ODARP and the OBPP the objective function depends on a single criterion (travel length or single processing time) whereas the objective function of the OOBP includes travel time and pick time. Therefore, the lower bound for the competitive ratio for the ODARP does not imply the competitive ratio for the OOBP.

However, for some (unrealistic) warehouse configurations each algorithm in which idle times are prohibited if unprocessed orders are known behaves like an optimal off-line algorithm. Since a general lower bound cannot be identified, it is required to analyze the solution quality of the proposed algorithms in problem classes which represent more practical situations (cf. Section 7).

Zhang et al. [23] proposed for the OBPP that the competitive ratio of their algorithms H^B and MH^B is at most 2. These algorithms follow a similar basic principle to $\mathcal{A}$ with a modified waiting criterion if only one batch can be released. Poon and Yu [24] stated that every algorithm which is based on the same principle of H^B and MH^B is 2-competitive. For a special case, in which the number of batches which can be processed in parallel is 2, an algorithm with competitive ratio of 7/4 exists. In the OBPP the processing time of a batch is only determined by the longest single processing time of an assigned job. Since the service time of a batch in the OOBP is at least as long as the longest single service time of the assigned customer orders, the OBPP is a special case of the OOBP. Therefore, algorithms for the OBPP with a competitive ratio of at most 2 have to exist. For the ODARP the IGNORE strategy [25,26] and the REPLAN strategy [25] are 5/2-competitive. It should be noted that due to the different problems the REPLAN strategy for the Dial-a-Ride Problem is different to the REPLAN strategy for the OOBP ($\mathcal{A}$ with $\alpha=0$). For the ODARP the SMARTSTART strategy, in which idle times during unserved requests exist are possible, results in a competitive ratio of 2. Also, $\mathcal{A}$, which allows for those idle times, has the best possible competitive ratio.

6. Purpose and design of the numerical experiments

6.1. Purpose

In order to determine the solution quality of the described algorithms, an extensive series of numerical experiments has been carried out. A variety of problem classes have been considered and the behavior of the suggested approaches was simulated. The experiments aim at investigating which value for α should be chosen in order to minimize the maximal completion time and which selection rule should be combined with a particular batching strategy. Furthermore, the solution quality of the proposed algorithms is compared to the solution quality of benchmark heuristics.

6.2. Warehouse parameters

In the experiments a single-block warehouse with two cross-aisles, one in the front and one in the back of the picking area, is assumed. This layout type (cf. Fig. 2) has been used frequently in experiments described in the literature [8,17]. The picking area consists of 900 storage locations, where a different article has been assigned to each storage location. The storage locations are arranged into 10 aisles (U=10) with 90 storage locations each (45 cells on both sides of an aisle, i.e. C=45). The aisles are numbered from 1 to 10; aisle no. 1 is the leftmost aisle and aisle no. 10 the rightmost. Each cell has a width of one length unit (LU) ($l_{cw}=1$) and the center-to-center distance between two aisles is 5 LU (l_c=5). The depot is 1.5 LU away from the first storage location of the leftmost aisle and the distance between the front cross-aisle and the depot amounts to 0.5 LU. It is further assumed that an order picker walks 10 length units in 30 s and he/she needs 10 s to search and collect an item from a storage location, i.e. $v_{\text{travel}} = 48$ LU/min and $v_{\text{pick}} = 6$ items/min. The setup time t_{setup} amounts to 3 min.

6.3. Problem classes

To analyze the quality of the proposed algorithms several problem parameters are varied. For the demand structure two options are considered. At first, a *class-based storage assignment* (C) is assumed: the articles are grouped into three classes A, B, and C according to their expected demand frequency. Class A contains articles with high, B with medium and C with low demand frequency. Articles of class A are only stored – near to the depot – in aisle no. 1, articles of B in the aisles no. 2, no. 3, and no. 4., and articles of class C in the remaining six aisles. It is assumed that 52% of the demanded articles belong to articles in class A, 36% to articles in B and 12% to articles in C. Within a class, the location of an article is determined randomly. These assumptions are typical for real world problems and in line with those of de Koster et al. [14]. Secondly, an *uniform storage assignment* (U) is considered, i.e. for each article the probability that it is included in a customer order is identical.

For the capacity W two different values are assumed, namely 45 and 75 items. For a customer order the quantity of items is uniformly distributed in {5, 6,...,25}, resulting in 3 or 5 customer orders per batch on average, in accordance with the above defined capacities of the picking device.

For the total number of customer orders n the values 30, 60, 90 and 120 are considered. The customer orders should arrive within a planning period of 8 h. The *interarrival times* – the time between the arrival of customer order i and customer order $i+1$ – are exponentially distributed with the *arrival rate* λ. Let $X(t)$ be the number of incoming customer orders in the time interval [0,t]. The stochastic process $\{X(t)|t \geq 0\}$ is called *arrival process*. In case of exponentially distributed interarrival times $E[X(t)] = \lambda \cdot t$ holds. λ is chosen in a way that $E[X(t)]$ is equal to n for t=8[h]. In summary, the following values are used: for n=30: $\lambda = 0.08625$, for n=60: $\lambda = 0.125$, for n=90: $\lambda = 0.1875$, and for n=120: $\lambda = 0.25$.

Chew and Tang [7] provide the expected travel time related to a batch containing k items, when the routes are determined by means of the S-Shape heuristic. In order to determine the service time st_i of a single customer order i a simplified version of this expression is used

$$E[st_i] \approx \frac{l_a}{v_{\text{travel}}}\left(U - \sum_{u=1}^{U}(1-p_u)^k + \frac{1}{2}\right) + \frac{2l_c}{v_{\text{travel}}}\left(U - \sum_{u=1}^{U-1}\left(\sum_{r=1}^{u} p_r\right)^k\right) + \frac{k}{v_{\text{pick}}} + t_{\text{setup}} \tag{8}$$

where p_u ($u \in \{1,\dots,U\}$) is the probability that an item is picked in aisle u. For the class-based storage assignment the expected travel time amounts to 12.5 min for a customer order with 15 items. If each customer order is assigned to a single batch, 48 customer orders can be processed within 8 h. The different values for n lead to the following situations: for n=30 all customer orders can be processed within the 8 h and the order picker can process each open customer order in a single tour. During this tour new customer orders arrive. In problem classes where n is greater than 30, customer orders arrive faster than the order picker can process the open orders (the system is overloaded). Consequently, the number of open orders increases till the last customer order has arrived. Afterwards the number of open orders decreases. Since the order picking system does not operate day and night, enough time resources are available in order to process this pool. Fig. 3 shows the number of open orders (#$P(t)$) during the 8 h for different n. The values on the y-axis have been determined in the experiments.

Combination of the test parameters described above leads to 32 problem classes. For each problem class 50 instances have been generated, which provided 1600 instances in total.

6.4. Benchmarks

As first benchmarks serve applications of the IGNORE strategy, in which the respective batches are determined by FCFS, C&W(ii) and ILS. As further benchmark the best-performing seed algorithm [12] is used. Whenever the order picker is idle and unprocessed customer orders are known, exactly one batch is determined as follows. An order is selected as seed order which possesses the smallest number of aisles the order picker has to visit in order to collect the items of this order. To this batch orders with minimal average mutual-nearest-aisle distance are added until the capacity of the picking device is reached or no further orders are available. This distance is determined as follows: at first, the sum of the distances on the cross-aisle from each pick location of the order to the respective closest pick locations of the batch is divided by the number of items in the order. Secondly, the sum of the distances on the cross-aisle from each pick location of the batch to the respective closest pick location of the order is divided by the number of items in the batch. The mean of these two values gives the average mutual nearest-aisle distance. Then, the order picker starts to process this batch immediately.

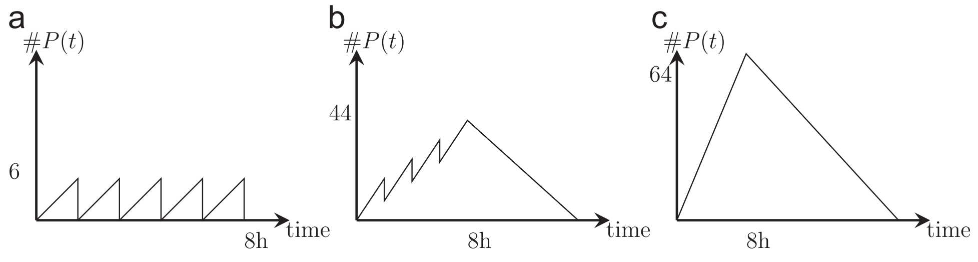

Fig. 3. Number of open orders for different numbers of customer orders. (a) n=30. (b) n=90. (c) n=120.

Table 1
Average maximum completion time in minutes for S-Shape routing, a class-based storage policy and $n=30$.

W	α	$\mathcal{A}$ **with**											
		FCFS and				**C&W(ii) and**				**ILS and**			
		FIRST	SHORT	LONG	SAV	FIRST	SHORT	LONG	SAV	FIRST	SHORT	LONG	SAV
45	0.00	473	473	473	473	473	473	473	473	473	473	473	473
	0.25	480	485	481	481	481	486	479	479	480	486	479	479
	0.50	480	488	480	480	482	488	480	480	480	488	479	479
	0.75	480	488	481	480	481	489	480	480	479	489	479	478
	1.00	480	489	480	480	481	489	479	480	479	489	478	478
75	0.00	473	473	473	473	472	472	472	472	472	472	472	472
	0.25	481	486	481	481	481	487	481	481	481	486	481	481
	0.50	483	490	483	483	482	491	481	481	481	491	481	481
	0.75	482	493	482	482	482	492	480	479	479	491	479	479
	1.00	482	494	482	482	482	491	480	480	481	493	480	480

6.5. *Algorithm parameters and implementation*

For the experiments the parameters of the algorithms have to be specified. Five different values for α (0, 0.25, 0.5, 0.75, 1) are investigated. The topic of this paper is a real-time problem. Therefore, the fast generation of solutions is an essential requirement for each solution approach. Since the point in time when the order picker returns to the depot is known at the start time of a batch, the time interval between start time and completion time can be used to generate solutions. Since orders, which should be considered as well, may arrive during this time interval, a limitation of the computing time is necessary. The computing times required by FCFS and C&W(ii) to generate a set of batches and the computing times needed by the selection rules and the seed algorithm in order to identify the next batch are – for those problem sizes which occur during the experiments – smaller than a second. Whereas these times can be neglected, the computing time for each call of ILS has to be restricted. ILS terminates after 100 perturbations without improvements of the best-found objective function value but not later than 1 min computing time. The rearrangement factor θ is set to 0.3, the threshold factor μ to 0.05, and the time interval $t_{\text{incumbent}}$ to 0.2 min.

The computations for all 1600 instances have been carried out on a Pentium processor with 2.21 GHz and 2.0 GB RAM. The algorithms have been implemented with C++ using the DEV Compiler Version 4.9.9.2.

7. Results of the experiments

7.1. *Outline*

In this section the results of the numerical experiments are presented. In configuration tests the value for α is determined for which the best maximal completion times are obtained. Afterwards, the numerical results are depicted with respect to the routing strategies S-Shape and Largest Gap. For each routing strategy the impact of the selection rules for a particular batching method is compared. Tables 2 and 5 depict the average maximum completion times provided by the suggested combinations of $\mathcal{A}$. The first column in each table describes the problem class, represented by 'storage policy/total number of customer orders/capacity of the picking device in number of items' (C or U/n/W). The entries in the other columns show the results of $\mathcal{A}$ combined with the batching heuristics FCFS, C&W(ii), and ILS and the selection rules: FIRST, SHORT, LONG, and SAV. The best average value generated by a selection rule for the application of a particular batching heuristic is highlighted bold. Afterwards, the solution quality of the different batching strategies combined with the corresponding selection rule, which leads to the best results, is compared to the solution quality of the benchmark heuristics (cf. Tables 3 and 6). Additionally, Tables 4 and 7 contain the average turnover times of a customer order for different problem classes.

7.2. *Configuration tests*

From the configuration tests it can be concluded that for $\mathcal{A}$ the value $\alpha=0$ leads to smaller maximal completion times than for other values of α in each problem class. Table 1 depicts the obtained maximal completion times for the different choices of α for a class-based storage policy and 30 customer orders. The results obtained for the different values for α in the other problem classes are very similar and omitted here. For the problem classes, in which the workload of the system is small ($n=30$), the interarrival times between two customer orders are longer than the time period that the order picker needs to process a customer order on average. Therefore, postponing the release of a batch is not necessary in order to minimize the maximal completion time. Nevertheless, in the problem classes with a small number of customer orders an increasing α reduces the number of released batches. Since $\alpha=0$ also obtains the shortest maximal completion times for medium-sized problem classes, it can be concluded that variants of $\mathcal{A}$ with a waiting time ($\alpha>0$) schedule the batches too late. Also batches in which the remaining capacity is small are not released directly, but the next customer order it is waited for, for which a second batch is necessary. This effect is also validated considering the average number of how often the selection rules are called, i.e. the situation in which more than one batch can be released. Whereas for $\alpha=0$ this value is small (for $n=30$, $W=45$ and the combination FCFS/FIRST: 1; $n=60$, $W=45$ and FCFS/FIRST: 9), the number is significantly larger for $\alpha=0.25$ ($n=30$, $W=45$, FCFS/FIRST: 6; $n=60$, $W=45$ and FCFS/FIRST: 18). For the large problem classes ($n=90; n=120$) the system is overloaded and, therefore, at each decision point more than one batch can be released. In this case all choices of α result in the same maximal completion time, since situations in which only one single batch is available do not exist.

Similar to the number a selection rule is called, the average number of customer orders at a decision point depends on the characteristics of the problem classes (mainly n). For example, the problem class (C/./45) and the combination C&W(ii)/LONG are considered. Here, the average numbers of customer orders at a decision point amount to 1.0 ($n=30$), 3.7 ($n=60$), 13.6 ($n=90$) and 22.2 ($n=120$). The corresponding largest numbers of open customer orders at a decision point are 6 ($n=30$), 18 ($n=60$), 44 ($n=90$) and 64 ($n=120$).

7.3. S-Shape routing

Applying $\mathcal{A}$ in combination with the batching heuristic FCFS, LONG obtains the shortest maximum completion times for each problem class on average. The results provided by LONG and SAV are nearly identical. They differ only for problem classes with uniformly distributed demand and large n. Whereas FIRST provides similar maximum completion times for small problem classes ($n=30$, 60), the deviation to those obtained by LONG increases for larger problem classes. By the application of SHORT the maximum completion times are significantly longer than those generated by LONG.

If the batching heuristic C&W(ii) is used in $\mathcal{A}$, SAV and LONG provide the best results for 14 and 11 problem classes, respectively. Application of FIRST also results in similar maximum completion times. Larger deviations can only be observed in problem classes with 90 and 120 customer orders. The results obtained by SHORT are significantly larger for those problem classes with more than 60 customer orders.

By the application of ILS, SAV leads to the best results for 12 problem classes, LONG for 12, FIRST for 7 problem classes, and SHORT for 3 problem classes. The behavior of the selection rules FIRST, LONG, and SAV is almost identical in terms of the maximum completion time. The differences of the solutions provided by SHORT as compared to the ones provided by SAV are substantially larger for the problem classes with $n \geq 90$.

To sum up, the results of the different selection rules for a particular batching heuristic are almost identical for small n. Due to the fact that more than 30 customer orders can be processed during the eight hours, the number of generated batches is small at each decision point. Therefore, the selection rules determine identical batches, which are released. Also, in case of a higher capacity of the picking device the number of available batches at a specific decision point is smaller than in the case of a small capacity. Thus, the different selection rules release identical batches in problem classes for which the capacity amounts to 75 items.

For larger n the differences of the results provided by the selection rules for a particular batching strategy increase. The deviations observed in the uniformly distributed demand structure are larger than in the class-based case, since the customer orders are more homogeneous in the class-based case than in the uniform case. For all batching heuristics, application of SHORT results in the longest maximal completion times. SHORT tends to select batches with a small capacity utilization, since a small number of items in a batch results in a short service time. As a consequence, in case of SHORT the number of released batches is larger compared to the remaining selection rules, which results in a longer maximum completion time.

Comparing the results of FCFS/LONG (the batching heuristic FCFS in combination with the best selection rule LONG) to the results obtained by C&W(ii)/LONG, the latter outperforms the first one for large problem classes, while the results are very similar for the classes with $n=30$ and 60 and the problem classes with $n=90$ and $W=75$. In the remaining problem classes the difference amounts to more than 30 min ($n=90/W=45$), 64 min (120/45) and 19 min (120/75). In total, it can be concluded that the difference increases with a smaller capacity of the picking device, as well as with a larger number of customer orders. If ILS/SAV is compared to C&W(ii)/LONG, the use of ILS is slightly preferred. This advantage is caused in the number of possible solutions which are evaluated. Whereas FCFS and C&W(ii) generate only a single solution, ILS evaluates a large number of possible solutions. In summary, the impact of the batching strategy is negligible for small n. By application of ILS and C&W(ii) the results obtained by the used selection rules do not differ as highly as in the case where batching is done by means of the FCFS rule. A more sophisticated batching heuristic generates more balanced batches (in terms of service time), as well as a smaller number of batches.

In these three variants, the number how often the selection rules are used does not differ for small problem classes (C/30/45: 1; C/30/75: 0; C/60/45: 8; C/60/75: 1; C/90/45: 10). For larger problem sizes the application of ILS/SAV reduces the number of times the selection rules are called in comparison to FCFS/LONG

Table 2
Average maximum completion time in minutes provided by $\mathcal{A}$ for S-Shape routing.

Class	$\mathcal{A}$ with $\alpha=0$ and											
	FCFS and				C&W(ii) and				ILS and			
	FIRST	SHORT	LONG	SAV	FIRST	SHORT	LONG	SAV	FIRST	SHORT	LONG	SAV
C/30/45	**473**	**473**	**473**	**473**	**473**	**473**	**473**	**473**	**473**	**473**	**473**	**473**
C/30/75	**473**	**473**	**473**	**473**	**472**	**472**	**472**	**472**	**472**	**472**	**472**	**472**
C/60/45	504	519	**503**	**503**	502	514	**501**	**501**	502	509	**501**	**501**
C/60/75	**497**	499	**497**	**497**	**497**	498	**497**	**497**	497	497	**496**	**496**
C/90/45	598	656	**588**	590	560	619	558	**556**	555	598	554	**553**
C/90/75	509	555	**508**	**508**	505	538	**502**	**502**	**502**	527	**502**	**502**
C/120/45	787	854	**769**	773	705	784	**704**	**704**	**693**	747	697	695
C/120/75	618	699	**613**	614	584	659	582	**580**	577	625	579	**575**
U/30/45	**481**	482	**481**	**481**	**481**	482	**481**	**481**	**481**	482	**481**	**481**
U/30/75	**479**	**479**	**479**	**479**	**479**	**479**	**479**	**479**	**479**	**479**	**479**	**479**
U/60/45	544	589	**537**	538	535	585	**528**	530	530	573	**526**	528
U/60/75	**508**	521	**508**	**508**	508	519	**507**	**507**	**507**	517	**507**	**507**
U/90/45	744	819	**713**	724	685	778	**674**	679	670	738	**663**	666
U/90/75	548	635	**544**	546	551	633	**534**	537	538	604	**534**	**534**
U/120/45	985	1061	**942**	956	886	997	**878**	883	866	934	**863**	864
U/120/75	700	806	**685**	691	682	787	**666**	672	663	742	658	**657**

(C/90/45: 29 (FCFS/LONG), 26 (C&W(ii)/LONG), 24 (ILS/SAV); C/120/45: 41 (FCFS/LONG), 38 (C&W(ii)/LONG), 27 (ILS/SAV); C/120/75: 22 (FCFS/LONG), 21 (C&W(ii)/LONG), 19 (ILS/SAV)). Comparing the results provided by the different variants of $\mathcal{A}$ to the ones of the benchmark heuristics (Table 3) it can again be observed that differences between the obtained maximum completion times increase when the problem size increases. The differences between the obtained completion times increases not only in terms of absolute values, but also in relative terms. Fig. 4 visualizes the relative development of the solution quality for the problem classes with $W=45$. The results provided by IGNORE: FCFS – providing the worst results – serve as baseline and the improvements to this baseline obtained by the other solution approaches are plotted. Since the system is underloaded for $n=30$, the benchmark heuristics obtain nearly identical results to the variants of $\mathcal{A}$. Among all heuristics $\mathcal{A}$: ILS/SAV provides the shortest maximum completion times. Furthermore, the results obtained by $\mathcal{A}$: C&W(ii)/LONG are smaller than those of IGNORE: ILS. Therefore, it can be concluded that a larger number of updates of the solution reduces the maximal completion times. For $n=120$ a large part of the problem is off-line, since after 8 h no additional customer order arrives and a large number of unprocessed orders exists. Despite this fact, $\mathcal{A}$: ILS/SAV and IGNORE: ILS provide different results. Therefore, it can be concluded that selected batches formed by $\mathcal{A}$: ILS/SAV during the on-line part of the problem have a strong impact on the maximum completion time. $\mathcal{A}$: ILS/SAV and $\mathcal{A}$: C&W(ii)/LONG also outperform the seed algorithm, which can be explained as follows: whereas ILS and C&W(ii) generate for a set of customer orders (simultaneously) a set of batches with respect to its total service time, the seed algorithm generates batches successively.

If – as an alternative criterion – the minimization of the average turnover times of a customer order is chosen the combination $\mathcal{A}$: ILS/SAV obtains the smallest average turnover times (cf. Table 4).

7.4. *Largest Gap routing*

In the problem classes where routing is done by the Largest Gap heuristic, very similar results as in case of S-Shape routing can be observed. In $\mathcal{A}$ for FCFS, C&W(ii) and ILS the selection rule LONG leads to the best results on average (cf. Table 5). The relations of the results provided by the application of the different selection rules and a particular batching heuristic are very similar

Table 3
Average maximum completion times in minutes provided by $\mathcal{A}$ and benchmark heuristics for S-Shape routing.

Class	**$\mathcal{A}$ with $\alpha=0$ and**			**IGNORE**			**Seed**
	FCFS LONG	C&W(ii) LONG	ILS SAV	FCFS	C&W(ii)	ILS	
C/30/45	**473**	**473**	**473**	**473**	**473**	**473**	**473**
C/30/75	473	**472**	**472**	473	473	473	473
C/60/45	503	**501**	**501**	507	504	504	502
C/60/75	508	497	**496**	521	511	509	504
C/90/45	588	558	**553**	612	573	566	564
C/90/75	508	**502**	**502**	521	511	509	504
C/120/45	769	**704**	695	802	722	708	718
C120/75	613	582	**575**	637	598	590	585
U/30/45	**481**	**481**	**481**	482	**481**	**481**	**481**
U/30/75	**479**	**479**	**479**	**479**	**479**	**479**	**479**
U/60/45	537	**528**	**528**	558	546	541	531
U/60/75	508	**507**	**507**	511	510	510	**507**
U/90/45	713	674	**666**	758	701	684	681
U/90/75	544	**534**	**534**	578	561	555	**534**
U/120/45	942	878	**864**	999	902	881	893
U/120/75	685	666	**657**	725	697	684	666

Table 4
Average turnover times in minutes provided by $\mathcal{A}$ for S-Shape routing.

Class	**$\mathcal{A}$ with $\alpha=0$ and**			**IGNORE**			**Seed**
	FCFS LONG	C&W(ii) LONG	ILS SAV	FCFS	C&W(ii)	ILS	
C/30/45	16	16	16	16	16	16	16
C/30/75	16	16	16	16	16	16	16
C/60/45	28	27	26	31	29	29	27
C/60/75	24	24	24	22	22	22	22
C/90/45	82	60	59	91	74	72	62
C/90/75	43	40	40	50	45	44	40
C/120/45	168	149	116	179	141	138	115
C/120/75	90	86	72	103	85	82	71
U/30/45	23	24	23	24	24	24	23
U/30/75	23	23	23	23	23	23	23
U/60/45	50	47	44	61	55	53	46
U/60/75	34	34	34	36	36	35	33
U/90/45	143	132	108	168	140	135	111
U/90/75	64	65	60	81	75	73	59
U/120/45	247	235	189	234	230	230	194
U/120/75	126	130	107	136	131	131	115

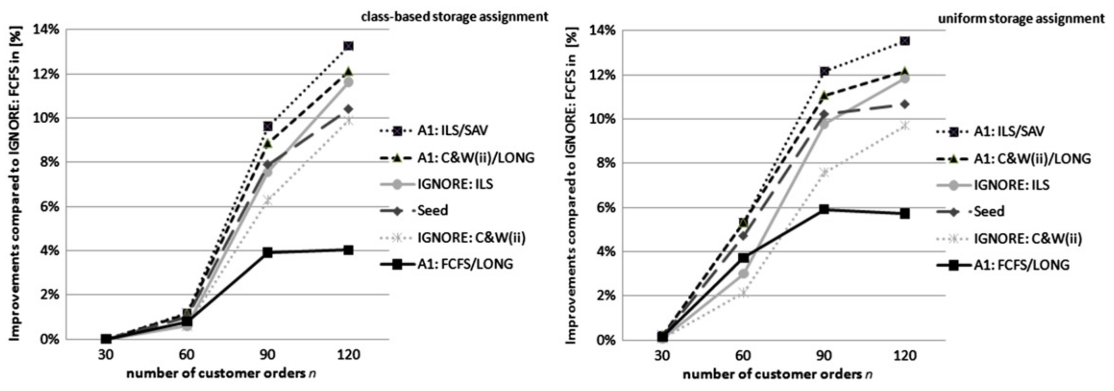

Fig. 4. (Relative) development of the solution quality for the problem classes with $W=45$ and S-Shape routing compared to IGNORE:FCFS.

Table 5
Average maximum completion times in minutes provided by $\mathcal{A}$ for Largest Gap routing.

Class	$\mathcal{A}$ with $\alpha=0$ and											
	FCFS and				C&W(ii) and				ILS and			
	FIRST	SHORT	LONG	SAV	FIRST	SHORT	LONG	SAV	FIRST	SHORT	LONG	SAV
C/30/45	**472**	**472**	**472**	**472**	**472**	**472**	**472**	**472**	**472**	**472**	**472**	**472**
C/30/75	**471**	**471**	**471**	**471**	**471**	**471**	**471**	**471**	**471**	**471**	**471**	**471**
C/60/45	502	514	**501**	**501**	500	506	**499**	**499**	500	505	**499**	**499**
C/60/75	**496**	498	**496**	**496**	**495**	497	**495**	**495**	**495**	496	**495**	**495**
C/90/45	588	645	**577**	580	558	613	**552**	554	551	596	**549**	**549**
C/90/75	**506**	549	**506**	**506**	504	541	**502**	**502**	502	527	501	**500**
C/120/45	773	837	**752**	761	703	779	**700**	701	**691**	746	692	**691**
C/120/75	614	690	**609**	611	589	659	584	**583**	580	631	580	**577**
U/30/45	**478**	479	**478**	**478**	**478**	479	**478**	**478**	**478**	479	**478**	**478**
U/30/75	**477**	**477**	**477**	**477**	**477**	**477**	**477**	**477**	**477**	**477**	**477**	**477**
U/60/45	541	578	**536**	538	534	571	**527**	529	529	561	**526**	527
U/60/75	**508**	520	**508**	**508**	**508**	521	**507**	**507**	**507**	517	**507**	**507**
U/90/45	743	805	**719**	726	687	761	**677**	682	672	726	**667**	670
U/90/75	575	646	**570**	572	569	640	**556**	559	556	609	**550**	**550**
U/120/45	986	1053	**951**	963	890	983	**883**	889	871	928	**867**	870
U/120/75	746	830	**733**	737	714	808	**704**	708	694	758	**689**	**689**

Table 6
Average maximum completion times in minutes provided by $\mathcal{A}$ and benchmark heuristics for Largest Gap routing.

Class	$\mathcal{A}$ with $\alpha=0$ and			IGNORE			Seed
	FCFS LONG	C&W(ii) LONG	ILS LONG	FCFS	C&W(ii)	ILS	
C/30/45	**472**	**472**	**472**	**472**	**472**	**472**	**472**
C/30/75	**471**	**471**	**471**	**471**	**471**	**471**	**471**
C/60/45	501	**499**	**499**	504	502	502	500
C/60/75	496	**495**	**495**	496	496	496	496
C/90/45	577	552	**549**	601	569	563	558
C/90/75	506	502	**501**	519	510	509	503
C/120/45	752	700	**692**	785	717	704	714
C/120/75	609	584	**580**	632	602	593	589
U/30/45	**478**	**478**	**478**	**478**	**478**	**478**	**478**
U/30/75	**477**	**477**	**477**	**477**	**477**	**477**	**477**
U/60/45	536	527	**526**	553	542	538	531
U/60/75	508	**507**	**507**	511	510	509	508
U/90/45	719	677	**667**	754	700	687	689
U/90/75	570	556	**550**	599	581	570	559
U/120/45	951	883	**867**	997	904	886	909
U/120/75	733	704	**689**	766	730	710	711

Table 7
Average turnover times in minutes provided by $\mathcal{A}$ and benchmark heuristics for Largest Gap routing.

Class	$\mathcal{A}$ with $\alpha=0$ and			IGNORE			Seed
	FCFS LONG	C&W(ii) LONG	ILS LONG	FCFS	C&W(ii)	ILS	
C/30/45	15	15	15	15	15	15	15
C/30/75	15	15	15	15	15	15	15
C/60/45	26	26	25	28	27	27	25
C/60/75	23	23	23	23	23	23	23
C/90/45	76	70	67	86	71	70	60
C/90/75	41	40	40	48	44	44	39
C/120/45	157	147	140	170	137	135	113
C/120/75	88	86	81	99	86	83	72
U/30/45	21	21	21	21	21	21	21
U/30/75	21	21	21	21	21	21	21
U/60/45	50	47	45	57	53	51	45
U/60/75	35	35	34	37	36	36	34
U/90/45	146	137	129	164	140	136	116
U/90/75	77	78	72	92	84	80	72
U/120/45	255	242	233	280	234	230	202
U/120/75	150	149	138	168	153	146	138

to the case of S-Shape routing. Comparing the results of the different batching heuristics, the combination $\mathcal{A}$: ILS/LONG outperforms the other variants of $\mathcal{A}$ and the benchmark heuristics (cf. Table 6). Fig. 5 visualizes the improvements for $W=45$ and Largest Gap routing compared to IGNORE: FCFS. The average turnover times for Largest Gap routing are depicted in Table 7.

8. Conclusions and outlook

This paper deals with the on-line variant of the Order Batching Problem, one of the three main planning problems in picker-to-parts warehouses. The problem is to transform customer orders, arriving over time, into picking orders such that the maximum completion time is minimized. Existing methods for the corresponding off-line Order Batching Problem, namely First-Come-First-Served, C&W(ii), and Iterated Local Search have been modified for this on-line problem. By means of a competitive analysis it is shown that the general principle of the proposed algorithm $\mathcal{A}$ is at most 2-competitive in combination with an optimal batching algorithm. Extensive numerical experiments have been carried out to evaluate which selection rule and which batching heuristic in $\mathcal{A}$ lead to the best results. The selection rules LONG and SAV provide the best completion times

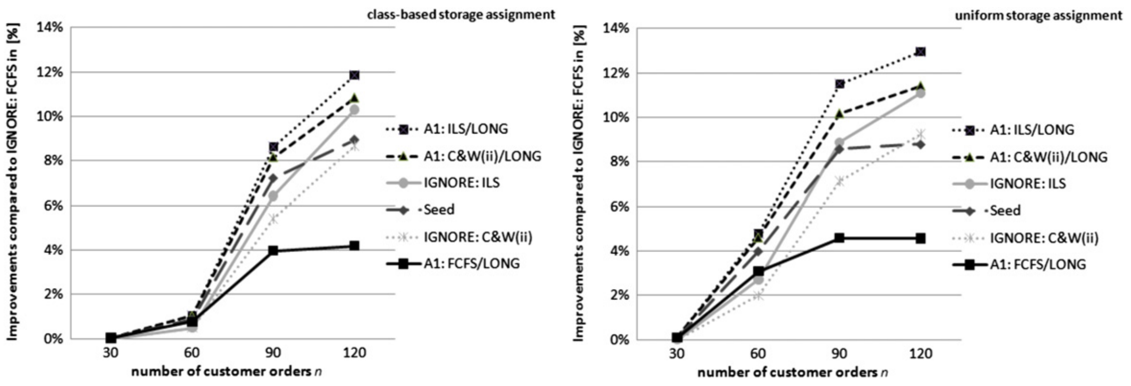

Fig. 5. (Relative) development of the solution quality for the problem classes with $W=45$ and largest gap routing compared to IGNORE:FCFS.

independent of the choice of the batching heuristic. Since ILS provides significantly better results than FCFS and C&W(ii), it is recommended that batching is done as well as possible while the search for an appropriate selection rule is less significant. Compared to benchmark heuristics it can be observed that a frequent update of the solution can reduce maximal completion times significantly.

The presented results are of practical importance with respect to two aspects: firstly, the reduction of the maximal completion time establishes the opportunity to reduce the regular working hours and/or overtime of the order pickers. This will turn out to be a powerful measure for improving the low profit margins still prevalent for warehouse operations. Also, reduced completion times can result in shorter delivery lead times and, therefore, in improved customer services.

For further research it is suggested to investigate the impact of different warehouse layouts (two block warehouses, non standard warehouses, etc.) and other kinds of storage policies. Another research object should be to incorporate due dates in the algorithms, since in many industries customer orders have to be fulfilled at a certain time. For the problem of minimizing average or maximal turnover times competitive results are missing. Analysis for on-line batch scheduling [31] and on-line dial-a-ride problems [32] have shown that there does not exist any strictly competitive algorithm. Therefore, it remains to be shown whether these results are transferable to on-line order batching.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.cor.2011.12.019.

References

[1] de Koster R, Le-Duc T, Roodbergen K. Design and control of warehouse order picking: a literature review. European Journal of Operational Research 2007;182(2):481–501.
[2] Wäscher G. Order picking: a survey of planning problems and methods. In: Dyckhoff H, Lackes R, Reese J, editors. Supply chain management and reverse logistics. Berlin: Springer; 2004. p. 323–47.
[3] Caron F, Marchet G, Perego A. Routing policies and COI-based storage policies in picker-to-part systems. International Journal of Production Research 1998;36(3):713–32.
[4] de Koster R, Roodbergen K, van Voorden R. Reduction of walking time in the distribution center of De Bijenkorf. In: Speranza M, Stähly P, editors. New trends in distribution logistics. Berlin: Springer; 1999. p. 215–34.
[5] Yu M, de Koster R. The impact of order batching and picking area zoning on order picking system performance. European Journal of Operational Research 2009;198(2):480–90.
[6] Ratliff H, Rosenthal A. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Operations Research 1983;31(2):507–21.
[7] Chew E, Tang L. Travel time analysis for general item location assignment in a rectangular warehouse. European Journal of Operational Research 1999;112(3):582–97.
[8] Gademann N, van de Velde S. Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions 2005;37(1):63–75.
[9] Bozer Y, Kile J. Order batching in walk-and-pick order picking systems. International Journal of Production Research 2008;46(7):1887–909.
[10] Gibson D, Sharp G. Order batching procedures. European Journal of Operational Research 1992;58(1):57–67.
[11] Elsayed E. Algorithms for optimal material handling in automatic warehousing systems. International Journal of Production Research 1981;19(5):525–35.
[12] Ho YC, Su TS, Shi ZB. Order-batching methods for an order-picking warehouse with two cross aisles. Computers & Industrial Engineering 2008;55(2):321–47.
[13] Clarke G, Wright J. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 1964;12(4):568–81.
[14] de Koster M, van der Poort E, Wolters M. Efficient orderbatching methods in warehouses. International Journal of Production Research 1999;37(7):1479–504.
[15] Hsu C, Chen K, Chen M. Batching orders in warehouses by minimizing travel distance with genetic algorithms. Computers in Industry 2005;56(2):169–78.
[16] Tsai CY, Liou J, Huang TM. Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. International Journal of Production Research 2008;46(22):6533–55.
[17] Henn S, Koch S, Doerner K, Strauss C, Wäscher G. Metaheuristics for the order batching problem in manual order picking systems. BuR: Business Research 2010;3(1):82–105.
[18] Kamin N. On-line optimization of order picking in an automated warehouse. Aachen: Shaker; 1998.
[19] Won J, Olafsson S. Joint order batching and order picking in warehouse operations. International Journal of Production Research 2005;43(7):1427–42.
[20] Elsayed E, Lee MK. Order processing in automated storage/retrieval systems with due dates. IIE Transactions 1996;28(7):567–77.
[21] van Nieuwenhuyse I, de Koster R. Evaluating order throughput time in 2-block warehouses with time window batching. International Journal of Production Economics 2009;121:654–64.
[22] Le-Duc T, de Koster R. Travel time estimation and order batching in a 2-block warehouse. European Journal of Operational Research 2007;176(1):374–88.
[23] Zhang G, Cai X, Lee CY, Wong C. On-line algorithms for minimizing make span on batch processing machines. Naval Research Logistics 2001;48(3):241–58.
[24] Poon C, Yu W. On-line scheduling algorithms for a batch machine with finite capacity. Journal of Combinatorial Optimization 2005;9:167–86.
[25] Ascheuer N, Krumke S, Rambau J. Online dial-a-ride problems: minimizing the completion time. In: Reichel H, Tison S, editors. Proceedings of the 17th symposium on theoretical aspects of computer science, Lecture notes in computer science, vol. 1770. Berlin: Springer; 2000. p. 639–50.
[26] Feuerstein E, Stougie L. On-line single-server dial-a-ride problems. Theoretical Computer Science 2001;268:91–105.
[27] Hall R. Distance approximations for routing manual pickers in a warehouse. IIE Transactions 1993;25(4):76–87.
[28] Fiat A, Woeginger G. Competitive analysis of algorithms. In: Fiat A, Woeginger G, editors. Online algorithms: the state of the art, Lecture notes in computer science, vol. 1442. Springer; 1998. p. 1–12.

[29] Sleator D, Tarjan R. Amortized efficiency of list update and paging rules. Communications of the ACM 1985;28(2):202–8.

[30] Angelelli E, Speranza G, Savelsbergh M. Competitive analysis for dynamic multiperiod uncapacitated routing problems. Networks 2007;49(4):308–17.

[31] Gfeller B, Peeters L, Weber B, Widmayer P. Online single machine batch scheduling. In: Kralovic R, Urzyczyn P, editors. Mathematical foundations of computer science 2006, 31st international symposium, Lecture notes in computer science, vol. 4162. Springer; 2006. p. 424–35.

[32] Krumke S, de Paepe W, Poensgen D, Lipmann M, Marchetti-Spaccamela A, Stougie L. On minimizing the maximum flow time in the online dial-a-ride problem. In: Erlebach T, Persiano G, editors. Approximation and online algorithms, approximation and online algorithms, third international workshop, WAOA 2005, Lecture notes in computer science, vol. 3879; 2006. p. 258–69.