Algorithms for On-line Order Batching in an Order-Picking Warehouse

S. Henn\textsuperscript{1,*}

\textsuperscript{1} Otto-von-Guericke University Magdeburg – Germany (sebastian.henn@ovgu.de)

\begin{abstract}
In manual order picking systems, order pickers walk or ride through a distribution warehouse in order to collect items required by customers. Order batching consists of combining these – indivisible – customer orders into picking orders. In static (off-line) batching all customer orders are known in advance. In dynamic (on-line) batching customer orders become available dynamically over time. This paper considers an on-line order batching problem in which the total completion time of all customer orders arriving within a certain time period has to be minimized. It is shown how heuristic approaches for the static order batching can be modified in order to deal with the dynamic situation. A competitive analysis shows that every on-line algorithm for this problem is at least 2-competitive. Numerical experiments demonstrate that the choice of an appropriate batching method can lead to a substantial reduction of the completion time of a set of customer orders.

\textbf{Keywords}: Warehouse Management, Order Picking, Order Batching, On-line Optimization
\end{abstract}

1 Introduction

Order Picking is a warehouse function dealing with the retrieval of items (articles) from their storage locations 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 (de Koster et al., 2007). 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 (Wäscher, 2004): 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 pickers. In systems of the first kind, which are considered in this paper, three activities at the operative level can be distinguished (Caron et al., 1998): the assignment of items to storage locations (item location), the transformation of (customer) orders into batches (or 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 batches and jointly released for picking. This activity has proven to be pivotal for the efficiency of warehouse operations (de Koster et al., 1999).

With respect to the availability of the customer orders, two situations for batching orders can occur (Yu and de Koster, 2009): In \textit{off-line (static) batching} all orders are known at the beginning of the (short term) planning period (shift or a day). In \textit{on-line (dynamic) batching} orders become available dynamically over time. Batches have to be formed based only on the known orders. The latter type considers the stochastic arrival process. 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 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 will be defined and an optimization model will be given. Section 3 contains an overview of the relevant literature. Algorithms for the On-line Order Batching Problem 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 completion times, will be performed in Section 5. Moreover, extensive numerical experiments have been carried out to evaluate the performance of the proposed algorithms. Purpose and aim 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 have to travel through the warehouse and collect items from different storage locations. 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 (Ratliff and Rosenthal, 1983), 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 (de Koster et al., 1999). Two well known strategies are the S-Shape and the Largest Gap heuristic which provide the required non confusing routing schemes. Figure 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, 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 an order into two or more batches is prohibited, since it would result in additional unacceptable sorting effort. If an order picker has started a tour through the warehouse, an interruption of this process and a rearrangement of the orders is also not permitted. The time period necessary to complete a batch is called (batch) service time (batch processing time). It 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 setup tasks at the beginning and end of each tour (Chew and Tang, 1999).

In the situation considered here the orders are not known in advance, but become available over time. At a specific point in time, it is only known if at least one orders will arrive in future. However, no information about how many orders or their characteristics will arrive is given. The decision, which orders should be processed directly, has to be made without considering information of future incoming orders. The point in time, when an 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 collecting all items is called completion time of a batch or of an 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 an order. The turnover time (response time) is the time period for which an order stays in the system, i.e. the time period between the completion and the start time of an order. If the number of arriving 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 consists of grouping customer orders into batches such that the completion time of all customer orders — 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

We now formulate an optimization model for the off-line version of the Order Batching Problem described above. The model, which requires the complete information of all incoming orders, is presented in order to analyze the structure of the problem. The following constants are used:

- \( n \): number of orders;
- \( m \): number of batches;
- \( r_i \): arrival time of order \( i \) for all \( i \in \{1, \ldots, n\} \);
- \( \tau \): number of tours.

In this model, the objective is to minimize the completion time of a batch or of an order, i.e. the time period for which an order stays in the system, i.e. the time period between the completion and the start time of an order. The time period necessary to complete a batch is called (batch) service time (batch processing time). It 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 setup tasks at the beginning and end of each tour (Chew and Tang, 1999).

In the situation considered here the orders are not known in advance, but become available over time. At a specific point in time, it is only known if at least one orders will arrive in future. However, no information about how many orders or their characteristics will arrive is given. The decision, which orders should be processed directly, has to be made without considering information of future incoming orders. The point in time, when an 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 collecting all items is called completion time of a batch or of an 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 an order. The turnover time (response time) is the time period for which an order stays in the system, i.e. the time period between the completion and the start time of an order. If the number of arriving 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 consists of grouping customer orders into batches such that the completion time of all customer orders — 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

We now formulate an optimization model for the off-line version of the Order Batching Problem described above. The model, which requires the complete information of all incoming orders, is presented in order to analyze the structure of the problem. The following constants are used:

- \( n \): number of orders;
- \( m \): number of batches;
- \( r_i \): arrival time of order \( i \) for all \( i \in \{1, \ldots, n\} \);
- \( \tau \): number of tours.
In the objective function (1) the expression $d(x_j) = d'_{\text{setup}} + \sum_{i=1}^{n} w_i x_{ji} + \sum_{i=1}^{n} w_i x_{ji} / v_{\text{travel}}$ 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 maximal completion time of all batches with at least one assigned order. Equations (2) ensure the assignment of each order to exactly one batch. Furthermore, inequalities (3) guarantee that the capacity of the picking device is not violated. The conditions (4) indicate that start times are non-negative and the variables $x_j$ are binary vectors, respectively.

\[ t_{\text{setup}}: \text{setup time; } v_{\text{travel}}: \text{travel velocity, i.e. no. of length units the order picker can cover in the warehouse per time unit; } v_{\text{pick}}: \text{pick velocity, i.e. no. of items the order picker can search and pick per time unit; } w_i: \text{no. of items of order } i (\forall i \in \{1, \ldots, n\}; W: \text{maximal no. of items can be included in a batch.} \]

The model uses the following variables.

\[ s_j: \text{start time of batch } j (\forall j \in \{1, \ldots, m\}; x_j: = (x_{j1}, \ldots, x_{jn})^T \text{ a vector in } \mathbb{B}^n \text{ where } \begin{align*} x_{ji} &= 1 \quad \text{if order } i \text{ is assigned to batch } j \\ &= 0 \quad \text{else} \end{align*} \text{ for all } j \in \{1, \ldots, m\}. \]

Without loss of generality, we assume 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 $d_j : \mathbb{B}^n \rightarrow \mathbb{Q}$. For an empty batch the function value is zero, i.e. $d(0) = 0$. The optimization problem can be formulated as follows:

\[
\min \max_{j \in \{1, \ldots, m\}} \left\{ s_j + \frac{d(x_j)}{v_{\text{travel}}} + \sum_{i=1}^{n} w_i x_{ji} \right\} + t_{\text{setup}} \\
\quad \text{[at least one } x_{ji} = 1 \}\]

\[ \text{s.t. } \sum_{j=1}^{m} x_{ji} = 1, \forall i \in \{1, \ldots, n\} \]

\[ \sum_{i=1}^{n} w_i x_{ji} \leq W, \forall j \in \{1, \ldots, m\} \]

\[ s_j \geq \max_{i \in \{1, \ldots, n\}} \{ r_i \cdot x_{ji} \}, \forall j \in \{1, \ldots, m\} \]

\[ s_j \geq s_{j-1} + \frac{d(x_{j-1})}{v_{\text{travel}}} + \sum_{i=1}^{n} w_i x_{ji-1} \frac{1}{v_{\text{pick}}} + t_{\text{setup}}, \forall j \in \{2, \ldots, m\} \]

\[ s_j \geq 0, \forall j \in \{1, \ldots, m\} \]

\[ x_j \in \mathbb{B}^n, \forall j \in \{1, \ldots, m\} \]

In the objective function (1) the expression $d(x_j) / v_{\text{travel}} + \sum_{i=1}^{n} w_i x_{ji} / v_{\text{travel}} + t_{\text{setup}}$ 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 maximal completion time of all batches with at least one assigned order. Equations (2) ensure the assignment of each order to exactly one batch. Furthermore, inequalities (3) guarantee that the capacity of the picking device is not violated. The conditions (4) indicate that a batch is started after all orders assigned to this batch are known. From (5) it follows that a batch is started after the previous one is completed. Finally, the constraints of types (6) and (7) indicate that start times are non-negative and the variables $x_j$ are binary vectors, respectively.

### 3 Literature Review

For off-line order batching Gademann and van de Velde (2005) show the $\mathcal{NP}$-hardness of the problem minimizing the total travel time, if the number of orders per batch is greater than two and present a branch-and-price algorithm with column generation that was able to solve small instances to optimality in reasonable computing time. For the case of S-Shape routing, Bozer and Kile (2008) present 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 orders are ranked according to a priority value and then are assigned to batches following this rank (Gibson and Sharp, 1992), e.g. the First-Come-First-Served rule (FCFS). The second group consists of *seed algorithms*, introduced by Elsayed (1981), which generate batches sequentially. They select one order as a start order for a batch. Additional orders are assigned to that batch according to an order-congruency rule. An overview of the various seed selection and order-congruency rules is given by Ho et al. (2008). Methods of the third group, *savings algorithms*, are based on the Clarke-and-Wright Algorithm for the Vehicle Routing Problem (Clarke and Wright, 1964) and have been adapted in several ways for the Order Batching Problem. For each pair of orders, the savings can be obtained by collecting the items of the two orders one in (large) tour instead of collecting them in two separate tours (de Koster et al., 1999; Elsayed and Unal, 1989). Finally, the last group contains *metaheuristics*. Hsu et al. (2005) present 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. (2008) describe an integrated approach, in which solutions to the batching problem as well as to the routing problem is determined by a Genetic Algorithm. Iterated Local Search and a variant of Ant Colony Optimization are applied to the Order Batching Problem by Henn et al. (2009).

With respect to on-line batching Kamin (1998) describes 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 orders. Besides a theoretical analysis, the system is simulated and evaluated according to different objectives including the time needed to complete all orders of a day. In contrast to our problem, Kamin selects the next batch which should be processed according to a due date, i.e. a point in time when the order is due to be completed.
Apart from this approach, time window batching is prevalent in the dynamic situation (Van Nieuwenhuysen and de Koster, 2009). Time window batching can be carried out in two different variants, namely 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 one waits until a particular number of orders has arrived. These orders are assigned to a batch.

Chew and Tang (1999) describe an on-line problem in which the number of order pickers is limited. They carry 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 simulate 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 in a batch. In variable time window batching one waits until a particular number of orders has arrived. These orders are assigned to a batch.

Elsayed and Lee (1996) describe an automated storage and retrieval system where some articles have to be picked from the warehouse (retrieval orders) and some have to be stored in the warehouse (storage orders). The arrivals of the retrieval 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 of the orders is minimized. It is distinguished between a static and a dynamic case. In the static case, orders arriving in a particular time interval form a group. In the dynamic case, an order arriving while a group of orders is being processed is added to the set of non-processed orders. This results in a new set of batches. In their approach, orders are sequenced according to their due dates and the times needed to process the orders in a single batch. According to this sequence, three decision rules are proposed for the selection of batches: a shortest service time rule, a most common location rule and a nearest schedule rule, which achieves the best results in numerical experiments.

4 Algorithms

4.1 Basic Principle

The difference between on-line and off-line problems consists in the availability of the input parameters. An algorithm for the On-line Order Batching Problem has to form and release batches without having complete information on the types and the arrival times of future 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 classes. A decision of the first kind is one point when a set of unprocessed (also called open) orders exists and an order picker becomes available. Decision points of this type appear at the beginning of the planning period or at the 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. A decision point of the second type is a point in time when an order picker is idle and a new order arrives. The algorithm can rearrange the set of batches and determine a start time for the next batch. Decision points of the third type are the points in time when the last order arrives. At this point the order picker can start all remaining batches one after another.

The basic principle of the proposed on-line algorithm combines ideas of Kamin (1998) for a special warehouse type and of Zhang et al. (2001) for the related on-line scheduling problem of minimizing the makespan on a batch processing machine \((1/r_j, B, \text{on-line/C}_{\text{max}})\). At each decision point \(t\) we determine a solution for the off-line version of the Order Batching Problem for all orders which are known but have not been satisfied at that time. The set \(P(t)\) of these orders is called set of open orders. In contrast to the scheduling problem mentioned by Zhang et al. (2001) the off-line version of the Order Batching Problem is \(NP\)-hard. Therefore, the application of a batching heuristic \(H_b\) is suggested in order to obtain a set of batches \(B(t)\). From this set a batch is selected and released according to a selection rule \(H_s\). An exception would be the case where a solution for batching \(P(t)\) leads only to a single batch \(j'\) with service
Algorithm 1 Basic principle

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>set ( t = 0 ), ( P(0) = { i \in { 1, \ldots, n }</td>
</tr>
<tr>
<td>2:</td>
<td>repeat</td>
</tr>
<tr>
<td>3:</td>
<td>generate ( B(t) ) by means of ( \mathcal{H}_b ) to ( P(t) );</td>
</tr>
<tr>
<td>4:</td>
<td>if (</td>
</tr>
<tr>
<td>5:</td>
<td>set ( t ) to the arrival time of the next order;</td>
</tr>
<tr>
<td>6:</td>
<td>else if (</td>
</tr>
<tr>
<td>7:</td>
<td>let ( j' ) be the batch in ( B(t) );</td>
</tr>
<tr>
<td>8:</td>
<td>( i' = \arg \max { st_i</td>
</tr>
<tr>
<td>9:</td>
<td>if ( t &lt; 2r_{j'} + st_{i'} - st_{j'} ) and the last order is not known then</td>
</tr>
<tr>
<td>10:</td>
<td>set ( t ) to the minimum of ( 2r_{j'} + st_{i'} - st_{j'} ) and the arrival time of the next order;</td>
</tr>
<tr>
<td>11:</td>
<td>else</td>
</tr>
<tr>
<td>12:</td>
<td>start batch ( j' ) and set ( t ) to ( t + st_{j'} );</td>
</tr>
<tr>
<td>13:</td>
<td>end if</td>
</tr>
<tr>
<td>14:</td>
<td>else</td>
</tr>
<tr>
<td>15:</td>
<td>select ( j \in B(t) ) according to selection rule ( \mathcal{H}_a );</td>
</tr>
<tr>
<td>16:</td>
<td>release batch ( j );</td>
</tr>
<tr>
<td>17:</td>
<td>set ( t ) to ( t + st_{j} );</td>
</tr>
<tr>
<td>18:</td>
<td>end if</td>
</tr>
<tr>
<td>19:</td>
<td>update ( P(t) );</td>
</tr>
<tr>
<td>20:</td>
<td>until no further order arrives and ( P(t) ) is empty;</td>
</tr>
</tbody>
</table>

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 pair with largest savings will be combined if it does not violate the capacity constraint. Otherwise, the pair with the second largest savings is considered. This step is repeated until a combination is possible or no pair exists with positive savings which can be combined. In the latter case the algorithm stops. In case that two batches were combined, the savings are computed again, and one searches for a further combination of batches.

Iterated Local Search has been successfully applied to the Order Batching Problem (Henn et al., 2009). It 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. Therefore, 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. For the detailed description we refer to Henn et al. (2009).

4.3 Selection Rules

For the selection rule \( \mathcal{H}_a \), four different options (FIRST, SHORT, LONG, SAV) are suggested. 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 orders have minimal arrival times among all open orders.

The other three selection rules are meant to control the choice of a specific batch in a way that will influence the set of orders and subsequent decisions. SHORT determines a batch with smallest service time. Batches with a small service time may not include 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 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 requiring the longest service time. With this rule the time interval between two batching steps is large and while the batch is processed new orders may arrive which can be combined more favorably with the non-processed orders.

The selection rule SAV computes for each batch a savings value, i.e. the sum of the single service times of the assigned orders and subtracts the total service time of the batch. By selecting the batch with a largest sum, one intends to release a batch with similar orders. In the case of more than one batch with largest savings value, select the batch with lowest index.
5 Competitive Analysis

5.1 Scope

Competitive analysis is a common approach for the evaluation of the performance of an on-line algorithm. It is based on the considerations of Sleator and Tarjan (1985) and analyzes the performance on each input sequence. This is done by comparison of an on-line algorithm with an optimal off-line algorithm, an algorithm that determines an optimal solution for the complete input sequence. More formally, an on-line algorithm $A$ for a minimization problem is called $c$-competitive if a constant $\alpha$ and the selection rules FIRST, LONG or SAV holds, where $A(I)$ is the objective function value provided by algorithm $A$ for instance $I$ and $OPT(I)$ is the objective function value of an optimal off-line solution for $I$. The infimum over all $c$ for which the inequality holds is called competitive ratio.

For the On-line Order Batching Problem we show that the algorithms presented in Section 4.1 are at least 2-competitive for each combination of a batching heuristic with a selection rule. Additionally, we show that this bound is tight for an optimal batching approach, independently of the used selection rule. We assume a single-block warehouse with two cross aisles, one in the front and one in the back of the picking area. The depot is located in front of the leftmost (picking) aisle and all (picking) aisles are vertically oriented. The layout of the warehouse is depicted in Figure 1. The warehouse consists of $U$ aisles with $C$ storage locations (cells) on both sides of an aisle. Being positioned in the center of an aisle, the order picker can pick items from cells on the left, as well as from the 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 $l_{cu}$ in vertical direction from the first storage location, or from the last storage location, respectively. Let $l_i$ denote the center-to-center distance between two aisles, i.e. the distance an order picker has to move in horizontal direction from one aisle to the next aisle.

5.2 Lower Bound

Selection Rules FIRST, LONG, SAV

In order to obtain the proposed lower bound for the competitive ratio, we first describe an example for the case that Algorithm 4.1 is applied with an arbitrary batching heuristic and the selection rules FIRST, LONG or SAV. We assume that the number of storage locations on a side of an aisle $C$ is greater than $\max[2, (U - 1) l_{cu} + 1]$ and that the maximal number $W$ of items in a batch is even. Let two different orders $i_1$ and $i_2$ be available at time 0, of which $i_1$ requires $W/2 + 1$ items stored in the farmost cell on the left of the leftmost aisle. The second order $i_2$ demands $W/2 + 1$ items stored in the first cell of the rightmost aisle. Since both orders cannot be included in the same batch, two batches are needed for completing $i_1$ and $i_2$. The routing of the order picker for both batches according to the S-Shape heuristic and the Largest Gap strategy is identical. Therefore, the batch containing order $i_1$ has the service time of

$$st_{i_1} = \frac{2 \cdot C \cdot l_{cu}}{v_{travel}} + \frac{W}{2} + 1 + t_{setup},$$

and the second one containing $i_2$ has the service time

$$st_{i_2} = \frac{2 \cdot (U - 1) \cdot l_c + 2 \cdot 1 \cdot l_{cu}}{v_{travel}} + \frac{W}{2} + 1 + t_{setup}.$$

According to Algorithm 1, the order picker processes one of these batches immediately. By the application of selection rules FIRST, LONG, and SAV the batch with order $i_1$ will be processed first. Let at time 1 an order $i_3$ become available, i.e. when the order picker has just started to collect the articles of order $i_1$. This order $i_3$ is similar to the first order $i_1$ and requires $W/2 - 1$ items, all of which are located in the leftmost aisle, while one of them is stored in the last cell of the aisle. The on-line algorithm generates a solution in which orders $i_2$ and $i_3$ are assigned to one batch with a completion time of

$$A_1(I) = st_{i_1} + st_{i_2,i_3} = \frac{2(U - 1) \cdot l_c + 2(C + 1) \cdot l_{cu}}{v_{travel}} + \frac{W}{2} + 1 + 2t_{setup}.$$ 

An optimal off-line algorithm generates a solution in which orders $i_2$ and $i_3$ are processed in one batch together. This leads after some rearrangements, which are omitted here, to

$$OPT(I) = \frac{2(U - 1) \cdot l_c + 2(C + 1) \cdot l_{cu}}{v_{travel}} + \frac{W}{2} + 1 + 2t_{setup}.$$ 

In order to provide a lower bound for the competitive ratio we consider after some rearrangements

$$A_1(I) / OPT(I) = 1 + \frac{2 \cdot C \cdot l_{cu}}{v_{travel}} + \frac{W}{2} + 1 + 2t_{setup}.$$ 

From $C \to \infty$ follows that $A_1(I) / OPT(I) \to 2$.

Selection Rule SHORT

Obviously, for the previous instance the selection rule SHORT does not lead to a competitive ratio of $2$. For SHORT we consider another example. At time 0, let two orders be available. Order $i_4$ requires $W - 1$ items stored in cells of the first aisle, including one item located in the
last cell of the first aisle. Order \( i_5 \) requires two items stored in the first cell of the leftmost aisle. The algorithm forms two batches, of which one includes order \( i_4 \) with service time

\[
sl_{i_4} = \frac{2 \cdot C \cdot lw}{v_{travel}} + \frac{W - 1}{v_{pick}} + t_{\text{setup}},
\]

and of which the other includes order \( i_5 \) with processing time

\[
sl_{i_5} = \frac{2 \cdot lw}{v_{travel}} + \frac{W}{v_{pick}} + t_{\text{setup}}.
\]

According to selection rule SHORT, the batch including order \( i_5 \) will be started first. At the completion time of the batch containing \( i_5 \), an order identical to \( i_5 \) is available and is processed next. This step is repeated \( k - 1 \) times where \( k := \min \{ \frac{W}{2}, \lfloor \frac{sl_{i_5}}{lw} \rfloor \} \). Algorithm 1 using SHORT will sequentially process each order identical to \( i_5 \) in a single batch, followed by the batch containing order \( i_4 \). In total, this leads to

\[
A_1(I) = k \cdot sl_{i_5} + sl_{i_4} = \min \{ \frac{W}{2}, \lfloor \frac{sl_{i_5}}{lw} \rfloor \}(\frac{2 \cdot lw}{v_{travel}} + \frac{W - 1}{v_{pick}} + t_{\text{setup}}) + \frac{2 \cdot C \cdot lw}{v_{travel}} + \frac{W}{v_{pick}} + t_{\text{setup}}.
\]

An optimal off-line algorithm releases the batch including \( i_4 \) and then a batch \( j \) containing \( i_5 \) and the \( k - 1 \) orders identical to \( i_5 \). The definition of \( k \) ensures that this second batch does not violate the capacity restriction. Therefore, we have

\[
\text{OPT}(I) = sl_{i_4} + sl_{i_5} = \frac{2 \cdot C \cdot lw}{v_{travel}} + \frac{W - 1}{v_{pick}} + t_{\text{setup}} + \frac{2 \cdot lw}{v_{travel}} + 2 \min \{ \frac{W}{2}, \lfloor \frac{sl_{i_5}}{lw} \rfloor \} \frac{1}{v_{pick}} + t_{\text{setup}}.
\]

For the competitive ratio the following expression can be calculated after few rearrangements

\[
\frac{A_1(I)}{\text{OPT}(I)} = \frac{\min \{ \frac{W}{2}, \lfloor \frac{sl_{i_5}}{lw} \rfloor \} - 1)(\frac{2 \cdot lw}{v_{travel}} + t_{\text{setup}})}{2(C+1)lw} + \frac{W - 1 + 2 \min \{ \frac{W}{2}, \lfloor \frac{sl_{i_5}}{lw} \rfloor \}}{v_{pick}} + 2t_{\text{setup}} + 1.
\]

Additionally to the warehouse assumptions, we assume that \( v_{pick} \to \infty \), \( t_{\text{setup}} = 0 \), and \( W = 2 \cdot C \). In this case, we have

\[
\frac{A_1(I)}{\text{OPT}(I)} = 1 + \frac{\min \{ \frac{W}{2}, \lfloor \frac{sl_{i_5}}{lw} \rfloor \} - 1)(\frac{2 \cdot lw}{v_{travel}})}{2(C+1)lw} = 1 + \frac{C - 1}{C + 1}.
\]

For \( C \to \infty \) we obtain \( A_1(I)/\text{OPT}(I) \to 2 \).

5.3 Upper Bound

It remains to show that an algorithm exists whose competitive ratio is at least 2, independent of the kind of warehouse. In the following the proof of Zhang et al. (2001) for the problem of minimizing the makespan on a single bounded batch processing machine is adapted to the On-line Order Batching Problem. For this proof it is necessary that an optimal batching strategy \( H_{opt} \) is used, i.e., the algorithm obtains an optimal solution for the (off-line) Order Batching Problem. Let \( A_1^* \) be the Algorithm 1 using \( H_{opt} \) and an arbitrary selection rule. Furthermore, let \( A_1^*(I) \) denote the completion time of the latest batch, where \( I \) is an arbitrary problem instance. In this instance the arrival time of the last order is \( r_n \) (for the sake of simplicity it is assumed that exactly one order has the arrival time \( r_n \)). In order to prove that the competitive ratio of \( A_1^* \) is at least 2, we distinguish between the following three cases.

All orders, which have arrived before order \( n \), are already completed before \( r_n \): The described algorithm is an optimal one.

There are unprocessed 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 the algorithm, the last batch \( j' \) includes all open orders without order \( n \) at time \( r_n \) and has the service time \( sl_j \). The batch \( j' \) would be started at \( 2r_{i'} + sl_{i'} + sl_{j'} \) where \( i' \) is an order assigned to batch \( j' \) with the longest single service time \( sl_{i'} \). Since the order picker is idle, \( 2r_{i'} + sl_{i'} + sl_{j'} \) must be greater than \( r_{n} \). Since the batching heuristic is optimal, \( A_1^* \) 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 \( A_1^*(I)_{4,1} \) one obtains:

\[
A_1^*(I) \leq r_n + sl_{j'} + sl_n < 2r_{i'} + sl_{i'} + sl_n
\]

If \( sl_{i'} \geq sl_n \) holds, we can estimate the upper bound for \( A_1^*(I) \) as follows:

\[
A_1^*(I) < 2r_{i'} + sl_{i'} + sl_{i'} = 2(r_{i'} + sl_{i'}) \leq 2\text{OPT}(I).
\]

Otherwise \( sl_{i'} < sl_n \) we obtain

\[
A_1^*(I) < 2r_{i'} + sl_{i'} + sl_n \leq 2r_n + 2sl_n \leq 2\text{OPT}(I).
\]

There are unprocessed orders and the order picker is not idle: Let \( j \) be the batch which is being processed while order \( n \) arrives. Let \( s_j \) be the start time and \( f_j \) be the completion time of \( j \), therefore \( s_j < r_n \leq f_j \). Let \( r \) be the earliest arrival time in the time interval \( (s_j, f_j] \), i.e., \( r = \min \{ r_i | s_j < r_i \leq f_j \} \) and \( A(r) \) be the set of the orders arriving after \( r \), more formal \( A(r) := \{ i | r_i > r \} \).
Let $st_{A(r)}^*$ and $st_{P(s)}^*$ be the total service time of $A(r)$ and the set of open orders $P(s)$ at time $s_j$, respectively. Furthermore, the inequalities $\text{OPT} (\mathcal{I}) \geq r + st_{A(r)}^*$ and $\text{OPT} (\mathcal{I}) \geq st_{P(s)}^*$ hold. Let $\mathcal{I}$ be the set of orders of batch $j$. All orders of $\mathcal{I}$ must be available before $j$ is started and, therefore, $\mathcal{I} \subset P(s_j)$. The solution of the on-line algorithm is $A^*_1(\mathcal{I}) = f_j + st_{A(r)}^* (P(s_j)) \setminus \mathcal{I}$, where $st_{A(r)}^* (P(s_j)) \setminus \mathcal{I}$ is the service time for a set of batches for the orders in $A(r) \cup (P(s_j) \setminus \mathcal{I})$. Since the algorithm uses an optimal batching strategy, $st_{A(r)}^* (P(s_j)) \setminus \mathcal{I}$ is the optimal service time for the orders in $A(r) \cup (P(s_j) \setminus \mathcal{I})$. Let further $st_{P(s)}^* \setminus \mathcal{I}$ be the optimal service time for $P(s) \setminus \mathcal{I}$. Then, $st_{A(r)}^* (P(s_j)) \setminus \mathcal{I} \leq st_{P(s)}^* \setminus \mathcal{I} + st_{A(r)}^*$. For the estimation of the upper bound, we obtain

\[
A^*_1(\mathcal{I}) = f_j + st_{A(r)}^* (P(s_j)) \setminus \mathcal{I} \\
\leq f_j + st_{A(r)}^* \setminus \mathcal{I} + st_{A(r)}^* \\
= s_j + st_{P(s)}^* \setminus \mathcal{I} + st_{A(r)}^* \\
\leq r + st_{P(s)}^* + st_{A(r)}^* \leq 2\text{OPT}(\mathcal{I}).
\]

To summarize, $A^*_1$ has at most the competitive factor 2. Combined with the consideration above, that the competitive factor is at least 2, we can conclude that this bound is tight if an optimal batching heuristic is used.

6 Design of the 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 has been considered and the behavior of the suggested approaches was simulated. The experiments aims at investigating which selection rule should be combined with a particular batching strategy in order to generate a short completion time of all orders. Furthermore, we explore and compare the solution quality of the different batching heuristics.

6.2 Warehouse Parameters

In our 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 has been used frequently in experiments described in the literature (Gademann and van de Velde, 2005; Henn et al., 2009). 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 one. Each cell has a width of one length unit (LU) ($l_{\text{ce}} = 1$) and the center-to-center distance between two aisles amounts to 5 LU ($l_{o} = 5$). 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. We further assume that an order picker walks 10 length units in 30 seconds and he/she needs 10 seconds to search and collect an article from a storage location. This results in a travel velocity of $v_{\text{travel}} = \frac{48\text{[LU]}}{30\text{[s]}}$ and a pick velocity of $v_{\text{pick}} = \frac{6\text{[articles]}}{10\text{[s]}}$. For each tour a setup time $t_{\text{setup}}$ of 3 minutes is needed. We consider a class-based storage assignment, the articles are grouped into three classes A, B, and C according to their expected demand frequency. A contains articles with high, B with medium and C with low demand frequency. Articles of class A are only stored 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. Furthermore, it is assumed that 52 percent of the demanded articles belong to articles in class A, 36 percent to articles in B and 12 percent to articles in C. Within a class, the location of an article is determined randomly.

6.3 Problem Classes

For the capacity of the picking device $W$ we assume two different values, namely 45 and 75 articles. For the routing strategy, the S-Shape heuristic and the Largest-Gap heuristic are used. For an order we choose the quantity of articles uniformly distributed in $\{5, \ldots, 25\}$, resulting in 3 or 5 orders per batch on average, in accordance with the above defined capacities of the picking device. For the total number of orders $n$ we consider 30, 60, 90 and 120. The orders should arrive within a planning period of eight hours. The interarrival times – the time between the arrival of order $i$ and order $i + 1$ – are exponentially distributed with a parameter $\lambda$ called arrival rate. Let $X(t)$ be the number of incoming orders in the time interval $[0, t]$. In the case of exponentially distributed interarrival times $E[X(t)] = \lambda \cdot t$ holds. In our numerical experiments we choose $\lambda$ in a way that the expectation $E[X(t)]$ is equal to $n$ for $t = 8\text{[h]}$. In summary, we use the following values for $\lambda$: for $n = 30$: $\lambda = 0.08625$, for $n = 60$: $\lambda = 0.125$, for $n = 90$: $\lambda = 0.1875$, and for $n = 120$: $\lambda = 0.25$.

In order to determine the service time of a single order we simplify an expression of Chew and Tang (1999) and obtain an expected travel time of 12.49 min for an order with 15 items. If we assume that each order is assigned to a single batch, 48 orders could be processed within eight hours. The different values for $n$ lead to the following situations: For $n = 30$ all orders can be processed within the eight hours and the order picker can process each open order in a single tour. During this tour new orders arrive. In problem classes where $n$ is greater than 30, orders arrive faster than the order picker can process the open orders. Consequently,
the number of open orders increases till the last order has arrived. After that the number of open orders decreases. Figure 2 shows the number of open orders during the eight hours for different \( n \).

Combination of the test parameters described above leads to 16 problem classes. For each problem class 50 instances have been generated, which provided 800 instances in total.

6.4 Algorithms and Implementation

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 becomes available again 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 new orders, which should be considered also, may arrive during this time interval, a limitation of the computing time is necessary. Therefore, we restrict the termination condition in ILS to one minute. The rearrangement factor \( \theta \) is set to 0.3, the threshold factor \( \mu \) to 0.05, and the time interval \( t_{\text{incumbent}} \) to 0.20.

The computations for all 800 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, differentiated with respect to the routing strategies S-Shape and Largest Gap. The tables 1 and 3 depict the average completion time of all orders provided by the algorithms for the problem class. Additionally, the tables 2 and 4 contain the average turnover times of an order for different problem classes. The first column describes the problem class, represented by "total number of orders / capacity of the picking device in number of items" \((n/W)\). The entries in the other columns show the results of Algorithm 4.1 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. The results are analyzed as follows: For each routing strategy we compare the impact of the selection rules for a particular batching method. Therefore, the selection rule, which leads to the best results, serves as benchmark. Afterwards, we evaluate the solution quality of the different batching strategies combined with the corresponding selection rule, which leads to the best results.

7.2 S-Shape-Routing

If orders are grouped by means of the FCFS rule, LONG obtains the smallest completion times for 7 out of 8 problem classes on average. SAV obtains the best objective function value for 5, FIRST for 3, and SHORT for no problem class. The results provided by the selection rules LONG, FIRST and SAV are nearly identical for 30, 60 orders, as well as and for the problem class with 90 orders and a capacity of 75 items. For the remaining problem classes the results obtained by LONG and FIRST amounts to 14 min for the problem class with \( n = 90 \) and \( W = 45 \); for the problem classes with 120 orders the difference amounts to 16 and 7 min. The deviation of the results obtained by LONG and SAV differ by at least 7 min. The application of SHORT the completion times of all orders are significantly larger than those generated by LONG: around 30 min for problem classes with 30 and 60 orders, more than 70 and less than 90 min for problem classes with 90 orders, and more than 90 and less than 106 min for \( n = 120 \).

If C&W(ii) is used, LONG and SAV provide the best results for 7 and 6 problem classes, respectively. The remaining selection rules do not find any best (average) result. The completion times provided by LONG and SAV are very similar. Applying FIRST all orders are completed up to 15 min later than in case of LONG. The deviation between the results of SHORT and LONG amounts to 20 min for \( n = 30 \), to 30 min for \( n = 60 \) and a small capacity, and to 19 min for \( n = 60 \) and a large capacity. By the application
<table>
<thead>
<tr>
<th>Class</th>
<th>FCFS and</th>
<th>Algorithm 4.1 with</th>
<th>ILS and</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FIRST</td>
<td>SHORT</td>
<td>LONG</td>
</tr>
<tr>
<td>30/45</td>
<td>467</td>
<td>488</td>
<td>467</td>
</tr>
<tr>
<td>30/75</td>
<td>457</td>
<td>482</td>
<td>457</td>
</tr>
<tr>
<td>60/45</td>
<td>519</td>
<td>551</td>
<td>518</td>
</tr>
<tr>
<td>60/75</td>
<td>651</td>
<td>722</td>
<td>638</td>
</tr>
<tr>
<td>90/45</td>
<td>499</td>
<td>526</td>
<td>499</td>
</tr>
<tr>
<td>90/75</td>
<td>520</td>
<td>593</td>
<td>519</td>
</tr>
<tr>
<td>120/45</td>
<td>856</td>
<td>930</td>
<td>840</td>
</tr>
<tr>
<td>120/75</td>
<td>634</td>
<td>733</td>
<td>627</td>
</tr>
</tbody>
</table>

**Table 1:** Average completion time of the last batch in minutes for S-Shape-Routing

<table>
<thead>
<tr>
<th>Class</th>
<th>FCFS and</th>
<th>Algorithm 4.1 with</th>
<th>ILS and</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FIRST</td>
<td>SHORT</td>
<td>LONG</td>
</tr>
<tr>
<td>30/45</td>
<td>466</td>
<td>485</td>
<td>466</td>
</tr>
<tr>
<td>30/75</td>
<td>458</td>
<td>482</td>
<td>458</td>
</tr>
<tr>
<td>60/45</td>
<td>514</td>
<td>539</td>
<td>513</td>
</tr>
<tr>
<td>60/75</td>
<td>500</td>
<td>523</td>
<td>500</td>
</tr>
<tr>
<td>90/45</td>
<td>630</td>
<td>690</td>
<td>618</td>
</tr>
<tr>
<td>90/75</td>
<td>525</td>
<td>586</td>
<td>524</td>
</tr>
<tr>
<td>120/45</td>
<td>827</td>
<td>893</td>
<td>813</td>
</tr>
<tr>
<td>120/75</td>
<td>646</td>
<td>731</td>
<td>641</td>
</tr>
</tbody>
</table>

**Table 2:** Average turnover time per order in minutes for S-Shape-Routing

<table>
<thead>
<tr>
<th>Class</th>
<th>FCFS and</th>
<th>Algorithm 4.1 with</th>
<th>ILS and</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FIRST</td>
<td>SHORT</td>
<td>LONG</td>
</tr>
<tr>
<td>30/45</td>
<td>43</td>
<td>53</td>
<td>44</td>
</tr>
<tr>
<td>30/75</td>
<td>58</td>
<td>72</td>
<td>58</td>
</tr>
<tr>
<td>60/45</td>
<td>43</td>
<td>59</td>
<td>43</td>
</tr>
<tr>
<td>60/75</td>
<td>47</td>
<td>61</td>
<td>47</td>
</tr>
<tr>
<td>90/45</td>
<td>102</td>
<td>141</td>
<td>97</td>
</tr>
<tr>
<td>90/75</td>
<td>58</td>
<td>95</td>
<td>57</td>
</tr>
<tr>
<td>120/45</td>
<td>192</td>
<td>239</td>
<td>183</td>
</tr>
<tr>
<td>120/75</td>
<td>107</td>
<td>163</td>
<td>104</td>
</tr>
</tbody>
</table>

**Table 3:** Average completion time of the last batch in minutes for Largest Gap-Routing

**Table 4:** Average turnover time per order in minutes for Largest Gap-Routing
of SHORT all orders for problem classes with 90 orders are completed 89 and 65 min later than by the application of LONG. For 120 orders the deviation between SHORT and LONG amounts to more than 90 min.

By the application of ILS, SAV leads to the best results for 6 problem classes, LONG for 3, FIRST for 2 problem classes. SHORT does again not obtain a best result at all. The behavior of the selection rules FIRST, LONG, and SAV is almost identical in terms of the completion time of all orders. The differences of the solutions provided by SHORT as compared to the ones provided by SAV are significantly larger: approximately 20 min for 30 and 60 orders, more than 40 min for 90 orders, and around 60 min for \( n = 120 \).

The results of the different selection rules are almost identical for \( n \). Due to the fact that, more than 30 orders can be processed during the eight hours, the number of open orders is small during this time. At each decision point the number of generated batches is small. Therefore, the selection rules determine identical batches, which are released. Also, in the case of a larger 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. Therefore, the different selection rules release identical batches for the larger capacity. SHORT tends to select batches with a small capacity utilization, since a small number of items in a batch results in a small service time. As a consequence, in the case of SHORT the number of released batches is larger than in the remaining selection rules, which results in a larger completion time of all orders.

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 30 and 60 orders, and for the problem class with 90 orders and a capacity of 75 items. In the remaining problem classes the difference amounts to 45 min \( (n = 90/W = 45) \), 82 min \( (120/45) \) and 25 min \( (120/75) \). In total, we conclude, that the difference increases with a smaller capacity of the picking device, as well as with a larger \( n \). If ILS/SAV is compared to C&W(ii)/LONG, the use of ILS should be preferred. The results of both batching heuristics are similar for the problem classes with 30 and 60 orders and the problem class with 90 orders and a capacity of 75 items. For the problem class with 90 orders combined with a capacity of 45 items and in the problem class with 120 orders combined with a capacity of 75 items a difference of 17 min can be observed. In the class 120/45 the results even differ by 28 min.

In summary, the impact of the batching strategy is neglectable for a small number of orders. For large \( n \) the application of ILS/SAV can save up to 90 min in comparison to FCFS/LONG. By the application of ILS and C&W(ii) the results obtained by the used selection rules do not differ as large than 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.

If – as an alternative objective function – the minimization of the average turnover times of an order is chosen, slightly different results for the best combination of batching heuristic and selection rule can be identified. As described in Table 2 SAV outperforms the other selection rules. Comparing the batching heuristics, ILS also leads to significantly smaller turnover times than FCFS.

### 7.3 Largest Gap-Routing

In the problem classes where routing is done by the Largest Gap Heuristic, very similar results like in the case of S-Shape-Routing can be observed. For FCFS and C&W(ii) LONG leads again to the best results on average. Only for ILS a different selection rule, namely LONG, generates the best results for Largest Gap-Routing than in the case of S-Shape-Routing. The relations of the results provided by the application of the different selection rules and a particular batching heuristic are very similar to the case of S-Shape-Routing. Comparing the results of the different batching heuristic, ILS/LONG outperforms the other strategies.

### 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 a manual picker-to-parts warehouse. The problem is to transform orders, arriving over time, into batches such that the completion time of all orders 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 this algorithm is 2-competitive in combination with an optimal batching algorithm. The analysis also showed that every on-line algorithm for this problem is at least 2-competitive. Extensive numerical experiments have been carried out to evaluate which heuristic leads to the best results. The selection rules LONG and SAV provide the best completion times 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 good as possible while the search for an appropriate selection rule is less significant. Furthermore, if the turnover time of an order is important, the selection rule SAV should be preferred.

For further research we suggest to investigate the impact of different warehouse layouts (two block warehouses, non standard warehouses etc.) and other kinds of storage policies. An other research object should be to consider due dates.
References


