Henn, Sebastian; Koch, Sören; Doerner, Karl F.; Strauss, Christine; Wäscher, Gerhard

**Article**

# Metaheuristics for the Order Batching Problem in Manual Order Picking Systems

BuR - Business Research

**Provided in Cooperation with:**
VHB - Verband der Hochschullehrer für Betriebswirtschaft, German Academic Association of Business Research

*Suggested Citation:* Henn, Sebastian; Koch, Sören; Doerner, Karl F.; Strauss, Christine; Wäscher, Gerhard (2010) : Metaheuristics for the Order Batching Problem in Manual Order Picking Systems, BuR - Business Research, ISSN 1866-8658, VHB - Verband der Hochschullehrer für Betriebswirtschaft, German Academic Association of Business Research, Göttingen, Vol. 3, Iss. 1, pp. 82-105, https://doi.org/10.1007/BF03342717

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

# *Metaheuristics for the Order Batching Problem in Manual Order Picking Systems*

*Sebastian Henn, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg, E-mail: sebastian.henn@ovgu.de*
*Sören Koch, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg, E-mail: soeren.koch@ovgu.de*
*Karl F. Doerner, Faculty of Business, Economics and Statistics, University of Vienna, E-mail: karl.doerner@univie.ac.at*
*Christine Strauss, Faculty of Business, Economics and Statistics, University of Vienna, E-mail: christine.strauss@univie.ac.at*
*Gerhard Wäscher, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg, E-mail: gerhard.waescher@ovgu.de*

*Abstract*
*In manual order picking systems, order pickers walk or drive through a distribution warehouse in order to collect items which are requested by (internal or external) customers. In order to perform these operations efficiently, it is usually required that customer orders are combined into (more substantial) picking orders of limited size. The Order Batching Problem considered in this paper deals with the question of how a given set of customer orders should be combined such that the total length of all tours is minimized which are necessary to collect all items. The authors introduce two metaheuristic approaches for the solution of this problem: the first one is based on Iterated Local Search; the second on Ant Colony Optimization. In a series of extensive numerical experiments, the newly developed approaches are benchmarked against classic solution methods. It is demonstrated that the proposed methods are not only superior to existing methods but provide solutions which may allow distribution warehouses to be operated significantly more efficiently.*

*Keywords: warehouse management, order picking, order batching, iterated local search, ant colony optimization*

*Manuscript received March 12, 2009, accepted by Sönke Albers March 19, 2010.*

## 1 Introduction

Order picking is a warehouse function dealing with the retrieval of articles (items) from their storage location in order to satisfy a given demand specified by (internal or external) customer orders (Petersen and Schmenner 1999: 481). Order picking arises because incoming articles are received and stored in (large-volume) unit loads while customers order small volumes (less-than-unit loads) of different articles. As a warehouse function, order picking is critical to each supply chain, since underperformance results in an unsatisfactory customer service (long processing and delivery times, incorrect shipments) and high costs (labor cost, cost of additional and/or emergency shipments). The significance of improvements of warehouse operations becomes evident from a joint study of the European Logistics Association and the management consulting company A.T. Kearney (European Logistics Association and A. T. Kearney 2004) which revealed that the total warehousing costs of a company may amount to 25% of its total logistics costs. Of all warehouse operations, order picking is considered to include the most cost-intensive ones. According to Frazelle (2002) up to 50% of the total warehousing operating costs can be attributed to order picking. Drury (1988, also see Tompkins, White, Bozer, Frazelle, and Tanchoco 2003) and Coyle, Bardi, and Langley (1996) give estimations of up to 60% and 65%, respectively, for these costs. The large proportion of order picking (operations) costs originates from the fact that, like many other repetitive material-handling activities, order picking is still a function, which involves the employment of human operators on a large scale. Even though there have been different attempts to automate the picking process, manual order picking systems are still prevalent in practice. Such manual order picking systems can be differentiated into two categories: Picker-to-parts systems, in which

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

order pickers drive or walk through the warehouse and collect the requested articles; and parts-to-picker systems, in which automated storage and retrieval systems deliver the articles to stationary order pickers (Wäscher 2004: 324). With respect to systems of the first kind, three planning problems can be distinguished on the operative level (Caron, Marchet, and Perego 1998: 1), namely the assignment of articles to storage locations (storage location), the grouping of several customer orders into picking orders (order batching), and the routing of order pickers through the warehouse (picker routing). This paper focuses on order batching, which has been proven to be pivotal for the efficiency of warehouse operations (de Koster, Roodbergen, and van Voorden 1999: 232). Improved order batching reduces the (total) lengths of the tours which the order pickers have to cover in order to collect the requested articles. Likewise, the travel time and, consequently, the total picking time (i.e. the total time order pickers spend in the warehouse collecting the articles) will be reduced. According to our experience from practice, this can result in significant savings of labor cost, since it does not only allow for reducing the working time of the pickers, but also for reducing expensive overtime or even for downsizing the (picker) workforce. Since the picking time is also part of the time interval from the moment a customer order is placed until the requested articles are received by the customer, a reduction of the travel time can also immediately be translated into a reduction of the customer order's delivery lead time. In other words, improved order batching also contributes to the advancement of the customer service offered by an order picking warehouse. Both aspects, cost reduction on the one hand, and improvement of customer service on the other, have a positive impact on the competitiveness of the total supply chain.

In the past, for the Order Batching Problem predominantly traditional (constructive) heuristics have been suggested as solution methods. The aim of this article is to demonstrate that modern heuristics can lead to substantially improved solutions in order batching. We focus on two metaheuristics, which have demonstrated to provide excellent results for other combinatorial optimization problems. The first one is a variant of Iterated Local Search; the second one is a population-based approach, namely a variant of ant colony optimization – the Rank-Based Ant System.

The remainder of the paper is organized as follows: In Section 2 the Order Batching Problem will be defined and a mathematical formulation will be given. Section 3 contains a literature review of solution approaches for the Order Batching Problem. In the following two sections our implementation of the metaheuristics will be presented, Iterated Local Search (ILS) in Section 4, and Rank-Based Ant System (RBAS) in Section 5. Extensive numerical experiments have been carried out to evaluate the performance of the metaheuristics. The design of these experiments (including the description of warehouse parameters, algorithm parameters, and problem classes) will be presented in Section 6. A comprehensive analysis of the test results will be given in Section 7. The paper will conclude with a summary and an outlook on further research opportunities in Section 8.

## 2 Order Batching Problem

### 2.1 Problem Description

When performing his/her tasks, an order picker is guided by a so-called pick list. This list specifies the sequence according to which the requested articles should be collected, as well as the quantities in which they are to be picked. A pick list may contain the articles of a single customer order (pick-by-order) or of a combination of customer orders (pick-by-batch). In practice, the sequence in which the articles are to be picked and the corresponding route of the order picker (which starts at the depot, proceeds to the respective storage locations, and returns to the depot) is usually determined by means of a so-called routing strategy, e.g., by the S-Shape Heuristic or by the Largest-Gap Heuristic. 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 seem not to accept the optimal routes provided by the algorithm, because of their not-always-straightforward or sometimes even confusing routing schemes (de Koster, Roodbergen, and van Voorden 1999: 230). The heuristic routing schemes on the other hand, are fast to memorize and quite easy to follow. This helps to reduce the risk of missing an article to be picked – an aspect that might be more important than a small reduction of the tour length. The examples

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

of the routing schemes of the S-Shape Heuristic (Figure 1) and the Largest-Gap Heuristic Figure 2) demonstrate their self-evident character. The black rectangles symbolize the locations of articles to be picked on the respective routes.

The *S-Shape Heuristic* (or traversal strategy) gives a solution in which the order picker only enters an aisle if at least one requested article is located in that aisle and traverses it entirely. Afterwards the order picker proceeds to the next aisle containing a requested article. An exception could be the right-most aisle containing an article to be picked: if the order picker is positioned in the front cross-aisle, he would pick the articles of that right-most aisle and return to the front aisle, i.e. the cross-aisle which contains the depot.

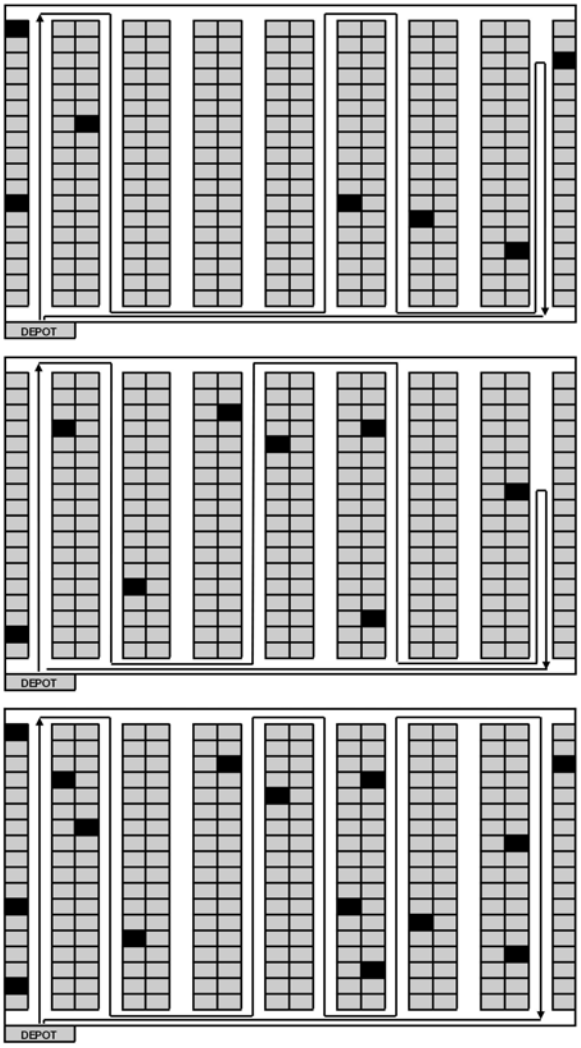
Solutions provided by the *Largest-Gap Heuristic* are characterized in the following way: The order picker traverses the first and last aisle containing an article to be picked entirely. All the other aisles are entered from the front and back in such a way that the non-traversed distance between two adjacent locations of articles to be picked in the aisle is maximal.

Figure 1 shows – for S-Shape Routing – that benefits may arise from collecting the articles of two (or more) customer orders on a single tour instead of two (or more) tours, in particular if the articles are identical or closely located to each other. The upper and middle pictures of Figure 1 depict the tours of two separate customer orders, while the picture at the bottom shows the tour resulting from batching the two customer orders. The length of the resulting tour is obviously smaller than the total length of the two separate tours. The same effect is demonstrated for Largest-Gap Routing in Figure 2.

Order pickers are typically using a picking device (a cart or a roll pallet) for the collection of the requested articles. Depending on the size of the picking device, customer orders can be combined until the capacity of the device is exhausted. On the other hand, splitting of customer orders is usually prohibited since it would result in an additional, not-acceptable sorting effort.

Order batching can be carried out as proximity batching, where orders are combined considering their locations in the warehouse, or as time window batching, where orders are combined according to their arrival time (Choe and Sharp 1991). Here we consider a situation where we assume that all

**Figure 1: Examples of S-Shape Routing**

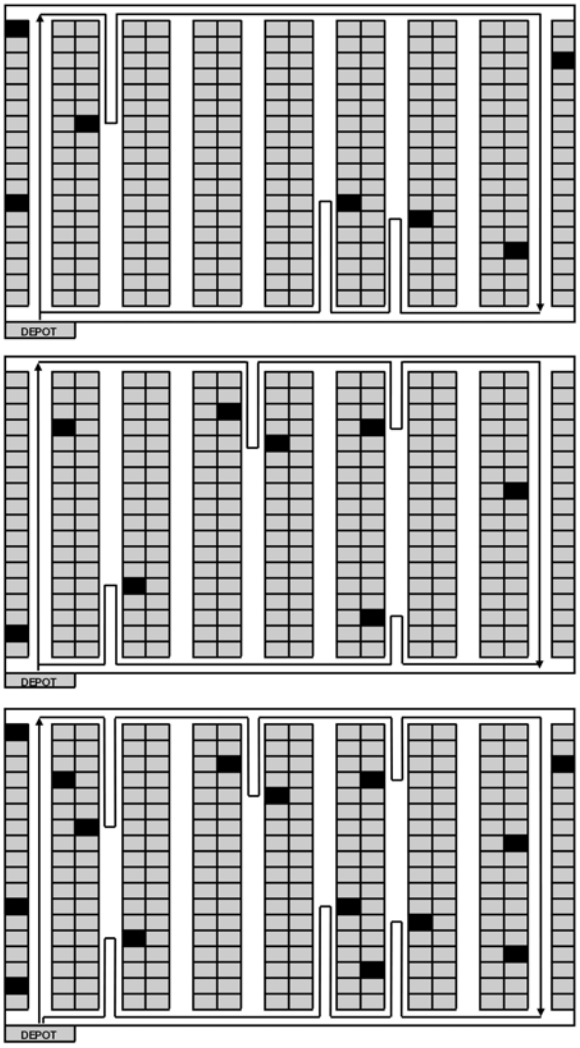



orders are known beforehand. Therefore, we will concentrate on proximity batching.

Due to the high proportion of time-consuming manual operations, order picking is considered to be the most labor-cost-intensive function in a warehouse (Drury 1988). Consequently, the minimization of picking times is of great importance for the efficient control of the picking process. The total order picking time (time spent by order pickers to collect the articles of all customer orders) consists of the setup times for the routes; the travel times that are needed to travel to, from, and between the locations of articles to be picked; the search times for the identification of the articles; and the times needed for picking the articles

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

**Figure 2: Examples of Largest-Gap Routing**




(Tompkins, White, Bozer, Frazelle, and Tanchoco 2003). Among these components, the travel time is of outstanding importance, since it consumes the major proportion of the total order picking time, while the other components can be looked upon to be constant (search times and pick times) or neglectable (setup times). Furthermore, given the pickers' travel velocity to be constant, the minimization of the total travel time is equivalent to the minimization of the total length of all picker tours (Jarvis and McDowell 1991: 94).
In other words, the Order Batching Problem can be defined as follows: How can a given set of customer orders, with given storage locations, given routing strategy and given capacity of the picking devices, be grouped (batched) into picking orders such that the total length of all necessary picking tours is minimized? (Wäscher 2004: 337)

### 2.2 Model Formulation

According to Gademann and van de Velde (2005), a mathematical formulation of the problem can be given as follows: Let $J = \{1, \ldots, n\}$ be the set of customer orders, $C$ be the capacity of the picking device, and $c_j$ the capacity required for order $j$ ($j \in J$). Furthermore, let each batch of customer orders be described by a vector $a_i = (a_{i1}, \ldots, a_{in})$ with binary entries $a_{ij}$ stating whether an order $j$ is included in a batch $i$ ($a_{ij} = 1$) or not ($a_{ij} = 0$). The set $I$ of all feasible batches is characterized by the fact that the capacity of the picking device is not violated, i.e. the following property holds:

$$(1) \quad \sum_{j \in J} c_j \cdot a_{ij} \leq C, \ \forall i \in I.$$

If we define binary decision variables $x_i$ ($i \in I$), which describe if a batch $i$ is chosen ($x_i = 1$) or not ($x_i = 0$), and let $d_i$ further represent the length of the picking tour – subject to a given routing strategy – on which all orders of a batch $i$ are collected, then the following optimization model can be formulated:

$$(2) \quad \min \sum_{i \in I} d_i \cdot x_i$$

$$(3) \quad \text{s.t.} \sum_{i \in I} a_{ij} \cdot x_i = 1, \ \forall j \in J;$$

$$(4) \quad x_i \in \{0, 1\}, \ \forall i \in I.$$

The sets of constraints (3) and (4) ensure that a set of batches is chosen in a way that each customer order is included in exactly one of the chosen batches.
It is important to keep in mind that the number of possible batches and, therefore, the number of binary variables grows exponentially with the number of orders.
The Order Batching Problem as described above is known to be $\mathcal{NP}$-hard (in the strong sense) if the number of orders per batch is greater than two (Gademann and van de Velde 2005).

## 3 Review of Solution Approaches

For the Order Batching Problem only a few approaches have been developed which try to solve

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

the problem to or get close to optimality. For the (general) model described in the previous section, Gademann and van de Velde (2005) present a branch-and-price algorithm with column generation that was able to provide optimal solutions for small instances (up to 32 customer orders). Bozer and Kile (2008) present a mixed-integer programming approach, that generates near optimal solutions for small order sizes (up to 25 customer orders). Their problem formulation is limited to S-Shape Routing since they consider complete traversed aisles in their objective function.

Furthermore, Chen and Wu (2005) describe an order batching approach based on data mining and integer programming. In this approach, first similarities in customer orders are determined by means of an association rule, then a 0-1 integer programming approach is applied in order to cluster the orders into batches. For larger problems – as they usually occur in practice – the use of heuristics is still suggested.

Heuristic solution approaches proposed for the Order Batching Problem are of the constructive type in the first place. They can be distinguished into three groups: priority rule-based algorithms, seed algorithms, and savings algorithms (Wäscher 2004).

*Priority rule-based algorithms* consist of a two-step procedure: In the first step, priorities are assigned to customer orders, which provide the sequence for the second step in which the orders are allocated to batches. Several rules have been described in literature according to which the priorities can be determined. The probably best known and most straightforward way is the application of the First-Come-First-Served (FCFS) rule. Other rules include the application of the two-dimensional and the four-dimensional space-filling curves by Gibson and Sharp (1992) or the six-dimensional space-filling curve by Pan and Liu (1995). The allocation of the customer orders to batches can either be done by means of the Next-Fit Rule (customer orders are added to a batch until the capacity limit is reached; then a new batch is opened), by the First-Fit Rule (all batches are numbered in the order in which they have been opened; the next customer order is allocated to the first batch which provides sufficient capacity), or the Best-Fit Rule (the next customer order is assigned to the batch with the least remaining capacity) (Wäscher 2004).

*Seed algorithms*, introduced by Elsayed (1981) and Elsayed and Stern (1983), generate batches sequentially. Their procedure can be divided into two phases: seed selection phase and order congruency phase. During the seed selection phase, an initial order – the so-called "seed" – is chosen. A large variety of rules is available for the selection of the seed, e.g., choose the first order, choose the largest order, or choose the order with the longest picking tour, and others. Furthermore, the seed can be determined in a single mode (where only the first order in the batch defines the seed) or in cumulative mode (all orders included in the batch define the seed). Afterwards, unassigned customer orders will be added to the seed according to an order-congruency rule, which measures the "distance" from an order not yet allocated to the seed of the batch. Examples of criteria according to which this "distance" can be determined are the number of additional aisles to be visited, the difference between the centers of gravity of the order and the seed, or the sum of the travel distance between every item of the order to the closest item in the seed. An overview of the various seed selection and order congruency rules is given by de Koster, van der Poort, and Wolters (1999), Ho and Tseng (2006) and Ho, Su, and Shi (2008).

*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. In the initial version for the Order Batching Problem, denoted by C&W(i), for each combination of customer orders $i$ and $j$ the savings $sav_{ij}$ are computed which can be obtained by collecting the articles of the two customer orders on one (large) tour instead of collecting them in two separate tours. Starting with the pair of orders with the highest savings, the pairs are considered for being assigned to a batch in a non-ascending order. This may lead to three situations: in case that none of the two orders have been assigned, a new batch will be opened for them; if one of the orders has already been assigned, the other one is added to the batch if the remaining capacity is sufficient; in the case that not enough capacity is available or that both orders have already been assigned, the next pair of orders will be considered. All orders which are left unallocated at the end of the process will be assigned to an individual batch each. The algorithm can be improved by calculating the savings anew each time

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

orders have been allocated to a batch. This variant of the algorithm is denoted by C&W(ii). In detail, the single orders which have been combined into a batch are deleted from further considerations and the new batch will be interpreted as a new "large" order when the savings are determined again. A savings algorithm that generates batches sequentially is the EQUAL algorithm (Elsayed and Unal 1989). In this method, the first pair of orders which has been allocated to a batch is considered as the initial seed. Then the order which – in combination with the seed – leads to the highest savings and does not exceed the capacity of the picking device, is added to the batch. All orders in the batch form the new seed. A new batch is opened if none of the remaining orders fits into the batch. In the Small-and-Large Algorithm (Elsayed and Unal 1989) two subsets are defined, namely a set of large orders and a set of small ones. In a first step, the large orders are assigned to batches according to the EQUAL algorithm described above. The small orders, in a non-ascending order of their size, are then assigned to the batch where they generate the highest savings without violating the remaining capacity. Remaining orders are again assigned to new batches.

Comprehensive numerical experiments (de Koster, van der Poort, and Wolters 1999) have shown that either seed algorithms or savings algorithms provide the best solutions with respect to the total length of all necessary tours, dependent on the warehouse layout and on customer order characteristics.

Apart from the constructive heuristics described above, Hsu, Chen, and Chen (2005) have suggested a genetic algorithm for the Order Batching Problem. Their approach includes an aisle-metric for the determination of the tour lengths and, therefore, is limited to S-Shape Routing, only. Tsai, Liou, and Huang (2008) describe an approach for the integrated solution of both the batching and the routing problem by means of a genetic algorithm. Due to these specific conditions under which they can be applied, both approaches will not be considered here, any further.

Furthermore, Gademann and van de Velde (2005) describe a simple form of an iterated descent algorithm as a part of their branch-and-price procedure.

## 4 Iterated Local Search

Iterated Local Search has been successfully applied to a variety of optimization problems, for instance to the Traveling Salesman Problem (Katayama and Narihisa 1999), Scheduling Problems (Brucker, Hurink, and Werner 1996), or the Quadratic Assignment Problem (Stützle 2006). The main principle can be described as follows (Lorenço, Martin, and Stützle 2003): Let $S$ be the set of feasible solutions of an optimization problem. A solution $s'$ ($s' \in S$) is called a neighbor of solution $s$ ($s \in S$) if it can be obtained by applying a single local transformation to $s$. The heuristic consists of two alternating phases: a *local search phase* and a *perturbation phase*. In the local search phase one starts from an initial solution $s_0 \in S$ and finds a (probably randomized) sequence of feasible solutions $s_0, s_1, \ldots, s_m = \hat{s}$. Each element $s_j$ ($j \in \{1, \ldots, m\}$) is a neighbor of its predecessor, possessing a smaller (minimization problems) or higher (maximization problems) objective function value than the previous one. Provided the problem is bounded, $\hat{s}$ is a local optimum.

Iterated Local Search aims at exploring the vicinity of this local optimum (used as an incumbent solution) more closely in order to identify a solution with an improved objective function value. Therefore, in the perturbation phase, the incumbent solution is partially modified (perturbed) and a further local search phase is applied to this modified solution. The new solution stemming from the local search phase 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. This criterion may allow for the acceptance of deteriorated solutions. These two phases are repeated until a termination condition is met. The challenge consists in choosing an appropriate perturbation scheme: if the perturbation is too marginal, one will often obtain identical solutions after different local search phases; if the perturbation is too extensive, one might unintentionally leave a good subspace of the solution space.

For the Order Batching Problem this general principle has been modified in the following way: An initial solution is generated by means of the FCFS rule. Since all customer orders have the same priority, applying the FCFS rule in this context is equivalent to generating a random sequence of the

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

customer orders, according to which the orders are assigned to batches afterwards.

Two different solutions, i.e. two different sets 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*). For the execution of these local transformations we consider a list of the customer orders which have been assigned to a batch. The sequence in which the customer orders appear on this list is set up in the chronological order in which the customer orders were assigned to the batch, e.g., the first order assigned to the batch is at position one, the second order assigned to the batch is at position two, etc. A *SWAP* is applied in a way that the exchanged orders are taking exactly the same positions in the new batches as their counterparts occupied before. In the case of a *SHIFT* the chosen customer order is assigned to the end of the new batch and is erased from the old batch in a way that all successive customer orders in that batch rise in rank.

In order to determine the objective function value of a neighbor, only the tour lengths of the two modified batches have to be recalculated (delta-evaluation). In the local search phase (Function 1), a systematic first improvement strategy is used: We try to reduce the objective function value by a *SWAP*. If an improvement can be obtained, we proceed from the new solution and search for another improvement by a *SWAP*. In the case that no *SWAP* can be identified which improves the objective function value, we try to achieve an improvement by *SHIFT*s. If no further improvement can be identified, we search again for a *SWAP* that leads to an improved solution. This is repeated until no improvement by *SWAP*s and *SHIFT*s can be identified. The local search phase as described above is a form of a variable neighborhood descent with only two neighborhoods.

The perturbation phase (Function 2) has been designed in the following way: We select two different batches $k$ and $l$ randomly and move the first $q$ orders from batch $k$ to batch $l$, and the first $q$ orders of $l$ 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, i.e. capacity constraints would be violated, remaining orders will be assigned to a new batch. The determination of the number of exchanges is a critical step. If the number is too small, we would probably fall back into the same local optimum. On the other hand, if this number is too high, we would not be able to intensify a good solution subspace. We restrict the number of rearrangements to $n^\star \cdot \lambda + 1$, where $n^\star$ is the number of batches in the best known solution and $\lambda$ (rearrangement parameter) a constant in $[0, 1]$. Regarding the acceptance criterion, a new solution is accepted as an incumbent solution if its objective function value is lower than the best currently known one. Furthermore, we also allow for a few deteriorating steps, i.e. 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 $t$. The solution obtained in the last local search phase will be chosen as the new incumbent solution, if the difference between the length of

**Function 1** local_search($s$)

```
repeat
  repeat
    try to exchange two orders from different
      batches of s in order to reduce the total
      length of all tours (SWAP);
  until no further improvement is possible;
  repeat
    try to assign an order from one batch of s to
      another batch in order to reduce the total
      length of all tours (SHIFT);
  until no further improvement is possible;
until no further improvement is possible;
return s;
```

**Function 2** perturbation($s, \gamma$)

```
for i = 1 to γ do
  choose two batches k and l from s randomly;
  choose q with q > 0 and smaller than half of
    the number of orders in k and l;
  remove the first q orders from k and l;
  insert the removed orders from k into l
    as long as the capacity constraint is not
    violated;
  insert the removed orders from l into k
    as long as the capacity constraint is not
    violated;
  assign the remaining orders to a new batch;
end for
return s;
```

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

all picking tours of the last solution $d(s)$ and the length of all picking tours of the best known solution $d(s^\star)$ is smaller than a threshold $\mu \cdot d(s^\star)$, where the threshold parameter $\mu$ is in $[0, 1]$. Otherwise the incumbent solution will be perturbated again. Algorithm 1 summarizes the heuristic.

**Algorithm 1** Iterated Local Search (ILS)

**Parameters:** rearrangement parameter $\lambda$, threshold parameter $\mu$, time interval $t$ for an incumbent solution;
let $s_{\text{initial}}$ be a solution provided by the FCFS rule;
$s^\star$ := local_search($s_{\text{initial}}$);
$s_{\text{incumbent}} = s^\star$;
**repeat**
  $s$ := perturbation($s_{\text{incumbent}}, n^\star \cdot \lambda + 1$);
  $s$ := local_search($s$);
  **if** $d(s) < d(s^\star)$ **then**
    $s^\star := s$;
    $s_{\text{incumbent}} := s$;
  **end if**
  **if** no improvement of $d(s^\star)$ during time interval $t$ and $d(s) - d(s^\star) < \mu \cdot d(s^\star)$ **then**
    $s_{\text{incumbent}} := s$;
  **end if**
**until** termination condition is met;
$s^\star$ is the solution of ILS;

## 5 Rank-Based Ant System

The general idea of ant colony optimization (Ant Systems) is inspired by nature, where it can be observed that ants easily find the shortest path from a nest to a food source by marking their trails with pheromones. Shorter paths will soon get marked with a higher amount of pheromones, such that following ants will choose those marked trails with a higher probability. In the subsequent course this will lead to a self-reinforcing process, ending in a situation where nearly all ants follow the shortest path.

Artificial Ant Systems for combinatorial optimization problems were first introduced by Colorni, Dorigo, and Maniezzo (1991); Dorigo, Maniezzo, and Colorni (1991) and Dorigo (1992) and have been thoroughly discussed for a multitude of problems, e.g., the Traveling Salesman Problem, Scheduling and Routing Problems (Dorigo and Stützle 2004). Apart from the basic Ant System, several extensions have been developed, e.g., Max-Min Ant Systems (Stützle and Hoos 2000), Ant Systems with elitist strategies or Rank-Based Ant Systems (Bullnheimer, Hartl, and Strauss 1999). In artificial Ant Systems, the pheromone effect taken from nature is combined with another aspect not common for real ants. This is done in order to integrate a greedy behavior into artificial Ant Systems, in addition to the adaptive behavior observed for real ants.

For the adaption to the Order Batching Problem, a savings-based Ant System has been chosen which combines greedy (savings) and adaptive (pheromone) aspects when deciding about the combination of two batches. Savings-based Ant Systems in general work very well for those types of problems where the Clarke-and-Wright Algorithm is a good heuristic (e.g., for the Capacitated Vehicle Routing Problem or, as in our case, for the Order Batching Problem). The Ranked-Based Ant System (RBAS) was implemented since it has proven to provide good results in combination with a savings-based approach (Doerner, Gronalt, Hartl, Reimann, Strauss, and Stummer 2002; Reimann, Doerner, and Hartl 2004).

The general principle of RBAS has been adopted for the Order Batching Problem as follows: In the beginning each order forms a single batch. In the subsequent steps, the batches are combined as long as the capacity constraint for the picking device is not violated. For each possible combination $(k, l)$ of two batches, we compute the corresponding savings $sav_{kl}$ (consult Section 3) and the pheromone intensity $\tilde{\tau}_{kl}$. The pheromone intensity of a batch combination is determined as the sum of pheromones of all pairs of customer orders (called order combinations), where one order is in batch $k$ and one order is in batch $l$, divided by the number $(n_k \cdot n_l)$ of possible order combinations between the batches $k$ and $l$:

$$\tilde{\tau}_{kl} = \frac{\sum_{i \in k} \sum_{j \in l} \tau_{ij}}{n_k \cdot n_l} \tag{5}$$

where

$\tau_{ij}$: pheromone intensity of order combination $(i, j)$,
$n_k$: number of orders assigned to batch $k$,
$n_l$: number of orders assigned to batch $l$.

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

The probability $p_{kl}$ for the combination of two batches $k$ and $l$ is defined as

$$(6) \quad p_{kl} = \frac{(\tilde{\tau}_{kl})^{\alpha} \cdot (sav_{kl})^{\beta}}{\sum_{(u,v)\in\Omega} (\tilde{\tau}_{uv})^{\alpha} \cdot (sav_{uv})^{\beta}},$$

where

$\tilde{\tau}_{kl}$: pheromone intensity of batch combination $(k,l)$,
$\alpha$: parameter for controlling the influence of the pheromone intensity $\tilde{\tau}_{kl}$ ($\alpha \geq 0$),
$sav_{kl}$: savings obtained by combining the batches $k$ and $l$,
$\beta$: parameter for controlling the influence of the savings $sav_{kl}$ ($\beta \geq 0$),
$\Omega$: set of all feasible batch combinations.

If no further feasible combinations of batches can be identified, an attempt is made to improve the obtained solution by applying the aforementioned basic local search function (compare Function 1). The process is repeated for each ant that is used. Each ant represents a solution. Afterwards, the pheromones for all order combinations are updated. First, in analogy to the processes in nature, a fraction $\rho$ of the pheromone evaporates. Likewise good solutions will receive an additional amount of pheromones. More precisely, the pheromone increase is calculated according to the ranking position $r$ within the set of the $m$ best solutions in the actual iteration. In addition, the best solution found so far in all iterations will be rewarded with an additional increase of the pheromone intensity. According to Bullnheimer, Hartl, and Strauss (1999) the pheromone intensity is updated as follows:

$$(7) \quad \tau_{ij} = (1-\rho) \cdot \tau_{ij} + \Delta\tau_{ij},$$

with $\Delta\tau_{ij} := \sum_{r=0}^{m} \Delta\tau_{ij}^{r}$,

$$\Delta\tau_{ij}^{r} = \begin{cases} (m+1-r)\cdot\theta, & \text{if order combination } (i,j) \text{ is in a batch of the } r\text{-th best solution } (r = 1,\ldots,m), \\ (m+1)\cdot\theta, & \text{if order combination } (i,j) \text{ is in a batch of the best solution so far } (r = 0) \\ 0, & \text{otherwise;} \end{cases}$$

where

$\theta$: uprating parameter ($\theta \geq 0$).

The updated pheromone intensities provide the basis for the subsequent iteration. The detailed description of the RBAS used here is given in the pseudo-code of Algorithm 2.

**Algorithm 2** Rank-Based Ant System (RBAS)

**Parameters:** number of iterations, number of ants, initialization of the $\tau_{ij}$, number of best solutions $m$, evaporation parameter $\rho$, uprating parameter $\theta$, control parameter $\alpha$, $\beta$;
**for** $it = 0$ to number of iterations **do**
  **for** $a = 0$ to number of ants **do**
    create start solution $s$ consisting of a single customer order per batch;
    **repeat**
      **for all** feasible pairs $(k,l)$ of $s$ **do**
        compute $sav_{kl}$;
        determine $\tilde{\tau}_{kl}$;
      **end for**
      determine $p_{kl}$;
      choose combination $(\tilde{k},\tilde{l})$ according to (6);
      update $s$ by combining $\tilde{k}$ and $\tilde{l}$;
    **until** no further combination is possible;
    $s$ :=local_search($s$);
    update the best $m$ solutions $(s_1^{\star},\ldots,s_m^{\star})$ of this iteration;
    update the best solution $s_0^{\star}$ found so far;
  **end for**
  **for all** order combinations $(i,j)$ **do**
    $\tau_{ij} := (1-\rho)\cdot\tau_{ij}$;
  **end for**
  **for** $r = 0$ to $m$ **do**
    **for all** batches $k$ in $s_r^{\star}$ **do**
      **for all** order combinations $(i,j)$ in $k$ **do**
        $\tau_{ij} := \tau_{ij} + (m+1-r)\cdot\theta$;
      **end for**
    **end for**
  **end for**
**end for**
$s_0^{\star}$ is the heuristic solution;

## 6 Design of the Experiments

### 6.1 Warehouse Parameters

In order to evaluate the performance of the proposed heuristics in our numerical experiments we assume a single-block warehouse with two cross

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

aisles, one in the front and one in the back of the picking area. All aisles are vertically orientated and the depot is located in front of the leftmost aisle. This type of layout was depicted in Figure 1 and Figure 2 already and is identical to one which is frequently used in the literature (compare de Koster, van der Poort, and Wolters 1999, Petersen and Schmenner 1999). The picking area consists of 900 storage locations (cells) and we assume that a different article has been assigned to each storage location. The storage locations are partitioned into 10 picking aisles with 90 storage locations each, i.e. 45 cells on both sides of each aisle. The aisles are numbered from 1 to 10, where aisle no. 1 is the left-most aisle and aisle no. 10 the right-most aisle. Within an aisle two-sided picking is assumed, i.e. being positioned in the center of an aisle, the order picker can pick items from cells on the right as well as from the cells on the left without additional movements. The length of each cell amounts to one length unit (LU). When picking an article, the order picker is positioned in the middle of the cell. Whenever the order picker leaves an aisle, he/she has to move one LU in vertical direction from the first storage location, or from the last storage location respectively, in order to reach the cross aisle. For a transition into the next aisle the order picker has to move 5 LU in horizontal direction, i.e. the center-to-center distance between two aisles amounts to 5 LU. The depot is 1.5 LU away from the first storage location of the left-most aisle, i.e. the distance between cross aisle and depot amounts to 0.5 LU. We assume a class-based storage assignment of articles to storage locations, i.e. the articles are grouped into three classes A, B and C by their demand frequency, where 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 class B in aisles no. 2, no. 3 and no. 4, and articles of class C only in the remaining six aisles. Furthermore, we assume that 52% of the demand belongs to articles in class A, 36% to articles in B and 12% to articles in C. Within a class, the location of each article is determined randomly. Table 1 summarizes all warehouse parameters fixed for the numerical experiments.

**Table 1: Warehouse Parameters**

| | |
|---|---|
| no. of aisles: | 10 |
| no. of cells on each side of an aisle: | 45 |
| no. of storage locations: | 900 |
| length of a cell [LU]: | 1 |
| center-to-center distance between two aisles [LU]: | 5 |
| distance between depot and front cross aisle [LU]: | 0.5 |
| storage policy: | class-based (ABC-storage) |

### 6.2 Problem Classes

In our numerical experiments we consider five different sizes of customer orders (20, 30, 40, 50, 60), where the number of articles per order is uniformly distributed in $\{5, \ldots, 25\}$. The capacity of the picking device (maximal number of articles that can be assigned to a batch) is set to 30, 45, 60 and 75 articles; in other words, a batch consists of 2, 3, 4 and 5 customer orders on average. The corresponding problem instances can be considered as typical for real-world applications which we are aware of from our own practical experience (fresh-food or deep-freeze warehouses); they are also in line with other real-world examples from the literature (compare de Koster, Roodbergen, and van Voorden 1999). In combination with the two routing strategies (S-Shape, Largest-Gap) we obtain 40 problem classes, and for each of these classes 40 instances are generated. In total, 1,600 instances have been considered, each with one trial per instance. An overview of the problem classes considered in the experiments is given in Table 2.

**Table 2: Problem Classes**

| | |
|---|---|
| total number of orders: | 20, 30, 40, 50, 60 |
| capacity of the picking device [no. of articles]: | 30, 45, 60, 75 |
| routing strategy: | S-Shape, Largest-Gap |

### 6.3 Benchmarks

In order to evaluate the solution quality and the necessary computing times of ILS and the RBAS, the performance of the proposed methods is compared to that of several other batching strategies. Application of the First-Come-First-Served (FCFS) rule represents the simplest way to generate solutions to the Order Batching Problem; it provides an initial upper bound which has been selected as

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

a baseline, here. The solution quality of all other batching strategies is measured in relation to that of FCFS.

From the class of constructive heuristics, the savings algorithm C&W(ii) has been proven to provide excellent results for different kinds of warehouse layouts (de Koster, van der Poort, and Wolters 1999). Thus, C&W(ii) serves as a first benchmark for the suggested methods.

No classic local search method has been reported for the Order Batching Problem in the literature so far. In order to provide a method of this class against which ILS and RBAS can be benchmarked, we combine C&W(ii) with the local search procedure described in Section 4, i.e. starting from the solution provided by C&W(ii), *SWAP*s and *SHIFT*s are interchanged until no further improvement can be obtained. This method will be denoted as C&W(ii)+LS.

As part of their branch-and-price approach, Gademann and van de Velde (2005) proposed an Iterative Descent approach (ID), which, in accordance with our approach, starts from an initial solution generated by means of the FCFS rule. Major differences can be identified with respect to the local search and the perturbation phase. The authors have implemented a first-improvement strategy in which only *SWAP*s are applied; i.e. only one type of neighborhood is considered in their approach. A new solution is taken as a new incumbent solution if the objective function value is better than that of the previous one. The perturbation phase consists of three random exchanges of three customer orders from three different batches. Like in the approach of Gademann and van de Velde (2005), in our experiments the number of calls of the local search and the perturbation phase has been limited to 25, too.

Furthermore, Gademann and van de Velde (2005) use a different definition of the capacity of the picking device, which restricts the number of customer orders – and not the number of articles like in our approach – that can be served simultaneously by the picking device. This might lead to an infeasible solution, when a *SWAP* has to be carried out with two customer orders which consist of a different number of articles. In order to avoid such infeasibilities, their approach has been modified in a way that a new batch is opened. To this batch we assign a randomly selected order from the batch which exceeds the capacity constraint.

In order to evaluate the solution quality of the proposed approaches not only in relative terms (i.e. in relation to the solution quality of other existing methods), we have also tried to benchmark their solutions against the corresponding optimal objective function values. For this purpose, it has been attempted to generate the respective Integer Programming Problem (IPP) – as presented in Section 2 – explicitly for each test problem instance as far as it was possible. An optimal solution to it was generated by means of a commercial LP/IP solver. Not unexpectedly, this approach was successful only for a limited number of problem classes, but gives some insights in the absolute solution quality of the proposed methods, nevertheless. Tables 3 and 4 depict the considered problem classes and the average number of feasible batches (number of columns of the mathematical model) which have been generated. This shows the exponential increase of the problem size by an increasing number of customer orders and an increasing capacity of the picking device. Further, the number of in-

**Table 3: Optimal Solutions for S-Shape Routing**

| *n* | cap. [no. art.] | ∅ no. of batches | no. of opt. sol. | ∅ Gap [%] |
|---|---|---|---|---|
| 20 | 30 | 287 | 40 | - |
| | 45 | 2,240 | 40 | - |
| | 60 | 8,617 | 34 | 4.2 |
| | 75 | 31,884 | 23 | 6.2 |
| 30 | 30 | 871 | 40 | - |
| | 45 | 9,523 | 38 | 2.1 |
| | 60 | 68,993 | 21 | 3.3 |
| | 75 | - | - | - |
| 40 | 30 | 2,436 | 40 | - |
| | 45 | 39,295 | 39 | 1.3 |
| | 60 | 354,256 | 16 | 3.2 |
| | 75 | - | - | - |
| 50 | 30 | 4,211 | 40 | - |
| | 45 | 100,217 | 22 | 1.2 |

*"n": number of customer orders; "cap. [no. art.]": capacity of the picking device in the number of articles; "∅ no. of batches": average number of feasible batches which are generated; "no. of opt. sol.": number of instances per problem class for which the LP/IP was able to generate a solution and prove its optimality- "∅ Gap [%]": average gap between the lower bound and the best objective function value found for those instances the LP/IP solver was not able to prove the optimality.*

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

**Table 4: Optimal Solutions for Largest-Gap Routing**

| $n$ | cap. [no. art.] | ∅ no. of batches | no. of opt. sol. | ∅ Gap [%] |
|---|---|---|---|---|
| 20 | 30 | 323 | 40 | - |
| | 45 | 1,974 | 40 | - |
| | 60 | 10,657 | 40 | - |
| | 75 | 38,277 | 39 | 2.3 |
| 30 | 30 | 963 | 40 | - |
| | 45 | 10,956 | 38 | 1.2 |
| | 60 | 69,339 | 31 | 3.3 |
| | 75 | - | - | - |
| 40 | 30 | 2,731 | 40 | - |
| | 45 | 32,147 | 37 | 0.9 |
| | 60 | 465,547 | 15 | 2.3 |
| | 75 | - | - | |
| 50 | 30 | 4,183 | 40 | - |
| | 45 | 115,113 | 28 | 1.1 |

*For abbreviations see Table 3.*

stances is shown where the LP/IP solver was able to generate a solution and prove its optimality. In the remaining instances the LP/IP solver was only able to generate a solution but was unable to prove its optimality, since the computation violated memory restrictions of the used PC. For these instances the average optimality gap as calculated by the LP/IP solver is shown as well. The optimality gap is the difference between the lower bound (objective function value of the best remaining node) and the best integer objective function value found so far. These results demonstrate that the application of exact approaches is limited to Order Batching Problems of small sizes, whereas for larger sizes the application of heuristics is necessary.

### 6.4 Algorithm Parameters

The parameter settings have been determined in a series of pre-tests: For the RBAS we perform 100 iterations, and the number of ants in each iteration has been set to half the number of customer orders. The pheromone intensity of each order combination has been initialized by 2, and the control parameters $\alpha$ and $\beta$ have been set to 5. The evaporation parameter $\rho$ has been fixed to 0.1 and the uprating parameter $\theta$ to 0.001. During the process of updating the pheromone intensity, we consider the three best solutions in each iteration, i.e. $m = 3$.

In order to provide equivalent conditions for both suggested methods, we allow for an identical computing time, which is determined by RBAS. In other words, each problem instance is solved by RBAS and the resulting computing time is used as the termination condition for ILS.

In ILS the rearrangement parameter $\lambda$ has been set to 0.3. We further allow 10 deteriorations of maximal $\mu = 0.05$ each. In combination with the computing time, this results in $t = t_{RBAS}/10$, where $t_{RBAS}$ is the computing time of RBAS. Table 5 summarizes the parameter choice.

**Table 5: Algorithm Parameters**

| **Rank-Based Ant System** | |
|---|---|
| no. of iterations: | 100 |
| no. of ants: | n/2 |
| initial pheromone intensity $\tau_{ij}$: | 2.0 |
| control parameter $\alpha$: | 5 |
| control parameter $\beta$: | 5 |
| evaporation parameter $\rho$: | 0.1 |
| no. of best local solutions $m$: | 3 |
| uprating parameter $\theta$: | 0.001 |
| **Iterated Local Search** | |
| termination condition: | $t_{RBAS}$ |
| rearrangement parameter $\lambda$: | 0.3 |
| threshold parameter $\mu$: | 0.05 |
| time interval $t$ after which a deterioration is permitted: | $t_{RBAS}$ / 10 |

### 6.5 Implementation and Hardware

The computations for all 1,600 instances have been carried out on a desktop PC with a Pentium processor with 2.21 GHz and 2 GB RAM. The algorithms have been encoded in C++ using the DEV Compiler Version 4.9.9.2. The solver used for the IPP was CPLEX 10.1.

## 7 Results of the Experiments

### 7.1 S-Shape Routing

*Solution Quality*

The results from the numerical experiments for S-Shape Routing are depicted in Table 6.

**Table 6: Solution Quality for S-Shape Routing**

| *n* | cap. | FCFS | | C&W (ii) | | | C&W (ii) + LS | | | ID | | | ILS | | | RBAS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [no. art.] | ⌀ dist. [LU] | ⌀ no. bat. | ⌀ dist. [LU] | ⌀ impr. [%] | ⌀ no. bat. | ⌀ dist. [LU] | ⌀ impr. [%] | ⌀ no. bat. | ⌀ dist. [LU] | ⌀ impr. [%] | ⌀ no. bat. | ⌀ dist. [LU] | ⌀ impr. [%] | ⌀ no. bat. | ⌀ dist. [LU] | ⌀ impr. [%] | ⌀ no. bat. |
| 20 | 30 | 5,172 | 13.2 | 4,360 | 15.82 | 11.3 | 4,337 | 16.28 | 11.3 | 4,778 | 7.81 | 13.2 | **4,192** | 19.13 | 10.7 | **4,192** | 19.13 | 10.7 |
| | 45 | 3,427 | 7.8 | 2,855 | 16.63 | 7.1 | 2,795 | 18.35 | 7.1 | 3,008 | 12.18 | 7.8 | 2,697 | 21.25 | 6.8 | **2,693** | 21.36 | 6.7 |
| | 60 | 2,612 | 5.7 | 2,283 | 11.98 | 5.3 | 2,239 | 13.72 | 5.3 | 2,366 | 9.15 | 5.7 | **2,155** | 16.92 | 5.3 | **2,155** | 16.92 | 5.3 |
| | 75 | 2,076 | 4.6 | 1,878 | 8.97 | 4.3 | 1,826 | 11.59 | 4.3 | 1,942 | 6.29 | 4.6 | 1,747 | 15.37 | 4.2 | **1,744** | 15.49 | 4.2 |
| 30 | 30 | 7,796 | 19.7 | 6,426 | 17.68 | 16.5 | 6,394 | 18.11 | 16.5 | 7,106 | 8.99 | 19.7 | 6,205 | 20.54 | 15.8 | **6,192** | 20.69 | 15.7 |
| | 45 | 5,292 | 11.9 | 4,333 | 18.06 | 10.7 | 4,278 | 19.12 | 10.7 | 4,617 | 12.71 | 11.9 | **4,088** | 22.67 | 10.2 | 4,095 | 22.54 | 10.2 |
| | 60 | 3,928 | 8.6 | 3,288 | 16.13 | 7.8 | 3,209 | 14.91 | 7.8 | 3,485 | 11.22 | 8.6 | **3,083** | 21.40 | 7.6 | 3,090 | 21.21 | 7.6 |
| | 75 | 3,092 | 6.7 | 2,715 | 12.03 | 6.3 | 2,661 | 13.76 | 6.3 | 2,854 | 7.61 | 6.8 | **2,538** | 17.75 | 6.1 | 2,547 | 17.46 | 6.1 |
| 40 | 30 | 10,223 | 25.5 | 8,219 | 19.70 | 21.2 | 8,180 | 20.09 | 21.2 | 9,156 | 10.60 | 25.5 | 7,929 | 22.58 | 20.2 | **7,916** | 22.71 | 20.2 |
| | 45 | 6,883 | 15.4 | 5,504 | 19.98 | 13.8 | 5,418 | 21.25 | 13.7 | 5,929 | 13.81 | 15.4 | **5,211** | 24.26 | 13.2 | 5,224 | 24.07 | 13.2 |
| | 60 | 5,114 | 11.2 | 4,282 | 16.27 | 10.4 | 4,198 | 17.92 | 10.3 | 4,519 | 11.62 | 11.2 | **4,003** | 21.70 | 9.9 | 4,016 | 21.45 | 9.9 |
| | 75 | 4,027 | 8.7 | 3,479 | 13.48 | 8.2 | 3,409 | 15.20 | 8.1 | 3,693 | 8.22 | 8.8 | **3,263** | 18.83 | 8.0 | 3,279 | 18.44 | 8.0 |
| 50 | 30 | 13,075 | 32.8 | 10,509 | 19.63 | 27.0 | 10,481 | 19.85 | 27.0 | 11,816 | 9.72 | 32.8 | 10,141 | 22.49 | 25.8 | **10,125** | 22.62 | 25.8 |
| | 45 | 8,584 | 19.3 | 6,760 | 21.19 | 16.9 | 6,686 | 22.04 | 16.9 | 7,340 | 14.47 | 19.4 | **6,497** | 24.27 | 16.6 | **6,497** | 24.26 | 16.6 |
| | 60 | 6,381 | 13.9 | 5,254 | 17.63 | 12.8 | 5,164 | 19.03 | 12.8 | 5,636 | 11.64 | 14.0 | **5,015** | 21.37 | 12.6 | 5,023 | 21.24 | 12.6 |
| | 75 | 5,097 | 11.0 | 4,325 | 15.09 | 10.3 | 4,237 | 16.82 | 10.3 | 4,676 | 8.24 | 11.2 | **4,081** | 19.90 | 10.1 | 4,110 | 19.33 | 10.1 |
| 60 | 30 | 15,344 | 38.5 | 12,145 | 20.98 | 31.6 | 12,082 | 21.39 | 31.5 | 13,651 | 11.16 | 38.5 | 11,669 | 24.06 | 30.1 | **11,647** | 24.21 | 30.0 |
| | 45 | 10,082 | 22.6 | 7,930 | 21.33 | 20.1 | 7,710 | 23.21 | 20.3 | 8,584 | 10.75 | 22.6 | 7,649 | 24.10 | 19.6 | **7,624** | 24.35 | 19.6 |
| | 60 | 7,642 | 16.6 | 6,147 | 19.55 | 15.1 | 6,062 | 20.66 | 15.1 | 6,656 | 12.91 | 16.7 | **5,904** | 22.73 | 14.7 | **5,904** | 22.73 | 14.8 |
| | 75 | 5,984 | 12.9 | 5,029 | 15.90 | 12.1 | 4,936 | 17.47 | 12.1 | 5,431 | 9.21 | 13.0 | 4,808 | 19.63 | 11.9 | **4,805** | 19.67 | 11.9 |
| Average | | | | | 16.90 | | | 18.05 | | | 10.42 | | | 21.05 | | | 20.99 | |
| Minimum | | | | | 8.97 | | | 11.59 | | | 6.29 | | | 15.37 | | | 15.49 | |
| Maximum | | | | | 21.33 | | | 23.21 | | | 14.47 | | | 24.27 | | | 24.35 | |

*"n": number of customer orders; "cap. [no. art.]": capacity of the picking device in the number of articles; "⌀ dist. [LU]": average total length of picking tours in length units; "⌀ impr. [%]": improvement of the algorithm compared to the FCFS solution in percent; "⌀ no. bat.": average number of batches. For each problem class the best obtained average distance is highlighted bold.*

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

For all problem instances, application of C&W(ii) improved the total length of all picking tours on average by 16.9% (in comparison to the length of the tours resulting from the application of FCFS only). Differentiated with respect to the problem classes, the (average) improvement ranged from 8.97% (no. of customer orders: 20; capacity of the picking device: 75) to 21.33% ($n$ = 60; cap = 45). In general, it can be observed that only small improvements were obtained by C&W(ii) for problem classes characterized by a small number of customer orders. This can be explained by the fact that – for instances from these problem classes – there is just a small probability that the total tour length will be improved since the algorithm opens a new batch for each pair of unassigned orders. We note that our results are in line with those from de Koster, van der Poort, and Wolters (1999), who have run numerical experiments for a similar warehouse and obtained a reduction of the total travel time of the order pickers of 18% (constant travel velocity assumed) for problem instances with 30 customer orders and a capacity of the picking device of 24 articles. Therefore, we conclude that our implementation of C&W(ii) is credible and that the results obtained by C&W(ii) provide a good benchmark.

The addition of a local search phase to C&W(ii) provided even better solutions. C&W(ii) + LS raised the average improvement to 18.05%, ranging from 11.59% ($n$ = 20; cap = 75) to 23.21% ($n$ = 60; cap = 45) for the different problem classes. However, these additional improvements should not be overestimated. The reduction of the total length of all tours achieved by C&W(ii)+LS in comparison to C&W(ii) amounted to only about one percentage point on average. This demonstrates that the solutions provided by C&W(ii) are already close to local minima. Hence, classic local search cannot succeed in improving these solutions significantly. On average, the length of all picking tours provided by the (modified) Iterative Descent method (ID) was by 10.42% shorter than the length of the tours generated by means of FCFS. The results for the different problem classes varied between an increase of only 6.29% ($n$ = 20; cap = 75) and up to 14.47% ($n$ = 50; cap = 45). The results were inferior even to those of the basic savings method C&W(ii), which offers tours that are by 6.48 percentage points shorter on average. This result can be attributed to the fact that ID – as proposed by Gademann and van de Velde (2005) – only uses the *SWAP* operator, which does not allow for reducing the number of batches in the solution. It seems reasonable to conclude that the application of a single type of neighborhood is not a promising strategy for the design of a local search method for the Order Batching Problem.

By application of the newly proposed heuristics ILS and RBAS, further improvements (again, in comparison to FCFS) could be obtained. The reduction of the length of all picking tours amounted to 21.05% for ILS and to 20.99% for RBAS for all problem instances on average, and varied between 15.37% and 24.27% for ILS, and between 15.49% and 24.35% for RBAS, respectively, across the different problem classes. For both, ILS and RBAS, improvements were higher for problem classes with larger capacities of the picking device than for smaller ones.

For 12 of the 20 problem classes, ILS provided the best results on average. Independently from the order size, ILS was superior to RBAS for larger capacities of the picking device (i.e. for cap = 45, 60, 75) combined with medium-size order quantities, whereas RBAS outperformed ILS for a small capacity (cap = 30). It has to be noted, however, that the results for both methods do not differ significantly with respect to solution quality (average total length of all picking tours per problem instance). Even for the class for which the largest difference can be observed ($n$ = 40; cap = 75), this difference (0.39 percentage points) can hardly be considered as practically relevant. Likewise, the difference in the average number of batches was not significant and amounts to 0.1 batches at most. In comparison to C&W(ii), on average, additional improvements of 4.15 percentage points could be achieved by means of ILS, and 4.09 percentage points by means of RBAS, respectively. The additional improvements were as small as 2.77 percentage points (ILS; $n$ = 60; cap = 45) and 2.99 percentage points (RBAS; $n$ = 50; cap = 30), but went up to 6.52 percentage points (for both ILS and RBAS; $n$ = 20; cap = 75). If ILS and RBAS are compared to the combination of savings with local search (C&W(ii)+LS), the additional improvements amounted to 3.00% for ILS on average, and 2.94% for RBAS, respectively. This demonstrates that the solutions obtained by means of the presented metaheuristics were significantly better than those obtained with classic local search.

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

This result can be attributed to the fact that the metaheuristics were able to reduce the number of batches considerably, e.g., up to 1.4 batches for ILS and up to 1.5 for RBAS ($n$ = 60; cap = 30).
Table 7 relates the solution quality of ILS and RBAS to the corresponding optimal objective function values. For all problem classes, ILS and RBAS provided solutions close to the optimum. The average deviation from the optimal objective function value (related to the problem instances in each class which could be solved to a proven optimum) was smaller than 1% (except for the problem classes ($n$ = 40; cap = 60) and ($n$ = 50; cap = 45), where a deviation of up to 2% was observed). Average deviations from the optimal objective function value tended to be smaller for problems with a smaller capacity of the picking device than for those with a larger capacity (with the exception of the problem class with $n$ = 20 and cap = 60). RBAS outperformed ILS for small problem instances ($n$ = 20) and for larger instances with cap = 30. In brackets, Table 7 also depicts the average deviation of the total tour lengths obtained by ILS/RBAS from the total tour lengths provided by the LP/IP solver only for those instances in which the optimality of the solutions could not be proven. For some of the instances, where the LP/IP solver was not able to prove the optimality of the obtained solutions, ILS or RBAS even generated solutions with smaller objective function values. These cases are indicated by negative entries (e.g., $n$ = 20; cap = 75).

**Table 7: Solution Quality for S-Shape Routing**

| $n$ | cap. [no. art.] | ILS dev. [%] | RBAS dev. [%] |
|---|---|---|---|
| 20 | 30 | 0.00 (0.00) | 0.00 (0.00) |
| | 45 | 0.33 (0.33) | 0.18 (0.18) |
| | 60 | 0.05 (0.00) | 0.03 (0.11) |
| | 75 | 0.31 (-0.31) | 0.08 (-0.37) |
| 30 | 30 | 0.27 (0.27) | 0.08 (0.08) |
| | 45 | 0.50 (0.31) | 0.65 (0.92) |
| | 60 | 0.53 (0.09) | 0.92 (0.18) |
| | 75 | - | - |
| 40 | 30 | 0.28 (0.28) | 0.11 (0.11) |
| | 45 | 0.72 (0.04) | 0.99 (0.04) |
| | 60 | 1.27 (0.05) | 2.00 (0.11) |
| | 75 | - | - |
| 50 | 30 | 0.44 (0.44) | 0.28 (0.28) |
| | 45 | 1.82 (0.69) | 1.75 (0.79) |

*"ILS dev. [%], RBAS dev. [%]": average deviation between the objective function value obtained by ILS/RBAS and the objective function value of the optimal solution in percent; in brackets: deviation between the objective function value obtained by ILS/RBAS and by the LP/IP solver only for those instances, for which the LP/IP solver was not able to prove the optimality of the obtained solutions.*

*Computing Times*
Table 8 depicts the average computing times per problem instance for ILS and RBAS. The times for FCFS, C&W(ii), C&W(ii)+LS and ID have not been listed since these methods consumed less than five seconds per instance. The average computing time for RBAS (which – as has been explained before – sets the limit for the computing time allocated to ILS) varied between 7 seconds ($n$ = 20; cap = 30) and 1088 seconds ($n$ = 60; cap = 75). Basically, two problem parameters determined the computing times, namely the size of the problem instances (i.e. the number of customer orders which has to be dealt with) and the capacity of the picking device. Increasing values of these parameters resulted in increasing computing times.
While ILS and RBAS can be considered as equivalent with respect to solution quality, major differences become evident with respect to the computing time which passed (on average per problem instance) until the best solution was found. On average, across all problem instances, the best solution was found by RBAS after 61.3% of the total computing time. For small problem instances ($n$ = 20) it took 5.5% (cap = 30) and 31.2% (cap = 45), and for large instances ($n$ = 60) 73.4% (cap = 30) and 95.6% (cap = 75) of the total computing time. ILS, on the other hand, found the best solution much earlier; on average, ILS needed 17.4% of the total computing time, and even in the worst case only 30.9%. RBAS is particularly time consuming because the savings have to be recalculated for each ant and, likewise, a new tour length has to be determined for each feasible pair of batches. As for ILS, the only time-consuming part consists of the local search phase, which is also included in RBAS. Given the identical computing times, ILS generates and evaluates more solutions than RBAS does. The numerical experiments have shown that – depending on the problem class – up to 50% additional solutions can be considered.
The development of the solution quality of the two

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

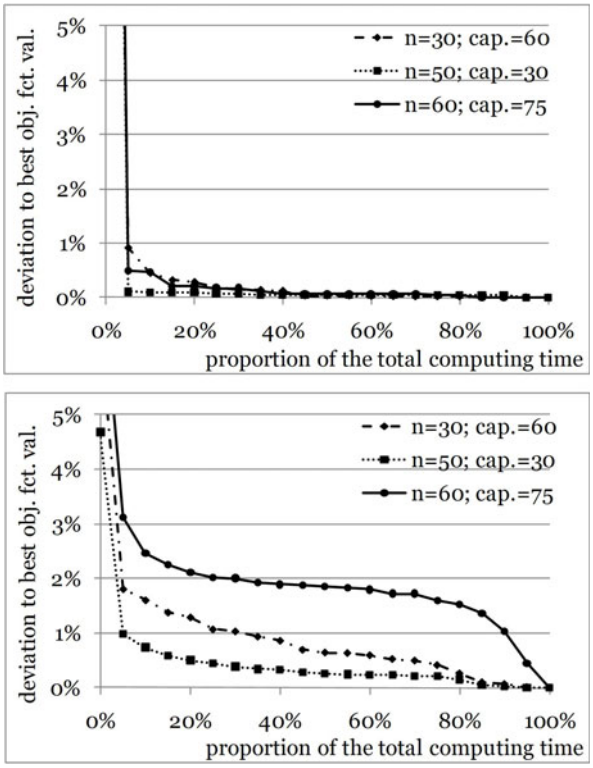
**Table 8: Computing Times for S-Shape Routing**

| *n* | cap. [no. art.] | ∅ time [sec.] | ILS [%] | RBAS [% ] |
|---|---|---|---|---|
| 20 | 30 | 7 | 7.0 | 5.5 |
| | 45 | 11 | 9.8 | 31.2 |
| | 60 | 13 | 17.1 | 18.4 |
| | 75 | 14 | 7.2 | 21.4 |
| 30 | 30 | 30 | 7.7 | 18.6 |
| | 45 | 55 | 23.3 | 48.7 |
| | 60 | 65 | 17.9 | 65.5 |
| | 75 | 69 | 16.5 | 57.4 |
| 40 | 30 | 88 | 13.4 | 43.0 |
| | 45 | 172 | 21.6 | 76.3 |
| | 60 | 208 | 24.3 | 83.6 |
| | 75 | 214 | 21.7 | 80.4 |
| 50 | 30 | 198 | 8.6 | 54.4 |
| | 45 | 410 | 26.6 | 87.3 |
| | 60 | 494 | 21.9 | 90.2 |
| | 75 | 513 | 20.1 | 87.5 |
| 60 | 30 | 432 | 13.0 | 73.4 |
| | 45 | 845 | 30.9 | 93.2 |
| | 60 | 1,013 | 14.6 | 93.7 |
| | 75 | 1,088 | 24.5 | 95.6 |
| Average | | | 17.4 | 61.3 |
| Minimum | | | 7.0 | 5.5 |
| Maximum | | | 30.9 | 95.6 |

*"∅ time [sec.]": average computing times of RBAS in seconds per instance; "ILS [%]": average percentage of computing time when ILS finds the best solution; "RBAS [%]": average percentage of time when RBAS finds the best solution.*

heuristics as a function of the computing time is depicted in Figure 3 for the three problem classes ($n$ = 30; cap = 60), ($n$ = 50; cap = 30) and ($n$ = 60; cap = 75). Each of the functions represents the development of the solution quality for one problem class. The x-axis depicts the computing time of the algorithms, with 100% representing the termination time of the algorithms. On the y-axis it is represented how far the best solution which has been found until a certain point in time deviates from the best solution obtained after 100% (final solution) of the computing time. For this representation the deviation from the objective function value of the final solution is measured in 20 intervals of identical size; each point in the graph

**Figure 3: Development of the Solution Quality over Time for S-Shape Routing and ILS (top) and RBAS (bottom)**




represents the average deviation computed over all 40 instances in a problem class.

The development for ILS is shown in the upper part, and for RBAS in the lower part of Figure 3. The graphs demonstrate that ILS finds the best solution much faster than RBAS does. The functions for ILS are always very similar to each other, independent from the problem class under consideration. In contrast to ILS, the functions for RBAS are different for every problem class. For the problem class with 50 customer orders and a picking device capacity of 30 articles ($n$ = 50; cap = 30) the objective function values are very close to the best ones (deviation only 1%) within the first 5% of the allocated computing time. For the problem class ($n$ = 30; cap = 60) already more than 30% of the computing time is needed to reduce the deviation to 1%. The problem class ($n$ = 60; cap = 75) requires 30% of the time to reach a deviation of 2% and even 90% to obtain a deviation of 1%; in order to identify the best solution, the entire computing time needs to be exploited. This development re-

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

veals that in case of RBAS the size of the picking device has a major influence on the development of the solution quality over time.

Table 9 provides information on the average computing times per problem instance for different phases of the optimization process when the LP/IP solver was applied. $t_{gen}$ represents the average time needed to set up the corresponding IPP, and $t_{opt}$ is the time spent from the moment when the optimization run has been started until the moment when the optimal solution has been found. $t_{prov}$ includes $t_{opt}$ plus the time necessary to prove the optimality of the solution. It has to be noted that only those instances are included in the table, for which the LP/IP solver was able to generate an optimal solution and verify its optimality. By comparing the computing times to those of Table 8, another strength of both ILS and RBAS becomes evident. The metaheuristics do not only provide high-quality solutions, they also do so in reasonable computing times, in particular for large problem instances. For problem classes with a small capacity of the picking device (cap = 30) the LP/IP solver is still faster than the metaheuristics for which it finds an optimal solution and proves its optimality in less than two seconds on average. For instances with ($n$ = 20; cap = 45) and ($n$ = 50; cap = 45) the LP/IP solver finds the optimal solution before ILS and RBAS terminate, but takes much additional time for proving its optimality. For all other problem classes the computing times of the metaheuristics are much shorter than the respective times needed by the LP/IP solver to even find the best solution. In general it can be observed that the computing times of the LP/IP solver increase exponentially while those of the metaheuristics exhibit an almost linear behavior.

It has to be noted again, however, that the number of optimal solutions which could be determined for each problem class varies quite significantly. As a consequence, Table 9 only gives a biased picture of the actual average computing times and, thus, should be read carefully. For instance, considering the increase of $t_{prov}$ for a move from smaller capacities of the picking device to larger ones (i.e. moving from cap = 30 to cap = 45, cap = 60 and cap = 75) for small problem sizes ($n$ = 20, 30), one would also expect $t_{prov}$ to increase for a transition from problem class ($n$ = 40; cap = 45) to problem class ($n$ = 40; cap = 60). The opposite is true, though, which must be attributed entirely to the fact that the LP/IP solver provided significantly fewer optimal solutions in the case of cap = 60 (16 instances) than in the case of cap = 45 (39).

Finally, another interesting aspect is that for some problem classes (e.g. ($n$ = 40; cap = 60)), the generation of the IPP only required almost as much computing time as did the application of the metaheuristics.

**Table 9: Computing Times for S-Shape Routing**

| $n$ | cap. [no. art.] | $t_{gen}$ [sec.] | $t_{opt}$ [sec.] | $t_{prov}$ [sec.] |
|---|---|---|---|---|
| 20 | 30 | 0 | 1 | 1 |
| | 45 | 3 | 4 | 25 |
| | 60 | 8 | 118 | 853 |
| | 75 | 16 | 72 | 1,041 |
| 30 | 30 | 0 | 1 | 1 |
| | 45 | 6 | 85 | 654 |
| | 60 | 29 | 680 | 2,163 |
| | 75 | - | - | |
| 40 | 30 | 2 | 1 | 1 |
| | 45 | 26 | 305 | 802 |
| | 60 | 198 | 622 | 713 |
| | 75 | - | - | - |
| 50 | 30 | 3 | 2 | 2 |
| | 45 | 134 | 705 | 1,130 |

*"$t_{gen}$ [sec.]": average computing times required to generate the IPP in seconds per instance; "$t_{opt}$ [sec.]": average computing times required by the LP/IP solver to obtain the best solution in seconds per instance; "$t_{prov}$ [sec.]": average computing times required by the LP/IP solver to prove the optimality of the provided solution in seconds per instance.*

### 7.2 Largest-Gap Routing

*Solution Quality*

The results from the numerical experiments for Largest-Gap Routing are presented in Table 10. Application of C&W(ii) provided solutions improving the objective function value of the solutions of FCFS by 16.17% on average (for all problem instances). The average improvements for the different problem classes ranged from 11.08% (no. of customer orders: 20; capacity of the picking device: 75) to 19.43% ($n$ = 60; cap = 60).

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

**Table 10: Solution Quality for Largest-Gap Routing**

| *n* | cap. [no. art.] | FCFS ∅ dist. [LU] | FCFS ∅ no. bat. | C&W (ii) ∅ dist. [LU] | C&W (ii) ∅ impr. [%] | C&W (ii) ∅ no. bat. | C&W (ii) + LS ∅ dist. [LU] | C&W (ii) + LS ∅ impr. [%] | C&W (ii) + LS ∅ no. bat. | ID ∅ dist. [LU] | ID ∅ impr. [%] | ID ∅ no. bat. | ILS ∅ dist. [LU] | ILS ∅ impr. [%] | ILS ∅ no. bat. | RBAS ∅ dist. [LU] | RBAS ∅ impr. [%] | RBAS ∅ no. bat. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 30 | 4,396 | 12.75 | 3,810 | 13.38 | 11.1 | 3,795 | 13.72 | 11.1 | 4,106 | 6.76 | 12.8 | 3,694 | 16.04 | 10.5 | **3,684** | 16.28 | 10.4 |
| | 45 | 3,279 | 7.95 | 2,816 | 14.05 | 7.3 | 2,779 | 15.18 | 7.3 | 2,933 | 10.51 | 8.0 | **2,684** | 18.08 | 6.9 | **2,684** | 18.08 | 6.9 |
| | 60 | 2,522 | 5.45 | 2,226 | 11.45 | 5.3 | 2,156 | 14.23 | 5.2 | 2,244 | 10.94 | 5.5 | **2,102** | 16.35 | 5.1 | **2,102** | 16.38 | 5.1 |
| | 75 | 2,168 | 4.43 | 1,922 | 11.08 | 4.2 | 1,852 | 14.30 | 4.2 | 1,929 | 10.99 | 4.4 | 1,800 | 16.70 | 4.1 | **1,799** | 16.74 | 4.1 |
| 30 | 30 | 6,628 | 19.53 | 5,661 | 14.74 | 16.8 | 5,642 | 15.03 | 16.8 | 6,147 | 7.43 | 19.5 | 5,448 | 17.99 | 15.7 | **5,436** | 18.18 | 15.6 |
| | 45 | 4,803 | 11.55 | 4,007 | 16.53 | 10.5 | 3,958 | 17.58 | 10.4 | 4,218 | 12.21 | 11.6 | 3,831 | 20.22 | 10.0 | **3,827** | 20.30 | 9.9 |
| | 60 | 3,974 | 8.58 | 3,349 | 15.71 | 7.9 | 3,285 | 16.49 | 7.9 | 3,458 | 12.94 | 8.6 | **3,172** | 20.14 | 7.7 | 3,176 | 20.05 | 7.7 |
| | 75 | 3,350 | 6.78 | 2,862 | 14.44 | 6.4 | 2,763 | 17.46 | 6.3 | 2,893 | 13.63 | 6.8 | **2,648** | 20.88 | 6.2 | **2,648** | 20.89 | 6.2 |
| 40 | 30 | 8,718 | 25.38 | 7,336 | 15.93 | 21.4 | 7,309 | 16.24 | 21.4 | 8,011 | 8.23 | 25.4 | 7,086 | 18.79 | 20.3 | **7,080** | 18.87 | 20.2 |
| | 45 | 6,519 | 15.55 | 5,347 | 17.87 | 14.0 | 5,296 | 18.67 | 13.9 | 5,694 | 12.65 | 15.6 | **5,148** | 20.96 | 13.4 | **5,148** | 20.97 | 13.4 |
| | 60 | 5,160 | 11.05 | 4,225 | 18.04 | 10.1 | 4,148 | 19.56 | 10.1 | 4,391 | 14.88 | 11.1 | **4,019** | 22.06 | 9.9 | 4,024 | 21.96 | 9.9 |
| | 75 | 4,390 | 8.75 | 3,694 | 15.84 | 8.2 | 3,588 | 18.24 | 8.2 | 3,742 | 14.75 | 8.8 | 3,463 | 21.06 | 8.1 | **3,462** | 21.08 | 8.1 |
| 50 | 30 | 11,184 | 32.63 | 9,400 | 16.01 | 27.6 | 9,384 | 16.16 | 27.6 | 10,282 | 8.15 | 32.6 | 9,111 | 18.60 | 26.2 | **9,073** | 18.94 | 25.9 |
| | 45 | 7,853 | 18.70 | 6,413 | 18.32 | 16.9 | 6,336 | 19.30 | 16.8 | 6,774 | 13.78 | 18.8 | 6,171 | 21.43 | 16.4 | **6,168** | 21.47 | 16.4 |
| | 60 | 6,473 | 13.93 | 5,259 | 18.75 | 12.6 | 5,169 | 20.15 | 12.6 | 5,483 | 15.34 | 13.9 | 5,001 | 22.74 | 12.3 | **4,998** | 22.79 | 12.3 |
| | 75 | 5,458 | 10.93 | 4,514 | 17.22 | 10.2 | 4,409 | 19.18 | 10.2 | 4,634 | 15.10 | 10.9 | **4,249** | 22.13 | 10.1 | **4,249** | 22.12 | 10.0 |
| 60 | 30 | 13,515 | 39.15 | 11,282 | 16.64 | 33.2 | 11,256 | 16.84 | 33.1 | 12,352 | 8.70 | 39.2 | 10,917 | 19.37 | 31.4 | **10,888** | 19.58 | 31.3 |
| | 45 | 9,651 | 22.93 | 7,779 | 19.35 | 20.4 | 7,710 | 20.08 | 20.3 | 8,280 | 14.18 | 23.0 | 7,520 | 22.04 | 19.9 | **7,516** | 22.09 | 19.9 |
| | 60 | 7,645 | 16.28 | 6,158 | 19.43 | 15.0 | 6,049 | 20.86 | 14.9 | 6,412 | 16.13 | 16.7 | **5,868** | 23.24 | 14.7 | 5,870 | 23.22 | 14.7 |
| | 75 | 6,615 | 13.10 | 5,386 | 18.53 | 12.2 | 5,259 | 20.42 | 12.2 | 5,548 | 16.10 | 13.0 | **5,059** | 23.47 | 11.9 | 5,071 | 23.30 | 11.9 |
| Average | | | | | 16.17 | | | 17.48 | | | 12.17 | | | 20.12 | | | 20.16 | |
| Minimum | | | | | 11.08 | | | 13.72 | | | 6.76 | | | 16.04 | | | 16.28 | |
| Maximum | | | | | 19.43 | | | 20.86 | | | 16.13 | | | 23.47 | | | 23.30 | |

*"n": number of customer orders; "cap. [no. art.]": capacity of the picking device in the number of articles; "∅ dist. [LU]": average total length of picking tours in length units; "∅ impr. [%]": improvement of the algorithm compared to the FCFS solution in percent; "∅ no. bat.": average number of batches. For each problem class the best obtained average distance is highlighted bold.*

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

As has been shown for S-Shape Routing, it can also be observed here that a large problem size induced an increase of the improvements. Improvements tended to be larger for medium-size capacities of the picking device (cap = 45, 60) than for small (cap = 30) and large capacities (cap = 75).

C&W(ii)+LS led to an average improvement of 17.48%, with a minimum improvement of 13.72% ($n$ = 20; cap = 30) and a maximum increase of 20.86% ($n$ = 60; cap = 60). Similar to the S-Shape case, the improvements obtained with the additional classic local search phase are not very convincing: the average improvement amounted to only 1.3 percentage points compared to C&W(ii).

Application of the (modified) ID resulted in an average improvement of only 12.17% with respect to the length of all picking tours. The improvements ranged from 6.76% ($n$ = 20; cap = 30) to 16.13% ($n$ = 60; cap = 60). Again, these results are inferior compared to the ones of the basic savings approach.

For Largest-Gap Routing the best solutions were provided by means of the newly proposed heuristics ILS and RBAS. Their application improved the objective function value, compared to FCFS, by 20.12% (ILS) and 20.16% (RBAS), respectively, on average for all instances. The improvements ranged between 16.04% and 23.47% for ILS, and between 16.28% and 23.30% for RBAS. For both, ILS and RBAS, improvements were again larger for problem classes with larger capacities of the picking device.

With respect to the average length of the picking tours, RBAS performed better than ILS on 11 out of the 20 problem classes, even though the differences in the objective function values are again only marginal and cannot really be looked upon as practical significant. RBAS slightly outperformed ILS on problem classes with small capacities of the picking device. Like in S-Shape Routing, also the average number of batches is practically not significantly different for ILS and RBAS across the different problem classes.

The improvements provided by ILS exceeded those of C&W(ii) by 3.95 percentage points on average across all instances, ranging from 2.59 ($n$ = 50; cap = 30) to 6.44 percentage points ($n$ = 30; cap = 75) for the different problem classes. For RBAS the additional improvement amounted to 3.99 percentage points on average for all instances, ranging from 2.74 ($n$ = 60; cap = 45) to 6.45 percentage points ($n$ = 30; cap = 75).

In comparison to C&W(ii)+LS, additional improvements of 2.64 percentage points (ILS) and 2.68 percentage points (RBAS) could be achieved by application of the new heuristics. Again, this demonstrates, that significant improvements can still be obtained.

The solution quality of ILS and RBAS is presented in Table 11, which depicts the average deviation of the obtained objective function values from the corresponding optimal values. Similar to the case of S-Shape Routing the average deviation is small (ILS: less than 0.85%; RBAS: less than 0.74%).

**Table 11: Solution Quality for Largest-Gap Routing**

| $n$ | cap. [no. art.] | ILS dev. [%] | RBAS dev. [%] |
|---|---|---|---|
| 20 | 30 | 0.35 (0.35) | 0.07 (0.07) |
| | 45 | 0.19 (0.19) | 0.18 (0.18) |
| | 60 | 0.03 (0.03) | 0.00 (0.00) |
| | 75 | 0.07 (0.07) | 0.01 (0.01) |
| 30 | 30 | 0.32 (0.32) | 0.08 (0.08) |
| | 45 | 0.40 (0.32) | 0.28 (0.28) |
| | 60 | 0.12 (0.00) | 0.28 (-0.07) |
| | 75 | - | - |
| 40 | 30 | 0.40 (0.40) | 0.30 (0.30) |
| | 45 | 0.47 (-0.06) | 0.45 (0.04) |
| | 60 | 0.53 (-0.35) | 0.14 (-0.23) |
| | 75 | - | - |
| 50 | 30 | 0.81 (0.81) | 0.38 (0.38) |
| | 45 | 0.85 (-0.05) | 0.74 (-0.14) |

*"ILS dev. [%], RBAS dev. [%]": average deviation between the objective function value obtained by ILS/RBAS and the objective function value of the optimal solution in percent; in brackets: deviation between the objective function value obtained by ILS/RBAS and by the LP/IP solver only for those instances, for which the LP/IP solver was not able to prove the optimality of the obtained solutions.*

*Computing Times*

Table 12 presents the computing times for ILS and RBAS; computing times for FCFS, C&W(ii), C&W(ii)+LS, and ID can be neglected and have been omitted again. Across all problem classes the average computing time per problem instance for RBAS (and – likewise – for ILS) varied between 12 seconds ($n$ = 20; cap = 30) and 2,660 seconds ($n$ = 60; cap = 75).

As for S-Shape Routing, the computing times are

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

**Table 12: Computing Times for Largest-Gap Routing**

| *n* | cap. [no. art.] | ∅ time [sec.] | ILS [%] | RBAS [% ] |
|---|---|---|---|---|
| 20 | 30 | 12 | 7.1 | 5.5 |
| | 45 | 23 | 7.0 | 14.7 |
| | 60 | 29 | 7.6 | 8.2 |
| | 75 | 33 | 12.3 | 8.3 |
| 30 | 30 | 55 | 10.6 | 21.9 |
| | 45 | 115 | 8.9 | 31.8 |
| | 60 | 146 | 15.0 | 33.6 |
| | 75 | 166 | 15.7 | 42.7 |
| 40 | 30 | 179 | 15.1 | 36.7 |
| | 45 | 362 | 26.0 | 51.2 |
| | 60 | 456 | 24.0 | 76.5 |
| | 75 | 525 | 18.0 | 77.1 |
| 50 | 30 | 410 | 19.3 | 55.0 |
| | 45 | 856 | 28.0 | 83.8 |
| | 60 | 1,099 | 24.2 | 87.7 |
| | 75 | 1,296 | 11.7 | 87.5 |
| 60 | 30 | 848 | 16.1 | 64.1 |
| | 45 | 1,829 | 30.5 | 86.1 |
| | 60 | 2,321 | 22.5 | 92.9 |
| | 75 | 2,660 | 25.4 | 91.3 |
| Average | | | 17.3 | 52.8 |
| Minimum | | | 7.0 | 5.5 |
| Maximum | | | 30.5 | 92.9 |

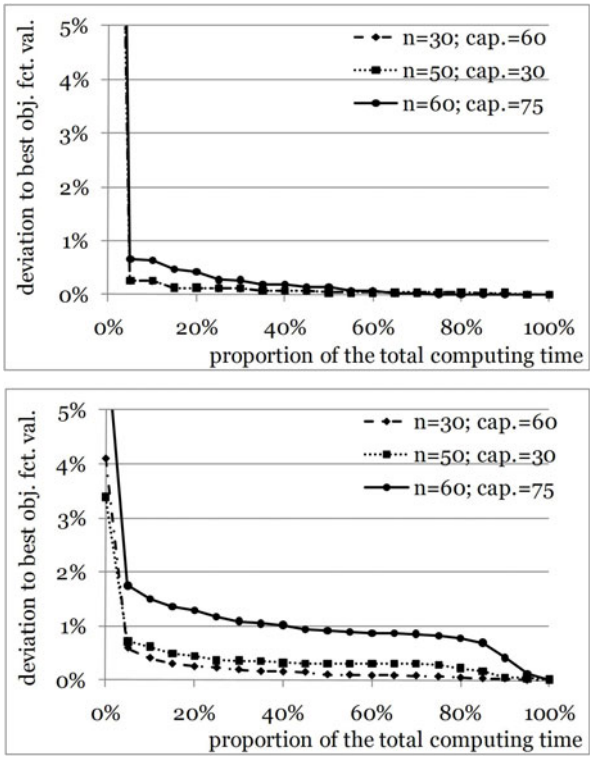
*"ILS dev. [%], RBAS dev. [%]": average deviation between the objective function value obtained by ILS/RBAS and the objective function value of the optimal solution in percent; in brackets: deviation between the objective function value obtained by ILS/RBAS and by the LP/IP solver only for those instances, for which the LP/IP solver was not able to prove the optimality of the obtained solutions.*

strongly dependent on the number of orders, as well as on the capacity of the picking device. The percentage of the actual computing time which passed until the best solution was found, turned out to be significantly smaller for ILS than for RBAS. On average, only 17.3% was needed by ILS, against 52.8% by RBAS. Across all problem classes, this percentage ranged from 7.0% ($n$ = 20; cap = 45) to 30.5% ($n$ = 60; cap = 45) for ILS, and from 5.5% ($n$ = 20; cap = 30) to 92.9% ($n$ = 60; cap = 60) for RBAS.

The development of the solution quality of the two heuristics as a function of the computing time is similar to the S-Shape case. Again, as shown in Figure 4, ILS (top) finds the best solutions much earlier than RBAS (bottom) does. Also, the functions of the different problem classes are similar for ILS and different for RBAS. For RBAS good solutions are found much faster for problem classes with smaller order sizes than for problem classes with larger ones.

The comparison of the computing times needed for the metaheuristics and the times required for solving the IPP (Table 13) reveals the same result as already observed for the S-Shape case. For the small problem classes with cap = 30 the computing times for solving the IPP and for proving the optimality of the obtained solutions are neglectable, since they are smaller than two seconds. For the other problem classes, again an exponential increase of the computing times can be observed.

**Figure 4: Development of the Solution Quality over Time for Largest-Gap Routing and ILS (top) and RBAS (bottom)**




## 7.3 Comparison of Routing Strategies

Subject to the parameters of the warehouse and the composition of the customer order data, one routing strategy can be more favorable than the

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

**Table 13: Computing Times for Largest-Gap Routing**

| $n$ | cap. [no. art.] | $t_{gen}$ [sec.] | $t_{opt}$ [sec.] | $t_{prov}$ [sec.] |
|---|---|---|---|---|
| 20 | 30 | 0 | 0 | 0 |
| | 45 | 1 | 4 | 9 |
| | 60 | 7 | 70 | 462 |
| | 75 | 22 | 493 | 2,463 |
| 30 | 30 | 1 | 1 | 1 |
| | 45 | 6 | 73 | 263 |
| | 60 | 30 | 720 | 1,759 |
| | 75 | - | - | |
| 40 | 30 | 2 | 2 | 2 |
| | 45 | 19 | 84 | 538 |
| | 60 | 218 | 1,012 | 1,184 |
| | 75 | - | - | - |
| 50 | 30 | 3 | 2 | 2 |
| | 45 | 124 | 1,167 | 1,382 |

*"ILS dev. [%], RBAS dev. [%]": average deviation between the objective function value obtained by ILS/RBAS and the objective function value of the optimal solution in percent; in brackets: deviation between the objective function value obtained by ILS/RBAS and by the LP/IP solver only for those instances, for which the LP/IP solver was not able to prove the optimality of the obtained solutions.*

other. Therefore, we continue our analysis with a comparison of the results obtained for the two routing strategies.

Hall (1993) reports that it can be observed that for a large number of articles to be picked on one route S-Shape Routing results in shorter tour lengths, while Largest-Gap Routing should be preferred for a small number of articles within one route. The same results can be observed in our experiments. Independent of the batching method and the problem size (total number of orders), Largest-Gap Routing – on average – outperforms S-Shape Routing for small capacities of the picking device, whereas for high capacities the opposite is true. This can be explained by the fact that an increase in the capacity of the picking device will also increase the picking intensity per aisle (i.e. average number of locations of articles to be picked per aisle). As a consequence, the maximal distance between two adjacent locations of articles to be picked in the same aisle will be reduced. This has no impact on the behavior of the S-Shape Heuristic, but application of the Largest-Gap Heuristic will result in solutions with longer travel distances within the aisles. In a situation, where the order picker has to collect an article from each storage location in an aisle, he/she would have to enter the aisle twice, once from the upper cross-aisle and once from the lower cross-aisle. In each case, the order picker would proceed to the center of the aisle and return from there to the cross-aisle, again. This would provide a tour which is twice as long as the length of the aisle.

When the heuristics in our experiment are applied, Largest-Gap Routing outperforms S-Shape Routing for small capacities of the picking device (cap = 30, 45). For a large capacity (cap = 75) the opposite is true. In the case that the picking device has a capacity of 60 articles, no specific precedence ordering of the routing strategies can be given with respect to the solution quality. Instead, which method has to be preferred depends on the size of the problem (number of customer orders). As mentioned before this result is valid for all batching heuristics. Therefore, it can be said that the decision about the routing strategy should only be based on the characteristics of the warehouse and the customer orders but is independent from the batching heuristics. Nevertheless, our experiments have shown that a better batching heuristic reduces the difference between the lengths of the picker tours resulting from the application of S-Shape on the one hand, and Largest-Gap on the other.

With respect to computing times, Largest-Gap Routing always requires more time than S-Shape Routing, due to the fact that one has to search for the largest gap between all picks within each aisle.

## 8 Conclusions and Outlook

In this article we considered the Order Batching Problem, a problem which is pivotal for the efficient management and control of manual picker-to-parts order picking systems in distribution warehouses. This problem is of major importance, since – due to the high amount of manual labor – order picking is the most cost-intensive function in a warehouse.

We introduced two algorithms, based on Iterated Local Search and on Ant Colony Optimization, for the solution of this problem. By means of extensive numerical experiments it could be demonstrated that – in terms of solution quality – both ap-

proaches are superior to existing methods. The results show further that the constructive heuristics as well as the basic local search approaches are not able to generate competitive solutions, whereas ILS and RBAS provide solutions which could improve the efficiency of distribution warehouse operations significantly. When compared to each other, both approaches do not differ very much with respect to the quality of the solutions they provide. However, ILS provides these solutions much faster, namely in only one third of the time required by RBAS. Also the development of the solution quality over time emphasizes how fast ILS provides high-quality solutions compared to RBAS. Additionally, ILS requires less parameter configuration than RBAS. Therefore, Iterated Local Search appears to be more suitable for being implemented in software systems for the solution of practical problems.

In detail, the numerical experiments show, that – on average – the application of the presented metaheuristics can reduce the length of the necessary picking tours by more than 20% in comparison to a FCFS solution. Compared to the best benchmark (C&W(ii) + Local Search) an improvement of up to three percentage points can be observed. Despite the fact, that the magnitude of the obtained improvements varies with respect to the problem size (number of orders), the characteristics of the used equipment (capacity of the picking device) and the chosen routing strategy, the results produced by RBAS and ILS are always superior to all considered benchmarks. Moreover, the experiments have shown that the achieved lengths of the picking tours are very close to the optimal ones (as far as it was possible to generate them). In addition to the superior solution quality it is important to point out that the metaheuristics are able to generate high-quality solutions in reasonable time, even for those problem classes which have proven to be difficult for the LP/IP solver (e.g., large number of orders and large capacity of the picking device).

Our research results are of practical importance with respect to two aspects: Firstly, the extraordinary reduction of the total picker tour lengths and the corresponding reduction of the total picking times establish the opportunity to reduce the regular working hours and/or overtime of the picking staff, and even downsize the workforce. This will turn out to be a powerful measure for improving the low profit margins still prevalent for warehouse operations and secure the existence of picking service providers in the long run. Secondly, reduced tour lengths result in shorter times for picking the customer orders. This gives rise to shorter delivery lead times and, more generally, to improved customer services which could be used by a warehouse operator as a central building block in a strategy for gaining a competitive advantage over its competitors.

Future research might follow two roads: From a methodological point of view, research could concentrate on speeding up the suggested methods (e.g., by introducing alternative neighborhoods), or on studying alternative, probably faster metaheuristics. From a practical point of view it would also be desirable to analyze dynamic order batching problems where customer orders come in continuously and batching is carried out under a rolling planning horizon. Situations of this kind evoke additional questions, such as when and how often picking plans should be updated in order to deal with newly arrived customer orders. Moreover – as in many industries customer orders have to be fulfilled at a certain time – it will be important to differentiate the various customer orders with respect to due dates, which necessitates the incorporation of time windows in the analysis. Nevertheless, algorithms will still have to be implemented which solve problems of the kind discussed in this paper as subproblems. We conclude that our solution approaches will also be valuable in dealing with such dynamic situations.

## References

Bozer, Yavuz A. and Justin W. Kile (2008): Order Batching in Walk-and-Pick Order Picking Systems, *International Journal of Production Research*, 46 (7): 1887–1909.

Brucker, Peter, Johann Hurink, and Frank Werner (1996): Improving Local Search Heuristics for Some Scheduling Problems - I, *Discrete Applied Mathematics*, 65: 97–122.

Bullnheimer, Bernd, Richard F. Hartl, and Christine Strauss (1999): A New Rank Based Version of the Ant System - A Computational Study, *Central European Journal of Operations Research*, 7 (1): 25–38.

Caron, Franco, Gino Marchet, and Alessandro Perego (1998): Routing Policies and COI-Based Storage Policies in Picker-to-Part Systems, *International Journal of Production Research*, 36 (3): 713–732.

Chen, Mu-Chen and Hsiao-Pin Wu (2005): An Association-Based Clustering Approach to Order Batching Considering Customer Demand Patterns, *Omega - International Journal of Management Science*, 33 (4): 333–343.

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

Choe, Kyung and Gunter Sharp (1991): Small Parts Order Picking: Design and Operation, http://www.isye.gatech.edu/logisticstutorial/order/article.htm (Access date: 2010-03-30).

Clarke, G. and John W. Wright (1964): Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research*, 12 (4): 568–581.

Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo (1991): Distributed Optimization by Ant Colonies, in: Francisco Varela and Paul Bourgine Varela (eds.): *Proceedings of the First European Conference on Artificial Life*, MIT Press: Cambridge, MA, 134–142.

Coyle, John J., Edward J. Bardi, and C. John Langley, Jr. (1996): *The Management of Business Logistics*, 6th ed., West Publishing Company: St. Paul.

de Koster, René, Kees Jan Roodbergen, and Ronald van Voorden (1999): Reduction of Walking Time in the Distribution Center of De Bijenkorf, in: M. Grazia Speranza and Paul Stähly (eds.): *New Trends in Distribution Logistics*, Springer: Berlin, 215–234.

de Koster, René, Edo van der Poort, and Matthijs Wolters (1999): Efficient Orderbatching Methods in Warehouses. *International Journal of Production Research*, 37 (7): 1479–1504.

Doerner, Karl, Manfred Gronalt, Richard F. Hartl, Marc Reimann, Christine Strauss, and Michael Stummer (2002): SavingsAnts for the Vehicle Routing Problem, in: Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and Günther R. Raidl (eds. ): *Applications of Evolutionary Computing, Lecture Notes in Computer Science 2279*, Springer: Berlin, 11-20.

Dorigo, Marco (1992): Optimization, Learning and Natural Algorithms, Ph. D. Thesis, Politecnico di Milano.

Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni (1991): Ant System: An Autocatalytic Optimizing Process, Technical Report No. 91-016, Dipartimento Elettronica e Informazione, Politecnico di Milano.

Dorigo, Marco and Thomas Stützle (2004): *Ant Colony Optimization*, MIT Press: Cambridge, MA.

Drury, Jolyon (1988): *Towards More Efficient Order Picking*, The Institute of Materials Management: Cranfield.

Elsayed, Elsayed A. (1981): Algorithms for Optimal Material Handling in Automatic Warehousing Systems, *International Journal of Production Research*, 19 (5): 525–535.

Elsayed, Elsayed A. and Richard G. Stern (1983): Computerized Algorithms for Order Processing in Automated Warehousing Systems, *International Journal of Production Research*, 21 (4): 579–586.

Elsayed, Elsayed A. and O.I. Unal (1989): Order Batching Algorithms and Travel-Time Estimation for Automated Storage/Retrieval Systems, *International Journal of Production Research*, 27 (7): 1097–1114.

European Logistics Association and A. T. Kearney (2004): *Excellence in Logistics 2004*, European Logistics Association, Brussels.

Frazelle, Edward (2002): *World-Class Warehousing and Material Handling*, McGraw-Hill: New York.

Gademann, Noud and Steef van de Velde (2005): Order Batching to Minimize Total Travel Time in a Parallel-Aisle Warehouse, *IIE Transactions*, 37 (1): 63–75.

Gibson, David R. and Gunter P. Sharp (1992): Order Batching Procedures, *European Journal of Operational Research*, 58 (1): 57–67.

Hall, Randolph W. (1993): Distance Approximations for Routing Manual Pickers in a Warehouse, *IIE Transactions*, 25 (4): 76–87.

Ho, Ying-Chin, Teng-Sheng Su, and Zhi-Bin Shi (2008): Order-Batching Methods for an Order-Picking Warehouse with two Cross Aisles, *Computers & Industrial Engineering*, 55 (2): 321–347.

Ho, Ying-Chin and Yu-Ying Tseng (2006): A Study on Order-Batching Methods of Order-Picking in a Distribution Center with two Cross Aisles, *International Journal of Production Research*, 44 (17): 3391–3417.

Hsu, Chih-Ming, Kai-Ying Chen, and Mu-Chen Chen (2005): Batching Orders in Warehouses by Minimizing Travel Distance with Genetic Algorithms, *Computers in Industry*, 56 (2): 169–178.

Jarvis, Jay M. and Edward D. McDowell (1991): Optimal Product Layout in an Order Picking Warehouse, *IIE Transactions*, 23 (1): 93–102.

Katayama, Kengo and Hiroyuki Narihisa (1999): Iterated Local Search Approach Using Genetic Transformation to the Traveling Salesman Problem, in: Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith (eds.): *Proceedings of the Genetic and Evolutional Computation Conference* 1, Morgan Kaufmann: San Fransisco, 321–328.

Lorenço, Helena R., Olivier C. Martin, and Thomas Stützle (2003): Iterated Local Search, in: Fred Glover and Gary A. Kochenberger (eds.): *Handbook of Metaheuristics*, Kluwer Academic Publishers: Norwell, 321–354.

Pan, Jason Chao-Hsien and S. Liu (1995): A Comparative Study of Order Batching Algorithms, *Omega - International Journal of Management Science*, 23 (6): 691–700.

Petersen, Charles G. and Roger W. Schmenner (1999): An Evaluation of Routing and Volume-based Storage Policies in an Order Picking Operation, *Decision Sciences*, 30 (2): 481–501.

Ratliff, H. Donald and Arnon S. Rosenthal (1983): Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem, *Operations Research*, 31 (3): 507–521.

Reimann, Marc, Karl Doerner, and Richard F. Hartl (2004): D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem, *Computers & Operations Research*, 31 (4): 563–591.

Stützle, Thomas (2006): Iterated Local Search for the Quadratic Assignment Problem, *European Journal of Operational Research*, 174 (3): 1519–1539.

Stützle, Thomas and Holger H. Hoos (2000): Max-Min Ant System, *Future Generation Computer Systems*, 16 (8): 889–914.

Tompkins, James A., John A. White, Yavuz A. Bozer, Edward Frazelle, and J.M.A. Tanchoco (2003): *Facilities Planning*, 3rd ed. , Wiley: New Jersey.

Tsai, Chieh-Yuan, James J.H. Liou, and Ting-Mei Huang (2008): Using a Multiple-GA Method to Solve the Batch Picking Problem: Considering Travel Distance and Order Due Time, *International Journal of Production Research*, 46 (22): 6533-6555.

Wäscher, Gerhard (2004): Order Picking: A Survey of Planning Problems and Methods, in: Harald Dyckhoff, Richard Lackes, and Joachim Reese (eds.): *Supply Chain Management and Reverse Logistics*, Springer: Berlin, 324–370.

BuR – Business Research
Official Open Access Journal of VHB
Verband der Hochschullehrer für Betriebswirtschaft e.V.
Volume 3 | Issue 1 | May 10 | 82–105

## Biographies

**Sebastian Henn** is a Ph.D. candidate with the Department of Management Science at the Faculty of Economics and Management of the Otto-von-Guericke University in Magdeburg. He studied Business Mathematics at the University of Kaiserslautern.

**Sören Koch** is a Ph.D. candidate with the Department of Management Science at the Faculty of Economics and Management of the Otto-von-Guericke University in Magdeburg. He studied Mechanical Engineering and Business Administration at the Technical University of Braunschweig.

**Karl F. Doerner** is Assistant Professor of Business Administration at the University of Vienna. He received his Ph.D. in Computer Science and Business Administration from the University of Vienna. Karl Doerner is scientific advisor of Salzburg Research Forschungsgesellschaft where he was employed as senior researcher for the period 2007-2008. His research interests include heuristic and hybrid optimization techniques in transportation and logistics. He has been involved in numerous national research projects. He has published more than 50 peer-reviewed scientific articles.

**Christine Strauss** is Associate Professor at the Faculty of Business, Economics and Statistics, University of Vienna. She holds a master's degree in business informatics from the University of Vienna and a doctoral degree in economics from the University of Zurich. She is the head of the research group on Electronic Business at the University of Vienna. Her current research focuses on the field of electronic business, with a particular emphasis on e-logistics.

**Gerhard Wäscher** is Professor and Chair of Management Science at the Faculty of Economics and Management, Otto-von-Guericke University in Magdeburg. His current research interests focus on Warehouse Management and Control and on Cutting and Packing. He has been Chairman of the Board of GOR – Gesellschaft für Operations Research from 2003 to 2006 and serves currently as Vice President of EURO – The Association of European Operational Research Societies.