OTTO-VON-GUERICKE-UNIVERSITY MAGDEBURG
FACULTY OF ECONOMICS AND MANAGEMENT

Metaheuristics for the Order Batching Problem in Manual Order Picking Systems

Sebastian Henn • Sören Koch • Karl Doerner
Christine Strauss • Gerhard Wäscher

FEMM Working Paper No. 20, June 2009

F E M M

Faculty of Economics and Management Magdeburg

Working Paper Series

Otto-von-Guericke-University Magdeburg
Faculty of Economics and Management
P.O. Box 4120
39016 Magdeburg, Germany
http://www.ww.uni-magdeburg.de/

Metaheuristics for the Order Batching Problem in Manual Order Picking Systems

Sebastian Henn*, Sören Koch*, Karl Doerner†,
Christine Strauss†, Gerhard Wäscher*

June 2009

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 one 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 for operating distribution warehouses significantly more efficient.

Keywords: Warehouse Management, Order Picking, Order Batching, Iterated Local Search, Ant Colony Optimization

*Otto-von-Guericke University Magdeburg, Faculty of Economics and Management

†University of Vienna, Faculty of Business, Economics and Statistics

Corresponding author:
Dipl.-Math. oec. Sebastian Henn
Otto-von-Guericke-University Magdeburg
Faculty of Economics and Management
Department of Management Science
Postbox 4120
39106 Magdeburg
{sebastian.henn@ovgu.de}

Contents

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 (cf. Petersen and Schmenner 1999, p. 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). Both aspects may have a negative impact on the competitiveness of the total supply chain. Like many other repetitive material-handling activities, order picking is still a function, which involves the employment of human operators at 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 order pickers drive or walk through the warehouse and collect the requested items; and parts-to-picker systems, in which automated storage and retrieval systems deliver the items to stationary order pickers (cf. Wäscher 2004, p. 324). With respect to systems of the first kind, three planning problems can be distinguished on the operative level (cf. Caron, Marchet, and Perego 1998, p. 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 (cf. de Koster, Roodbergen, and van Voorden 1999, p. 232), and for which predominantly traditional (constructive) heuristics have been suggested in the past. The aim of this article is to examine, whether 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 the new metaheuristics will be presented, Iterated Local Search in Section 4, and Rank-Based Ant System 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 in 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 (cf. 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 (cf. de Koster, Roodbergen, and van Voorden 1999, p. 230).
The examples of the routing schemes of the S-Shape Heuristic and the Largest Gap Heuristic (cf. Figure 2.1) demonstrate their straightforward character. The black squares symbolize the locations of articles to be picked on the respective routes.
The *S-Shape Heuristic* 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 the aisle entirely. Afterwards the order picker proceeds to the next aisle containing a requested article. An exception could be the last aisle: if the order picker is positioned in the front cross-aisle, he would pick the items of the last 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 entirely containing an article to be picked. 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 2.1 also shows that benefits may arise from collecting the articles of two (or more) customer orders in a single tour instead of two (or more) tours, in particular if the articles are identical or closely located to each other. The pictures on top and in the middle of Figure 2.1 depict the tours of two seperate customer orders, while the pictures at the bottom show tours resulting from the batching of the two customer orders and which are shorter than the total length of the seperate tours. Order pickers are usually 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 (cf. Choe and Sharp 1991). Here we consider a situation, where we assume that all 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 the

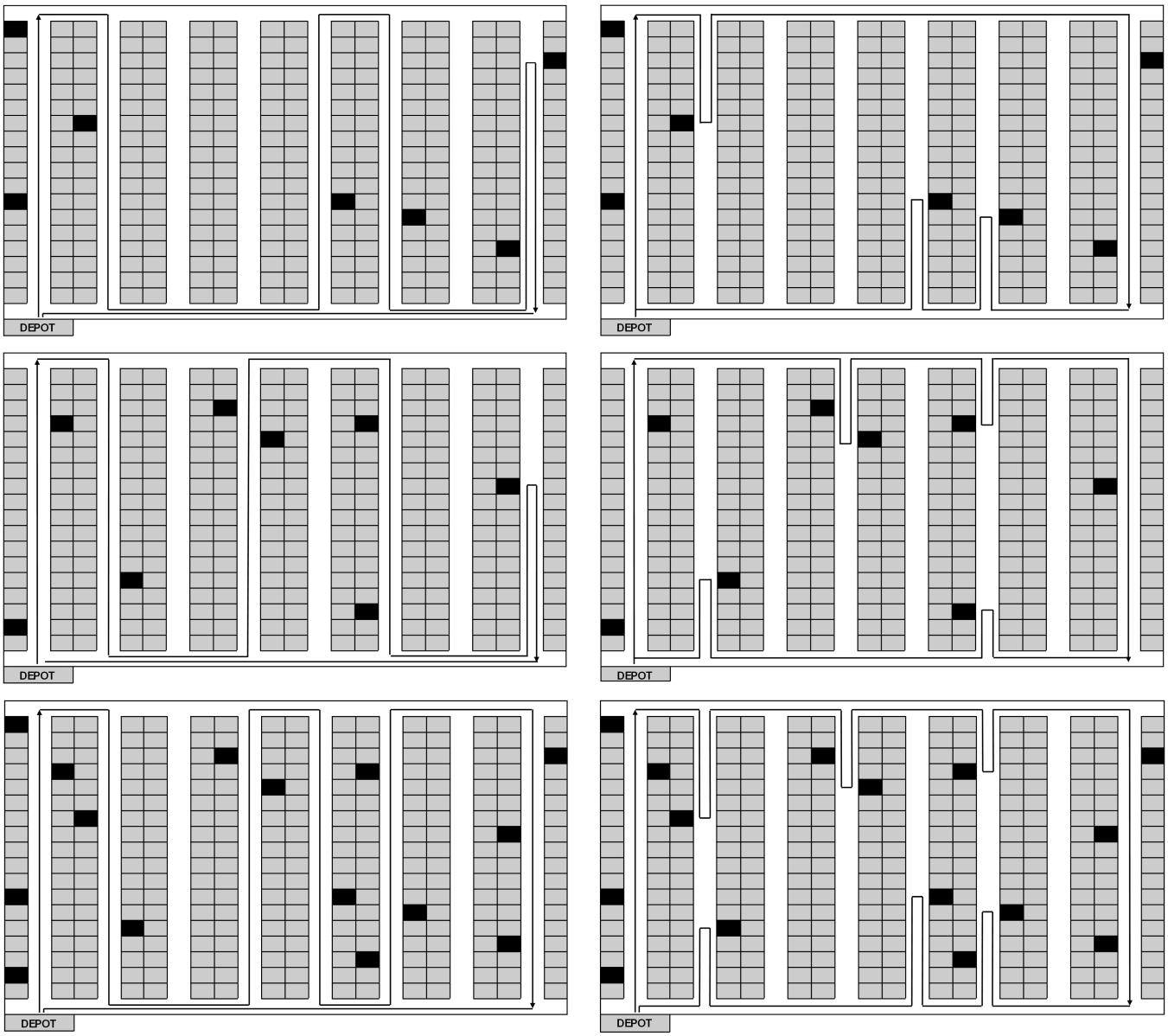

Figure 2.1: Examples of S-Shape-Routing (left) and Largest Gap-Routing (right)

most labor-cost intensive function in a warehouse (cf. Drury 1988). Consequently, the minimization of picking times is of great importance for the efficient control of the picking process. The components of the total order picking time (time spent by order pickers to collect the articles of all customer orders) are the setup times for the routes, the traveling times that are needed to travel to, from, and between the locations of articles to be picked, the search time for the identification of the articles, and the time needed for picking the articles (cf. Tompkins, White, Bozer, Frazelle, and Tanchoco 2003). Amongst these components, the traveling time is of outstanding importance, since it consumes the major proportion of the total order picking time, while the other components can either be assumed as constant (search times and pick times) or as neglectable (setup times). Furthermore, if the pickers' travel velocity is assumed to be constant, the minimization of the total traveling time is equivalent

to the miminization of the total length of the picker tours (cf. Jarvis and McDowell 1991, p. 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 strategies 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 (cf. Wäscher 2004, p. 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 $a_i = (a_{i1}, ..., a_{in})$ 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:

$$\sum_{j \in J} c_j \cdot a_{ij} \leq C, \ \forall i \in I. \tag{2.1}$$

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 in which all orders of a batch i are collected, then the following optimization model can be formulated:

$$\min \ \sum_{i \in I} d_i \cdot x_i \tag{2.2}$$

$$\text{s.t.} \ \sum_{i \in I} a_{ij} \cdot x_i = 1, \ \forall j \in J; \tag{2.3}$$

$$x_i \in \{0, 1\}, \ \forall i \in I. \tag{2.4}$$

The sets of constraints (2.3) and (2.4) ensure that a set of batches is choosen 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 rises 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 (cf. Gademann and van de Velde 2005).

3 Review of Solution Approaches

For the Order Batching Problem only few approaches have been developed which try to solve the problem to or 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 orders) in reasonable computing time. For the case of S-Shape routing Bozer and Kile (2008) present a mixed integer programming approach, that generates near optimal solutions for small (up to 25) order sizes.
Furthermore, Chen and Wu (2005) describe an order batching approach based on data mining and

integer programming. In this approach, at first, similarities of 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 suggested 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 (cf. 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 a second step in which the orders are allocated to batches. Several rules have been described in the 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) (cf. Wäscher 2004).

Seed algorithms, introduced by Elsayed (1981) and Elsayed and Stern (1983), generate batches sequentially. Their procedure can be divided in two phases: seed selection phase and order congruency phase. During the seed selection phase, an inital 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 in 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 (cf. 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 (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 seperate 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 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 recalculating the savings each time orders have been allocated to a batch (C&W(ii)). A savings algorithm that generates batches sequentially is the EQUAL algorithm (cf. 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 fit into the batch. In the small-and-large-algorithm (cf. 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 are then, in a non-ascending order of their size, 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 have shown that either seed algorithms or savings algorithms provide the best solution with respect to the total length of all necessary tours, dependent on the warehouse layout and customer order characteristics (cf. de Koster, van der Poort, and Wolters 1999).

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 is therefore limited to S-Shape-Routing, only. Tsai, Liou, and Huang (2008) describe an approach in which both batching and routing problems are solved by means of a Genetic Algorithm. Due to these special conditions of application, both approaches will not be considered any further, here.

4 Iterated Local Search

Iterated Local Search has been successfully applied to a variety of optimization problems, for instance to the Traveling Salesman Problem (cf. Katayama and Narihisa 1999), Scheduling Problems (cf. Brucker, Hurink, and Werner 1996), or the Quadratic Assignment Problem (cf. Stützle 2006). The main principle can be described as follows (cf. 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 solution. The new solution stemming from the local search phase has to pass an acceptance criterion in order to become the new inbumbent solution, otherwise the previous solution remains the incumbent solution for a further perturbation. 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. 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*). In the local search phase (cf. Function 4.1), a 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 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.

Function 4.1 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;
```

The perturbation phase (cf. Function 4.2) 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, that 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 λ (rearrangement factor) a constant. 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 new incumbent solution, if the difference between the length of all picking tours of the last solution $d(s)$ and 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 factor μ is in $[0, 1]$. Otherwise the incumbent solution will be perturbated again. Algorithm 4.1 summarizes the heuristic.

Function 4.2 perturbation(s, γ)

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;
  move - as long as the capacity constraint is not violated for l - q orders of k to l;
  move - as long as the capacity constraint is not violated for k - q orders of l to k;
  assign the remaining orders to a new batch;
end for
return s

Algorithm 4.1 Iterated Local Search (ILS)

Parameters: rearrangement factor λ, threshold factor μ, time interval t for an incumbent solution;
let s_{initial} be a solution provided by the FCFS rule;
$s^\star :=$ local_search(s_{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 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 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 (cf. Dorigo and Stützle 2004). Apart from the basic Ant System, several extensions have been developed, e.g. Max-Min Ant Systems (cf. Stützle and Hoos 2000), Ant Systems with elitist strategies or Rank-Based Ant Systems (cf. 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 (RBAS) has been chosen which combines greedy (savings) and adaptive (pheromone) aspects when deciding about the combination of two batches. The Ranked-Based Ant System was implemented since it has proven (cf. Doerner, Gronalt, Hartl, Reimann, Strauss, and Stummer 2002 and Reimann, Doerner, and Hartl 2004) to be a good variant of savings-based approaches.
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} (cf. 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 order combinations between the orders in batch k and the orders in batch l, divided by the number of possible order combinations:

$$\tilde{\tau}_{kl} = \frac{\sum_{i \in k} \sum_{j \in l} \tau_{ij}}{|k| \cdot |l|}$$

where

τ_{ij}: pheromone intensity of order combination (i, j),
$|k|$: number of orders in batch k,
$|l|$: number of orders in batch l.

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

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

where

$\tilde{\tau}_{kl}$: pheromone intensity of batch combination (k, l),
α: parameter for the control of the influence of the pheromone intensity $\tilde{\tau}_{kl}$ $(\alpha \geq 0)$,
sav_{kl}: savings obtained by combining the batches k and l,
β: parameter for the control of the influence of the savings sav_{kl} $(\beta \geq 0)$.

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 (cf. Function 4.1). The process is repeated for each ant that is used. Each ant represents a solution. Afterwards, the pheromones for all order combinations are updated. In analogy to the processes in nature a fraction ρ 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:

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

with $\Delta\tau_{ij} := \sum_{r=1}^{m} \Delta\tau_{ij}^{r}$,

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

and

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

where

θ: pheromone quantity $(\theta \geq 0)$.

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 5.2.

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 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 is depicted in Figure 2.1 and is identical to the one frequently used in literature (cf. de Koster, van der Poort, and Wolters 1999, Petersen and Schmenner 1999). The picking area

Algorithm 5.2 Rank-Based Ant System (RBAS)

Parameters: number of iterations, number of ants, initialization of the τ_{ij}, number of best solutions m, evaporation factor ρ, pheromone quantity θ, control parameter α, β;
  for $it = 0$ to number of iterations **do**
    for $a = 0$ to number of ants **do**
      create start solution s;
      repeat
        for all feasible pairs (k, l) of s **do**
          compute sav_{kl};
          determine $\tilde{\tau}_{kl}$;
        end for
        determine p_{kl};
        choose combination $(\tilde{k}, \tilde{l})$;
        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 $s_t^\star$ **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;

consists of 900 storage locations (cells) and we asume that a different article has been assigned to each storage location. The storage locations are partitioned in 10 (picking) aisles with 90 storage locations each (45 cells on both sides of each aisle). The aisles are numbered from 1 to 10, where aisle no. 1 is the leftmost aisle and aisle no. 10 the rightmost aisle. Within an aisle two-side-picking is assumed, i.e. beeing 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 leftmost 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 percent of the demand belongs to articles in class A, 36 percent to articles in B and 12 percent to articles in C. Within a class, the location of each article is determined randomly. Table 6.1 summarizes all warehouse parameters fixed for the numerical experiments.

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)

Table 6.1: Warehouse Parameters

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 (defining the maximal quantity of articles that can be assigned to a batch) is 30, 45, 60 and 75 articles, chosen in a way that a batch consists of 2, 3, 4 and 5 customer orders on average. 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 other words 1600 instances have been considered, each with one trial per instance. An overview of the problem classes considered in the experiments is given in Table 6.2.

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

Table 6.2: Problem Classes

6.3 Benchmarks

In the numerical evaluation, the performance of the newly proposed batching strategies Iterated Local Search (ILS) and Rank-Based Ant System (RBAS) is compared to the First-Come-First-Served (FCFS) rule, which is used as a baseline, while the savings algorithm C&W(ii) serves as a benchmark. C&W(ii) is chosen as benchmark, since numerical experiments (cf. de Koster, van der Poort, and Wolters 1999) have shown that it leads to good results for different warehouse layouts.

6.4 Algorithm Parameters

In a series of pre-tests, the parameter values have been determined: 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 α and β have been set to 5. The evaporation factor ρ has been set to 0.1, and the uprating factor θ 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 identical computing times, which is determined by RBAS. In other words, for each instance the problem is solved by RBAS and the resulting computing time is used as the termination condition for ILS.
In ILS the rearrangement factor λ 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 $t = t_{\text{RBAS}}/10$, where t_{RBAS} is the computing time of RBAS. Table 6.3 summarizes the parameter choice.

Rank-Based Ant System	
no. of iterations:	100
no. of ants:	no. of orders / 2
initial pheromone intensity τ_{ij}:	2.0
control parameter α:	5
control parameter β:	5
evaporation factor ρ:	0.1
no. of best local solutions m:	3
uprating factor θ:	0.001
Iterated Local Search	
termination condition:	t_{RBAS}
rearrangement factor λ:	0.3
threshold factor μ:	0.05
time interval t for an incumbent solution:	t_{RBAS} / 10

Table 6.3: Algorithm Parameters

6.5 Implementation and Hardware

The computations for all 1600 instances have been carried out on a desktop PC with a Pentium processor with 1.86 GHz and 1 GB RAM. The algorithms have been encoded in C++ using the DEV Compiler Version 4.9.9.2.

7 Results of the Experiments

7.1 S-Shape-Routing

The results from the numerical experiments for S-Shape routing are depicted in Table 4. For all problem instances, application of C&W(ii) improved the total length of all picking tours on average by 19.5 percent (in comparison to the application of FCFS only). Differentiated with respect to the different problem classes, the (average) improvement ranged from 10.27% (no. of customer orders: 20; capacity of the picking device: 75) to 22.39% (60; 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 – given these conditions – 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 traveling 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.

By application of the newly proposed heuristics ILS and RBAS, the objective function value could be improved (again, in comparison to FCFS) by 24.16% (ILS) and 23.91% (RBAS) on average for all problem instances, and varied between 20.02% and 26.95% for ILS, and between 20.19% and 26.56% for RBAS, respectively, across the different problem classes. For both, ILS and RBAS, improvements were larger for problem classes with larger capacities of the picking device. In general, the dispersion of the improvements tends to be smaller for ILS and RBAS than for C&W(ii).

In comparison to C&W(ii), on average, additional improvements of 5.56 percentage points could be achieved by application of ILS, and of 5.27 percentage points by application of RBAS, respectively. The additional improvements were as small as 3.53 percentage points (ILS; no. of customer orders: 40; capacity of the picking device: 30) and 3.60 percentage points (RBAS; 40; 30), but went up to 10.18 percentage points (for both ILS and RBAS; no. of customer orders: 20; capacity of the picking device: 75).

For 15 of the 20 problem classes, ILS provided the best results on average. Independently from the order size, ILS was always superior to RBAS for larger capacities of the picking device (i.e. for cap = 45, 60, 75), whereas RBAS outperformed ILS for a small capacity (cap = 30). It has to be noted, however, that the results from both methods do not differ significantly with respect to solution quality (average total length of picking tours per problem instance). This can also be seen from the

no. cust. ord.	cap. [no. art.]	**FCFS** ∅ dist. [LU]	∅ no. bat.	**C&W (ii)** ∅ dist. [LU]	∅ impr. [%]	∅ no. bat.	**ILS** ∅ dist. [LU]	∅ impr. [%]	∅ no. bat.	**RBAS** ∅ dist. [LU]	∅ impr. [%]	∅ no. bat.
20	30	5181	13.2	4364	15.90	11.3	4156	20.02	10.7	**4146**	20.19	10.7
	45	3431	7.8	2828	17.43	7.1	**2624**	23.51	6.8	2632	23.28	6.8
	60	2673	5.7	2348	11.32	5.3	**2086**	21.42	5.3	2090	21.26	5.3
	75	2137	4.6	1909	10.27	4.3	**1693**	20.45	4.2	**1693**	20.45	4.2
30	30	7813	19.7	6403	18.17	16.6	6086	22.27	15.8	**6077**	22.38	15.7
	45	5360	11.9	4273	20.20	10.9	**3999**	25.34	10.3	4015	25.02	10.3
	60	4005	8.6	3265	18.34	8.1	**2999**	25.05	7.7	3013	24.70	7.6
	75	3204	6.7	2709	15.23	6.2	**2461**	23.03	6.1	2479	22.44	6.1
40	30	10260	25.5	8174	20.42	21.2	7815	23.95	20.3	**7808**	24.02	20.3
	45	6966	15.4	5461	21.57	13.7	**5098**	26.95	13.2	5116	26.56	13.2
	60	5228	11.2	4242	18.86	10.3	**3898**	25.42	10.0	3933	24.76	10.0
	75	4151	8.7	3473	16.24	8.2	**3160**	23.78	8.0	3189	23.09	8.1
50	30	13093	32.8	10460	20.13	27.1	9963	23.95	25.9	**9950**	24.05	25.8
	45	8669	19.3	6746	22.11	17.2	**6377**	26.40	16.6	6386	26.27	16.5
	60	6525	13.9	5229	19.79	12.8	**4879**	25.17	12.6	4930	24.39	12.6
	75	5291	11.0	4316	18.37	10.2	**3995**	24.43	10.1	4032	23.76	10.1
60	30	15376	38.5	12037	21.84	31.7	11431	25.78	30.1	**11429**	25.79	30.1
	45	10192	22.6	7906	22.39	20.2	**7486**	26.52	19.7	7520	26.19	19.7
	60	7822	16.6	6082	22.25	15.0	**5761**	26.34	14.8	5797	25.87	14.8
	75	6199	12.9	5025	18.88	12.1	**4677**	24.51	11.9	4712	23.95	11.9
Average					19.50			24.16			23.91	
Minimum					10.27			20.02			20.19	
Maximum					22.39			26.95			26.56	

"no. cust. ord.": 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.

Table 7.1: Solution Qualities for S-Shape-Routing

difference in the average number of batches, which is 0.1 at most.

Table 7.2 depicts the average computing times per problem instance for ILS and RBAS. The times for FCFS and C&W(ii) have not been listed since these methods consumed less than five seconds per instance. The average computing time for RBAS (which determines the total computing time of ILS) varied between 6.9 seconds (no. of customer orders: 20; capacity of the picking device: 30) and 1,340.3 seconds (60, 75). Basically, two problem parameters determine 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 gave rise to increasing computing times. Even though no significant differences could be observed between ILS and RBAS 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 66.8% of the total computation time (results not shown in Table 7.2). For small problem instances (no. of customer orders: 20) it

took 22.7% (cap = 30) and 46.6% (cap = 75) of the total computing time, while for large instances (no. of customer orders: 60) 80.0% (cap = 30) and 90.8% (cap = 75) was needed. ILS, on the other hand, found the best solution much earlier; on average, ILS only needed 21.9%, and in the worst case only 36.6 percent of the total computation time. 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. What ILS is concerned, the only time consuming part is the local search phase, which is also included in RBAS. From this follows that – within identical computation times – ILS generates and evaluates far more solutions than RBAS does.

no. of cust. orders	cap. [no. art.]	∅ **time** [sec]	**ILS** [%]	**RBAS** [%]
20	30	6.9	14.4	22.7
	45	13.1	19.8	37.3
	60	15.8	20.0	33.4
	75	17.4	16.6	46.6
30	30	32.9	12.9	38.4
	45	64.8	23.1	61.2
	60	76.2	28.7	65.2
	75	86.0	24.2	65.0
40	30	106.1	15.5	66.0
	45	199.4	36.8	73.7
	60	242.8	23.0	81.8
	75	265.3	32.8	75.6
50	30	239.6	20.4	74.7
	45	477.2	20.9	76.8
	60	573.1	21.1	83.9
	75	655.6	20.5	89.8
60	30	523.3	24.2	80.0
	45	995.9	24.0	83.5
	60	1196.3	23.4	88.7
	75	1340.3	15.9	90.8

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

Table 7.2: Computing Times for S-Shape-Routing

7.2 Largest Gap-Routing

The results from the numerical experiments for Largest Gap routing are presented in Table 7.3. Application of C&W(ii) provided solutions improving the objective function value of the solutions of FCFS by 16.16% on average (for all problem instances). The average improvement for the different problem classes ranged from 10.73% (no. of customer orders: 20; capacity of the picking device: 75) to 19.43% (60; 60). As for S-Shape routing, it can also be observed here that improvements grow with

no. cust. ord.	cap. [no. art.]	**FCFS** ∅ dist. [LU]	∅ no. bat.	**C&W (ii)** ∅ dist. [LU]	∅ impr. [%]	∅ no. bat.	**ILS** ∅ dist. [LU]	∅ impr. [%]	∅ no. bat.	**RBAS** ∅ dist. [LU]	∅ impr. [%]	∅ no. bat.
20	30	4396	12.8	3810	13.38	11.1	3692	16.08	10.5	**3684**	16.28	10.4
	45	3279	8.0	2816	14.05	7.3	**2682**	18.14	6.9	2684	18.10	6.9
	60	2522	5.5	2226	11.45	5.3	**2102**	16.37	5.1	**2102**	16.38	5.1
	75	2168	4.4	1921	10.73	4.2	**1799**	16.73	4.1	**1799**	16.74	4.1
30	30	6628	19.5	5661	14.74	16.8	5463	17.76	15.8	**5436**	18.18	15.6
	45	4803	11.6	4007	16.53	10.5	3828	20.28	10.0	**3827**	20.30	9.9
	60	3974	8.6	3349	15.71	7.9	3176	20.05	7.7	**3175**	20.07	7.7
	75	3350	6.8	2862	14.51	6.4	**2647**	20.91	6.2	**2647**	20.91	6.2
40	30	8718	25.4	7336	15.93	21.4	7096	18.68	20.4	**7080**	18.86	20.2
	45	6519	15.6	5347	17.87	14.0	5151	20.93	13.5	**5148**	20.97	13.4
	60	5160	11.1	4224	18.06	10.1	**4016**	22.11	9.9	4022	21.99	9.9
	75	4390	8.8	3693	15.85	8.2	**3455**	21.24	8.1	3461	21.12	8.1
50	30	11184	32.6	9399	16.02	27.6	9106	18.65	26.2	**9075**	18.93	26.0
	45	7853	18.7	6413	18.32	16.9	6171	21.43	16.4	**6166**	21.49	16.4
	60	6473	13.9	5259	18.75	12.6	**4993**	22.87	12.3	4996	22.82	12.3
	75	5458	10.9	4514	17.26	10.2	**4247**	22.16	10.1	4249	22.12	10.0
60	30	13515	39.2	11282	16.64	33.2	10906	19.45	31.4	**10889**	19.58	31.3
	45	9651	22.9	7779	19.39	20.4	7518	22.07	19.9	**7515**	22.09	19.9
	60	7645	16.3	6158	19.43	15.0	**5865**	23.28	14.7	**5865**	23.28	14.7
	75	6615	13.1	5386	18.50	12.2	**5061**	23.45	11.9	5069	23.34	12.0
Average					16.16			20.13			20.28	
Minimum					10.73			16.08			16.28	
Maximum					19.43			23.43			23.34	

For abbreviations see Table 7.1.

Table 7.3: Solution Qualities for Largest Gap-Routing

an increase in the problem size (i.e. with an increasing number of customer orders). Improvements tend to be larger for medium-size capacities of the picking device (cap = 45, 60) than for small (cap = 30) and large capacities (cap = 75).

Again, in comparison to FCFS, application of ILS and RBAS improved the objective function value (average total length of picking tours per customer order) by 20,13% (ILS) and 20.28% (RBAS), respectively, on average for all instances. The average improvements ranged between 16.08% and 23.43% for ILS, and between 16.28% and 23.34% for RBAS. For both, ILS and RBAS, improvements were larger again for problem classes with larger capacities of the picking device, and the dispersion of the improvements was smaller for ILS and RBAS than for C&W(ii).

The improvements provided by ILS exceeded those of C&W(ii) by 3.97 percentage points on average across all instances, ranging from 2.63 (number of customer orders: 50; capacity of the picking device: 30.) to 6.40 percentage points (30; 75) for the different problem classes. For RBAS the additional improvement amounted to 4.12 percentage points on average for all instances, ranging from 2.70 (60; 45) to 6.40 percentage points (30; 75).

With respect to the average length of the pickings tours per customer order, RBAS performed better than ILS in 14 of the 20 problem classes, even though the differences in the objective function values are only marginal and cannot really be looked upon as significant. RBAS slightly outperforms ILS on problem classes with small capacities of the picking device, while RBAS is superior for classes with high capacities. Like in S-Shape routing, also the average number of batches is not significantly different for ILS and RBAS across the different problem classes.
Table 7.4 presents the computation times for ILS and RBAS; computation times for FCFS and C&W(ii) can be neglected and have been omitted again. Across all problem classes the average computational time per problem instance for RBAS (and – likewise – for ILS) varied between 14.3 seconds and 3,386 seconds (56.4 minutes). As for case of S-Shape routing, the computing time is 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 had to pass, until the best solution was found, again turned out to be significantly smaller for ILS than for RBAS. On the average, only 18.2% was needed by ILS, against 53.1% by RBAS. Across all problem classes, this percentage ranged from 5.2% to 34.1% (ILS), and from 5.6% to 91.9% for RBAS.

no. of cust. orders	cap. [no. art.]	∅ **time** [sec]	**ILS** [%]	**RBAS** [%]
20	30	14.3	5.2	5.6
	45	28.2	6.8	13.4
	60	37.6	6.9	8.6
	75	45.8	5.9	8.3
30	30	68.4	18.2	24.2
	45	144.3	20.8	32.8
	60	180.8	15.3	32.1
	75	207.6	20.2	40.9
40	30	218.3	7.4	38.6
	45	443.7	19.1	50.4
	60	568.0	21.7	76.6
	75	655.9	24.0	75.5
50	30	508.9	19.1	57.2
	45	1092.4	25.4	83.7
	60	1403.3	34.1	87.9
	75	1618.9	19.3	88.0
60	30	1060.3	16.0	67.2
	45	2258.5	24.4	86.4
	60	2891.5	30.8	91.6
	75	3386.8	23.0	91.9

For abbreviations see Table 7.2.

Table 7.4: Computing Times for Largest Gap-Routing

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 other. Therefore, we continue our analysis with a comparison of the results obtained for the two routing strategies.
In general, it can be observed that for a large number of articles to be picked on one route the S-Shape-Routing results in shorter tour lengths, while the Largest Gap-Routing should be prefered for a small number of articles within one route (cf. Hall 1993). 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 the application of the Largest Gap heuristic will result in solutions with longer travel distances within the aisles. In the worst case, the order picker has to collect an article from each storage location in an aisle, and Largest Gap routing would provide a tour within an aisle, which is twice as long as the length of the aisle.
As mentioned above, this effect is generally independent from the batching method. When FCFS and C&W(ii) 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 case that the picking device has a capacity of 60 articles, no specific ordering of the routing strategies can be given with respect to the solution quality. Instead, which method has to be preferred is dependent on the size of the problem (number of customer orders). The results for the two metaheuristics ILS and RBAS are similar apart from the fact that the switch from Largest Gap to S-Shape can already be observed for a capacity of 60 articles. The improvements (in comparison to the results from FCFS) obtained by joint application of C&W(ii) and S-Shape routing are up to 4 percentage points larger than improvements from C&W(ii) and Largest Gap routing for small capacities (30 and 45 articles) of the picking device. On the other hand, for large capacities, the joint application of C&W(ii) and Largest Gap routing results in improvements of only 0.5 percentage points. This suggests that C&W(ii) is able to compensate the weakness of FCFS, when the Largest Gap heuristic is used. If one of the metaheuristics is combined with S-Shape-Routing, the improvements are always larger than in the case where the metaheuristics are combined with Largest Gap.
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. We introduced two new metaheuristics, based on Iterated Local Search and on Ant Colony Opti-

mization, for the solution of this problem. By means of extensive numerical experiments it could be shown that – in terms of solution quality – both approaches are not only superior to existing methods but 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, since it provides these solutions in significantly shorter computing time, Iterated Local Search appears to be more suitable for being implemented in software systems for the solution of practical problems. Future research should concentrate on the development of alternative neighborhoods in order to speed up the local search step which, at present, is still fairly time-consuming.

References

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

Brucker, P.; Hurink, J.; Werner, F. (1996):
Improving Local Search Heuristics for Some Scheduling Problems - I. *Discrete Applied Mathematics* **65**, 97-122.

Bullnheimer, B,; Hartl, R.F.; Strauss, C. (1999):
A New Rank Based Version of the Ant System - A Computational Study. *Central European Journal of Operations Research* **7** (1): 25-38.

Caron, F.; Marchet, G.; Perego, A. (1998):
Routing Policies and COI-Based Storage Policies in Picker-to-Part Systems. *International Journal of Production Research* **36** (3), 713-732.

Chen, M.-C.; Wu, H.-P. (2005):
An Association-Based Clustering Approach to Order Batching Considering Customer Demand Patterns. *Omega - International Journal of Management Science* **33** (4), 333-343.

Choe, K.; Sharp, G. (1991):
Small Part Order Picking: Design and Operation. Available online at: http://www.isye.gatech.edu/logisticstutorial/order/article.htm.

Clarke, G.; Wright, J.-W. (1964):
Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research* **12**, 568-581.

Colorni, A.; Dorigo, M.; Maniezzo, V. (1991):
Distributed Optimization by Ant Colonies. In: Varela, F.; Bourgine, P. (eds.): Proceedings of the First European Conference on Artificial Life. MIT Press: Cambridge, MA, 134-142.

de Koster, R., Roodbergen, K.-J.; van Voorden, R. (1999):
Reduction of Walking Time in the Distribution Center of De Bijenkorf. In: Speranza, M.G.; Stähly, P. (eds.): New Trends in Distribution Logistics. Springer: Berlin, 215-234.

de Koster, R.; van der Poort, E.; Wolters, M. (1999):
Efficient Orderbatching Methods in Warehouses. *International Journal of Production Research* **37** (7), 1479-1504.

Dorigo, M. (1992):
Optimization, Learning and Natural Algorithms. PhD Thesis, Politecnico di Milano.

Dorigo, M.; Maniezzo, V.; Colorni, A. (1991):
Ant System: An Autocatalytic Optimizing Process. Technical Report No. 91-016, Dipartimento Elettronica e Informazione, Politecnico di Milano, Italy.

Dorigo, M.; Stützle, T. (2004):
Ant Colony Optimization. MIT Press: Cambridge, MA.

Doerner, K.; Gronalt, M.; Hartl, R.F.; Reimann, M.; Strauss, C.; Stummer, M. (2002):
SavingsAnts for the Vehicle Routing Problem. *Lecture Notes in Computer Science* **2279**, 73-109.

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

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

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

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

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

Gibson, D.R.; Sharp, G.P. (1992):
Order Batching Procedures. *European Journal of Operational Research* **58** (1), 57-67.

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

Ho, Y.-C.; Su, T.-S.; Shi, Z.-B. (2008):
Order-Batching Methods for an Order-Picking Warehouse with two Cross Aisles. *Computers & Industrial Engineering* **55** (2), 321-347.

Ho, Y.-C.; Tseng, Y.-Y. (2006):
A Study on Order-Batching Methods or Order-Picking in a Distribution Center with two Cross Aisles. *International Journal of Production Research* **44** (17), 3391-3417.

Hsu, C.-M., Chen, K.-Y.; Chen, M.-C. (2005):
Batching Orders in Warehouses by Minimizing Travel Distance with Genetic Algorithms. *Computers in Industry* **56** (2), 169-178.

Jarvis, J.M.; McDowell, E.D. (1991):
Optimal Product Layout in an Order Picking Warehouse. *IIE Transactions* **23** (1), 93-102.

Katayama, K.; Narihisa, H. (1999):
Iterated Local Search Approach Using Genetic Transformation to the Traveling Salesman Problem. In: Banzhaf, W.; Daida, J.; Eiben, A.E.; Garzon, M.H.; Honavar, V.; Jakiela, M.; Smith, R.E. (eds.): Proceedings of the Genetic and Evolutional Computation Conference, Volume 1, Morgan Kaufmann, San Fransisco, 321-328.

Lorenço, H. R.; Martin, O.C.; Stützle, T. (2003):
Iterated Local Search. In: Glover, F.; Kochenberger, G.A. (eds.): Handbook of Metaheuristics, Kluwer Academic Publishers, Norwell, 321-354.

Pan, J.C.-H.; Liu S. (1995):
A Comparative Study of Order Batching Algorithms. *Omega - International Journal of Management Science* **23** (6), 691-700.

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

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

Reimann, M.; Doerner, K.; Hartl, R.F. (2004):
D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem. *Computers & Operations Research* **31** (4), 563-591.

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

Stützle, T.; Hoos, H.H. (2000):
Max-Min Ant System. *Future Generation Computer Systems* **16** (8), 889-914.

Tompkins, J.A.; White, J.A.; Bozer, Y.A.; Tanchoco, J.M.A. (2003):
Facilities Planning. 3th ed., John Wiley & Sons: New Jersey.

Tsai, C.-Y.; Liou, J.J.H; Huang, T.-M. (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, G. (2004):
Order Picking: A Survey of Planning Problems and Methods. In: Dyckhoff, H.; Lackes, R.; Reese, J. (eds.): Supply Chain Management and Reverse Logistics, Springer: Berlin et al., 324-370.