WORKING PAPER SERIES

A Grouping Genetic Algorithm for the Order Batching Problem in Distribution Warehouses

Sören Koch/Gerhard Wäscher

Working Paper No. 26/2011

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Der Dekan

Verantwortlich für diese Ausgabe:
Sören Koch; Gerhard Wäscher
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

http://www.fww.ovgu.de/femm

Bezug über den Herausgeber
ISSN 1615-4274

A Grouping Genetic Algorithm for the Order Batching Problem in Distribution Warehouses

Sören Koch, Gerhard Wäscher

November 2011

Abstract: Order picking is a warehouse function that deals with the retrieval of articles from their storage locations in order to satisfy certain customer demands. Combining several single customer orders into one (more substantial) picking order can increase the efficiency of warehouse operations. The Order Batching Problem considered in this paper deals with the question of how different customer orders should be grouped into picking orders, such that the total length of all tours through the warehouse is minimized, which are necessary to collect all requested articles. For the solution of this problem, the authors introduce a Grouping Genetic Algorithm. This genetic algorithm is combined with a local search procedure which results in a highly competitive hybrid algorithm. In a series of extensive numerical experiments, the algorithm is benchmarked against a genetic algorithm with a standard item-oriented encoding scheme. The results show that the new genetic algorithm based on the group-oriented encoding scheme is preferable for the Order Batching Problem, and that the algorithm provides high quality solutions in reasonable computing times.

Keywords: Warehouse Management, Order Picking, Order Batching, Genetic Algorithms.

Corresponding author:

Dipl.-Wirtsch.-Ing. Sören Koch

Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
- Management Science -
P.O. Box 4120
39016 Magdeburg
soeren.koch@ovgu.de

Table of Contents

1 Introduction

Planning and control of warehouse processes is of significant importance to the efficient management of supply chains, since underperformance results in an unsatisfactory customer service and/or high costs. Considering the different warehouse functions (receiving, storage, order picking and shipping), this paper will focus on order picking which makes up for the most cost intensive operations. Different studies estimate that between 50% and 65% of the total warehousing operating costs can be attributed to order picking (Drury, 1988; Frazelle, 2002).

Order picking involves the retrieval of articles from their storage locations in order to satisfy a given demand specified by customer orders (Petersen and Schmenner, 1999). It arises since incoming articles are received and stored in unit loads while customers usually order small volumes of different articles. In manual-order-picking systems where order pickers collect the requested articles while walking or riding through the warehouse, three planning problems can be identified on the operative level: (i) assignment of articles to storage locations in the warehouse (article location assignment), (ii) grouping of customer orders into picking orders (order batching) and (iii) the determination of routes for the order pickers (picker routing).

This paper deals with order batching, which has a major effect on the efficiency of warehouse operations (de Koster et al., 1999a). Improved order batching reduces the total length of the picker tours significantly, which are necessary for picking all requested articles provided by a set of customer orders. This results in a reduction of the total picking time (time needed to collect the requested articles of all customer orders), which represents an increase in the efficiency of the picking operations (reduction of processing times and labor costs).

Hsu et al. (2005) have introduced a genetic algorithm for the Order Batching Problem, using a standard, item-oriented encoding scheme for the representation of solutions. Falkenauer (1992), however, has pointed out that for certain kinds of problems – so-called grouping problems – a different type of encoding scheme (i.e. a group-oriented encoding scheme) might be more appropriate. The Order Batching Problem considered in this paper is of such a kind. Thus, we will develop a corresponding genetic algorithm here, which makes use of a group-oriented encoding scheme. The new genetic algorithm will be additionally enhanced by a local search procedure, resulting in a hybrid algorithm. By means of extensive numerical experiments we will demonstrate that the proposed group-oriented approach is in fact superior to the existing item-oriented approach.

The paper is organized as follows: In Chapter 2, the Order Batching Problem will be described in detail and an overview on solution methods will be given. Chapter 3 contains a description of the general principle of genetic algorithms and the two specifications for the Order Batching Problem: the benchmark algorithm of Hsu et al. (2005) and the newly developed group-oriented genetic algorithm. In Chapter 4 the design of the numerical experiments will be outlined, before the numerical results will be shown in Chapter 5. A summary of the main findings and an outlook on further research opportunities will complete the paper.

2 The Order Batching Problem

2.1 Problem Description and Model Formulation

On their tours through the warehouse, order pickers are guided by so-called pick lists. A pick list consists of a set of order lines, where each line denotes an article requested by a customer, the location of the article in the warehouse, and the respective quantity of the article to be picked.

The articles of a pick list constitute a so-called picking order. With respect to the relationship between customer orders and picking orders three cases can be distinguished: (i) a customer order has to be split into several picking orders, since it contains too many articles, which cannot be picked on a single tour, (ii) a customer order is identical to a picking order, and (iii) a picking order consists of several customer orders. This paper deals with the third case, the grouping (batching) of customer orders into picking orders, which allows for a significant reduction of the total length of the picker tours that are necessary for picking all requested articles provided by a set of customer orders. Fig. 1 demonstrates exemplarily the obvious advantages of combining two customer orders into one picking order; the black rectangles symbolize the locations of articles to be picked and the black lines symbolize the tours that the order pickers are meant to take. Fig. 1a) and 1b) depict the picker routes if each customer orders is picked separately (pick-by-order). Fig. 1c) presents the corresponding route if the two customer orders are joint in a single picking order (pick-by-batch).

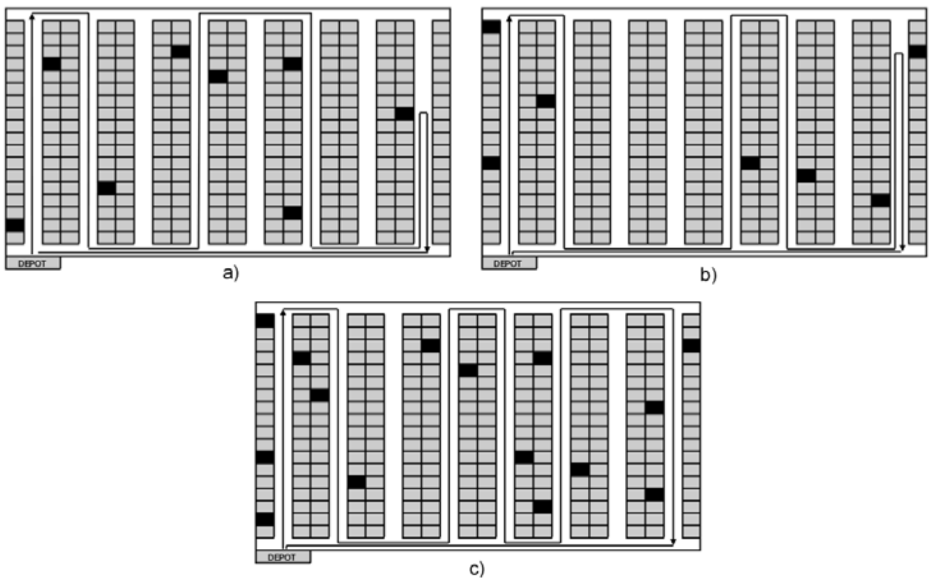

Fig. 1: *Different picker tours resulting from pick-by-order and pick-by-batch*

The batching of customer orders into picking orders can either be done as proximity batching or as time window batching. In proximity batching, customer orders are combined with respect to the locations of the articles in the warehouse; in time window batching this is done with respect to the (different) arrival times of the customer orders (Choe and Sharp, 1991). Time window batching can be further distinguished in variable time window batching (fixed number of customer orders in a time window of variable length) or fixed time window batching (variable number of customer orders in a time window of fixed length) (van Nieuwenhuyse and de Koster, 2009). However, in the following, proximity batching is considered only, since we assume a situation in which all customer orders are available at the beginning of the planning period.

Usually, the order lines on a pick list are already sorted in the sequence in which the articles are to be picked. This sequence determines the route that the order picker takes through the warehouse. According to these routes, each order picker starts at the depot, proceeds to the respective storage locations of the articles, and returns to the depot. Despite the fact that an optimal polynomial-time algorithm exists for the determination of optimal picker routes for some warehouse layouts (among which is the one depicted in Fig. 1; Ratliff and Rosenthal, 1983), in practice the routes are usually determined by means of so-called routing strategies, which can be interpreted as heuristic solution approaches. The advantage of such routing strategies is that they produce schemes that can be memorized fast and can be followed easily, whereas the routing schemes stemming from the optimal algorithm are not always straightforward and sometimes even confusing for the order pickers (de Koster et al., 1999b).

The prevalent working equipment for order pickers is a picking device, i.e. a cart or a roll pallet, which they take with them on their tours through the warehouse and on which the articles are placed when picked from their storage locations. If several customer orders are picked on one tour, a sort-while-pick strategy is usually applied. That means that the articles are already separated according to the respective customers, while the tour is being completed. Therefore, no additional sorting effort is necessary at the end of the tour. The avoidance of unnecessary sorting processes is also the reason why splitting of customer orders – case (i) described above – is usually not allowed. The number of customer orders that can be collected on a single tour is restricted by the capacity of the used picking device.

Order picking is considered to be the most labor-cost intensive function in a warehouse, due to the large amount of time-consuming manual operations (Drury, 1988). Therefore, the minimization of the picking times is of superior interest to a distribution company. In order to minimize the total picking time it is essential to indentify the different time components of the order picking process, which are the following: (i) setup times (ii) travel times, (iii) search times and (iv) times for taking the articles from their locations (Tompkins et al., 2003). From these four components, the travel time is the most important one, since it consumes the largest proportion of the total picking time and offers a significant potential for improvement. All other components can be considered as being constant. It should be noted that – assuming a constant travel velocity of the order pickers – a minimization of the total picking time (i.e. the time required to complete all picking tours) is equivalent to a minimization of the total length of all order picker tours. Efficient order batching can significantly reduce the necessary total tour length, and by doing so can help to reduce the total picking time.

Given a fixed assignment of articles to storage locations and a given routing strategy, the Order Batching Problem (OBP) can be defined in the following way: How can a set of customer orders be grouped (batched) into picking orders (batches) such that the capacity limitation of the picking device is not violated and the total length of all necessary picking tours is minimized. (Wäscher, 2004)

Gademann and van de Velde (2005) introduced a straightforward 0-1 optimization model for the Order Batching Problem whose columns consist of all feasible batches. In order to present this model, we introduce the following notation:

Index Sets:

J	set of customer orders, where $J = \{1,...,n\}$;
I	set of all feasible batches.

Constants:

a_{ij}	binary entry; $a_{ij} = 1$, if order j ($j \in J$) is included in batch i ($i \in I$), or $a_{ij} = 0$ otherwise;
C	capacity of the picking device;
c_j	capacity utilization required by customer order j ($j \in J$);
d_i	length of a picking tour in which all orders of batch i ($i \in I$) are collected.

Decision Variables:

x_i	binary decision variable; $x_i = 1$, if batch i ($i \in I$) is chosen, or $x_i = 0$, otherwise.

Each element $i \in I$ satisfies the following condition:

$$\sum_{j \in J} c_j \cdot a_{ij} \leq C, \qquad (1)$$

i.e., a batch is feasible, if it contains a set of customer orders that does not violate the available capacity of the picking device. The optimization model can then be formulated as follows:

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

subject to

$$\sum_{i \in I} a_{ij} \cdot x_i = 1, \qquad \forall j \in J; \qquad (3)$$

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

The sets of constraints (3) and (4) ensure that a set of batches is chosen in such a way that each customer order is included in exactly one of these batches.

The Order Batching Problem is known to be NP-hard (in the strong sense) if the number of customer orders per batch is larger than two (Gademann and van de Velde, 2005). Numerical experiments based on the above-given model formulation (Henn et al., 2010) revealed that only small instances can be solved to optimality if all columns (batches) are generated in advance. This can be explained by the fact that the number of possible batches and, consequently, the number of binary variables increases exponentially with the number of customer orders.

The model introduced above is general in the sense that it is not restricted to a specific routing strategy. Several such routing strategies are described in the literature (cf. Roodbergen, 2001). The one to which we refer here, the S-Shape Heuristic (or traversal strategy), appears to be the most frequently used one. Routing schemes resulting from the application of the S-Shape Heuristic are depicted in Fig. 1. The S-Shape Heuristic is characterized by the fact that each aisle that contains an article to be picked is traversed entirely, before the order picker moves on to the next aisle containing an article. In case that an odd number of aisles have to be visited, the last aisle containing an article to be picked must not be traversed entirely; instead the order picker will return directly to the depot after the last article has been picked.

2.2 Review of Solution Approaches

In the literature, only a few exact solution approaches have been proposed for the Order Batching Problem. These approaches include a branch-and-price algorithm with column generation (Gademann and van de Velde, 2005), and a mixed integer approach (Bozer and Kile, 2008). Both approaches share the same drawback of being limited to small problem sizes only, while problem instances of practical relevance need to be solved by means of heuristic methods.

Heuristics approaches to the OBP can be differentiated into two main categories: constructive heuristics and metaheuristics. The constructive heuristics can be further distinguished in priority rule-based algorithms, seed algorithms and savings algorithms (Wäscher, 2004).

In priority rule-based algorithms, customer orders are assigned to batches in the sequence of non-ascending priority values. The priority values can reflect any kind of ordering, e.g. one related to the size of the customer orders or one related to their arrival times. The latter corresponds to the First-Come-First-Served (FCFS)-rule, probably the best known representative of this sub-category. Other methods for the determination of priority rules include the space-filling curves by Gibson and Sharp (1992) and Pan and Liu (1995). The assignment of customer orders to batches can be done sequentially (Next-Fit-Rule) or simultaneously (First-Fit- and Best-Fit-Rule) (Wäscher, 2004): Using the Next-Fit-Rule, a customer order is assigned to a batch, if there is sufficient capacity available in the batch. In case the capacity utilization of a customer order would violate the capacity of the picking device, the current batch will be closed and a new one will be opened for it. Using the First-Fit-Rule, all batches are numbered in the sequence according to which they have been opened and a customer order will be assigned to the batch with the smallest number, which provides enough capacity. When the Best-Fit-Rule is applied, a customer order will be assigned to the batch with the least remaining capacity which is still sufficient for the inclusion of the customer order.

Seed algorithms for the OBP were first introduced by Elsayed (1981) and Elsayed and Stern (1983). In these algorithms the batches are generated sequentially. For each batch one customer order – the so-called “seed” – is chosen as an initial order in the batch. Afterwards, customer orders, which are “similar” to the seed, are added to the batch. In the literature a large variety of rules exist that describe how the initial customer order can be chosen and how the similarity of customer orders can be measured. An overview of various rules is given in de Koster et al. (1999a) and in Ho et al. (2008).

Solution methods of the savings type are inspired by the Clarke-and-Wright Algorithms for the Vehicle Routing Problem (Clarke and Wright, 1964). In the basic version of this algorithm, C&W(i), a savings value $sav_{ij} = s_i + s_j - s_{ij}$, is determined for each feasible combination of customer orders i and j, where s_i and s_j denote the lengths of the two tours on which the orders of customers i and j are collected separately, while s_{ij} denotes the tour length resulting from customer orders i and j being collected on a single tour. The different combinations (pairs) of customer orders are then sorted according to their savings values. Starting from the pair of customer orders possessing the highest savings value, three options are possible:

a) If none of the two customer orders has been assigned yet, a new batch will be opened for them.
b) If one of the customer orders has already been assigned to a batch and there is still enough capacity available in the batch, the second one is added to this batch. In case that the second order cannot be included in the batch due to a violation of the capacity constraint, the next pair of customer orders will be considered.
c) If both customer orders have already been assigned to batches, the next pair of customer orders will be chosen from the list.

An improved version of the algorithm, CW(ii), can be obtained if the savings are determined anew each time customer orders have been assigned to batches. For the computation of the new savings, customer orders included in one batch will be aggregated to one “large” customer order. In addition to these two savings algorithms presented here, several other variants are described in the literature, e.g. the EQUAL algorithm and the Small-and-Large algorithm (both introduced by Elsayed and Unal, 1989).

In the past few years several metaheuristics have been applied to the OBP. Gademann and van de Velde (2005) describe an iterated descent approach that is included in their branch-and-price algorithm. Albareda-Sambola et al. (2009) have developed a Variable Neighborhood Search Approach and Henn et al. (2010) have proposed an Iterated Local Search Method and a metaheuristic based on Ant Colony Optimization. A variant of Tabu Search, the Attribute Based Hill Climber, is presented by Henn and Wäscher (2010).

Only two papers have dealt with genetic algorithms so far. The approach of Tsai et al. (2008) describes a special case where the OBP and the routing problem are solved as an integrated problem. Due to this specific situation, their approach is not considered any further in this paper. The approach of Hsu et al. (2005) will be used as a benchmark method for our approach later. Section 3.2 provides a detailed description of their algorithm.

A detailed overview over solution methods for the OBP, including a detailed description of constructive heuristics as well as metaheuristics, is given in Henn et al. (2011).

3 Genetic Algorithms

3.1 General Principle

Genetic algorithms belong to the class of population-based metaheuristics. Their concept was first adapted to combinatorial optimization problems by Holland (1975). In a genetic algorithm, each individual represents a solution for the underlying optimization problem; a population consisting of several individuals represents a set of solutions. Over a certain number of iterations, called generations, different operations are applied to the individuals in analogy to the evolutionary processes in nature. By means of these operations new individuals are created that represent new solutions of the optimization problem. Over the course of several generations, fitter individuals (i.e. those individuals with a better solution quality) have a higher chance of survival than less fitter ones.

Fig. 2 depicts the general structure of a genetic algorithm. The design of such an algorithm requires several structural decisions, namely concerning the representation of solutions, the initialization of a starting population, and the components of the evolutionary process.

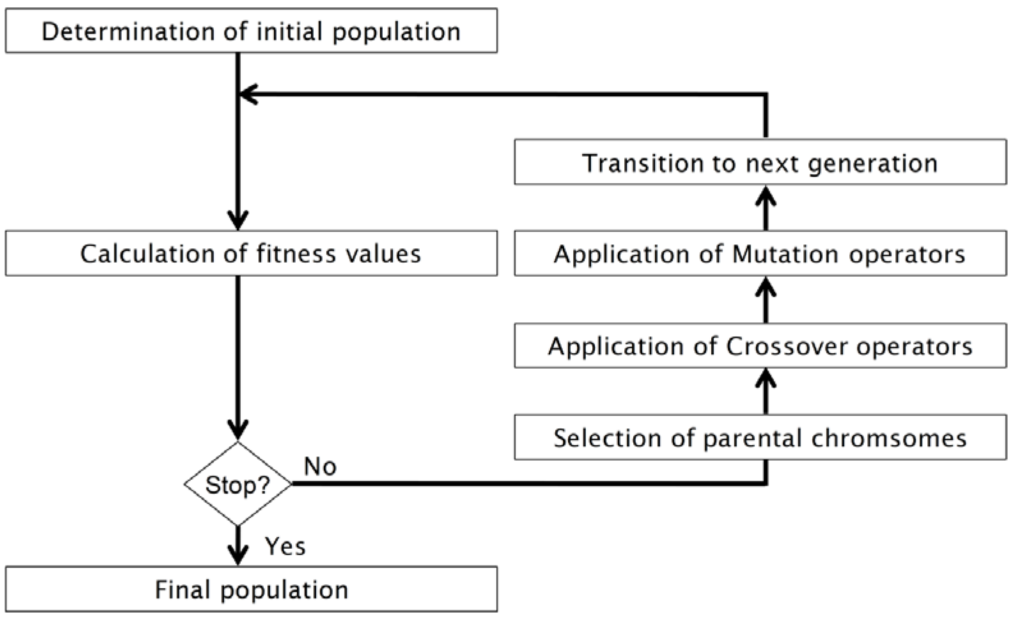
Determination of initial population
Transition to next generation
Calculation of fitness values
Application of Mutation operators
Application of Crossover operators
Selection of parental chromsomes
Stop?
No
Yes
Final population

Fig. 2: *General structure of a genetic algorithm (in analogy to Weicker, 2007)*

At first, an appropriate representation of the solutions, the encoding scheme (or alphabet), has to be selected. The encoding scheme describes how the information of a solution to the optimization problem will be represented by an individual. This information is captured in a string of genes, called the chromosome. Each of these genes represents a parameter of the solution. The representation can either be a direct one, i.e. the solution of the underlying optimization problem can be immediately identified from the chromosome, or an indirect one. In the latter case an additional decoding procedure is necessary which transforms the chromosome (genotype) into a solution of the optimization problem (phenotype).

During the initialization phase, a starting population (initial population) is determined. In order to obtain a good diversity of different solutions, the initial chromosomes are usually generated randomly.

The components of the evolutionary process include the calculation of fitness values, the selection of parental chromosomes, the crossover and mutation operators, and the transition to the next generation. In order to assess the quality of the current chromosomes, fitness values have to be computed at first. These fitness values are typically related to the objective function values of the optimization problem and, therefore, provide the connection between the genetic algorithm and the optimization problem. Different operators – called crossovers – can be applied to the individuals in order to simulate the evolutionary processes in nature (e.g. the mating of individuals). For this purpose, two individuals, the parental chromosomes, have to be chosen from the population by means of a selection mechanism. A large variety of such selection mechanisms (e.g. roulette wheel selection) exists in the literature. The crossover operators (e.g. one-point crossover, two-point crossover, uniform crossover) will then be applied to the parental chromosomes. This can be done separately or in combination with other crossover operators. In addition to the crossover, a second type of operator, the mutation, is performed in analogy to nature. This operator affects only single genes of a chromosome. Again, different mechanisms for mutation are described in the literature. After the mutation, the individuals are transferred to the next generation and the steps of the evolutionary process are repeated until a termination criterion (usually a certain number of generations) is reached. The best chromosome of the final population represents the solution of the genetic algorithm.

In the remaining parts of this chapter, two genetic algorithms for the OBP will be presented. The main differences between these two genetic algorithms are related to the implemented encoding scheme. The first one (Section 3.2), is the algorithm of Hsu et al. (2005) – which serves as a representative of an item-oriented genetic algorithm (IGA) – while the second one (Section 3.3) is the newly developed group-oriented genetic algorithm (GGA).

3.2 An Item-oriented Genetic Algorithm for the OBP

In the following, we describe the Genetic Algorithm of Hsu et al. (2005), which uses a standard item-oriented encoding scheme.

The length of the chromosomes is identical to the number of customer orders and the values of the genes correspond to the batch, to which the customer orders are assigned. Fig. 3 demonstrates this encoding scheme exemplarily for a solution consisting of eight customer orders:

Customer Orders		1	2	3	4	5	6	7	8	
Chromosome	(	B,	C,	A,	A,	B,	B,	C,	D	)

Fig. 3: *Representation of a solution based on an item-oriented encoding scheme*

In this example, the customer orders 3 and 4 are assigned to batch A, the customer orders 1, 5 and 6 are assigned to batch B, and so on. It is important to note that this encoding scheme might lead to infeasible solutions if too many customer orders are assigned to a single batch, resulting in a violation of the capacity constraint.

For the initial population, only feasible chromosomes are generated randomly. The chromosomes for the crossover operator are then determined by means of a roulette wheel selection, where the probability of choosing a chromosome into the mating pool is proportional to its fitness value. The fitness of an individual is related to the objective function value of the underlying optimization problem. However, the objective function value of the OBP has to be minimized, whereas the fitness values in genetic algorithms typically are to be maximized. In order to represent this situation, the fitness value of an individual is computed as the absolute value of the difference between the objective function value of the respective individual and the objective function value of the worst individual in the entire population. According to the description above, a fitness value of zero is assigned to the worst individual, resulting in a probability of zero for being chosen in the roulette wheel selection. In order to avoid this situation, a small epsilon value is added to each fitness value, which guarantees that each individual (even the worst one) can be chosen in the roulette wheel selection. With each spin of the roulette wheel one chromosome is chosen; the roulette wheel selection is repeated until the size of the mating pool equals the size of the population.

For the crossover operator the chromosomes in the mating pool are considered pairwise and a two-point crossover is applied with a certain crossover probability. In detail, this means that for each pair of chromosomes two crossover points are randomly determined and the genes between these points are exchanged. An example with crossover points in front of positions 4 and 7 is depicted in Fig. 4.

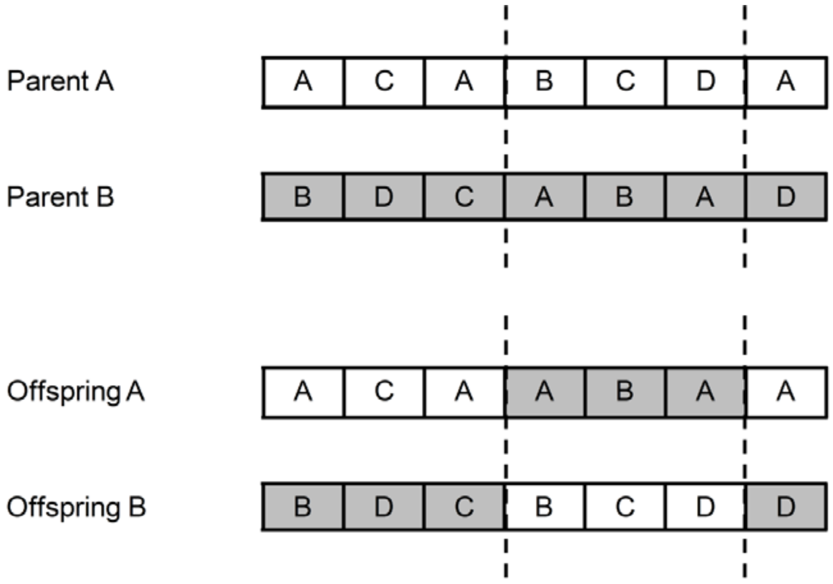

Fig. 4: *Example of a two-point crossover*

It is important to note that applying a crossover operator as shown in the example above might lead to infeasible solutions, due to violations of the capacity constraint. Such situations are dealt with by applying a correction mechanism to the infeasible offsprings which moves the last customer order from an infeasible batch to another batch with sufficient capacity according to the Best-Fit-Rule.

The mutation operator considered in this algorithm is of the SWAP type where the values of two genes are exchanged. This corresponds to an exchange of the batch assignment of two customer orders. A mutation for the genes three and six is shown in the following figure:

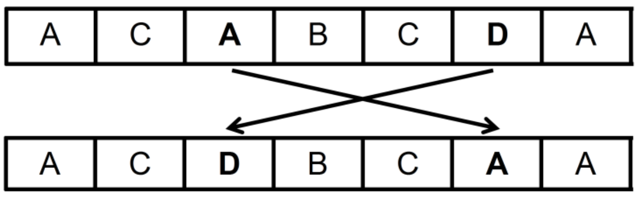

Fig. 5: *Example for a mutation*

A mutation of a chromosome occurs with a certain mutation probability. Similar to the crossover operator, the mutation operator may lead to an infeasible solution. In order to avoid such infeasibilities, we only allow for mutations which guarantee the feasibility of the solutions. After the mutation operator has been executed, the full population is transferred into the next generation. The algorithm terminates after a predefined number of generations.

3.3 A Group-oriented Genetic Algorithm for the OBP

The concept of a Grouping Genetic Algorithm was first introduced by Falkenauer (1992). He pointed out that the application of standard item-oriented encoding schemes (e.g. integer encoding or random key-based encoding) to grouping problems will result in two main problems. One is the large proportion of redundancy in the population, due to the fact that different chromosomes encode exactly the same solution. The second problem concerns the context insensitivity of the crossover operators, i.e. important information from the chromosome gets lost or is transferred incorrectly. Both aspects will deteriorate the successful application of genetic algorithms to grouping problems. In order to overcome these drawbacks, Falkenauer suggested a group-oriented chromosomal representation of solutions that reflects the structure of grouping problems much better (Falkenauer, 1996).

For the OBP, this translates into the fact that in a Grouping Genetic Algorithm the genes of the chromosome should represent batches instead of customer orders, as it is the case in the standard item-oriented encoding scheme. This implies that the chromosomes can be of variable length. A group-oriented encoding scheme for the OBP – as it is used in our algorithm – is depicted exemplarily for eight customer orders in Fig. 6. It shows that customer orders 3 and 7 are in one batch, customer orders 1 and 4 in another one, and so on.

Customer Orders		3, 7		1, 4		2, 5		6, 8	
Chromosome	(	A	,	B	,	C	,	D	)

Fig. 6: *Representation of a solution based on a group-oriented encoding scheme*

For the initial population, chromosomes are generated randomly. In our case, only chromosomes corresponding to feasible solutions have been generated, since application of a GGA that starts from a population with too many infeasible chromosomes will not result in high quality solutions.

Apart from the encoding scheme, the crossover operator in the GGA is group-oriented, too. For this reason the classic crossover operators have to be modified for being included in the GGA. Based on Falkenauer (1996), we suggest the following procedure:

1) Randomly select two parental chromosomes from the population!
2) Randomly select a crossing section in each of the chromosomes (cf. Fig. 7a)!
3) For each of the two chromosomes, insert the genes (batches) of the crossing section of one chromosome into the respective other one. The new genes (batches) will be inserted in front of the genes of the crossing section (cf. Fig. 7b)!
4) Mark all customer orders of the “old” genes (batches) which now occur twice in the chromosome (cf. circles in Fig. 7b)!
5) Delete all genes (batches) containing at least one of the marked customer orders; note the customer orders that have been deleted completely and are now missing (cf. Fig. 7c)!
6) Reinsert missing customer orders into new batches according to the First-Fit-Rule (Fig. 7d)!

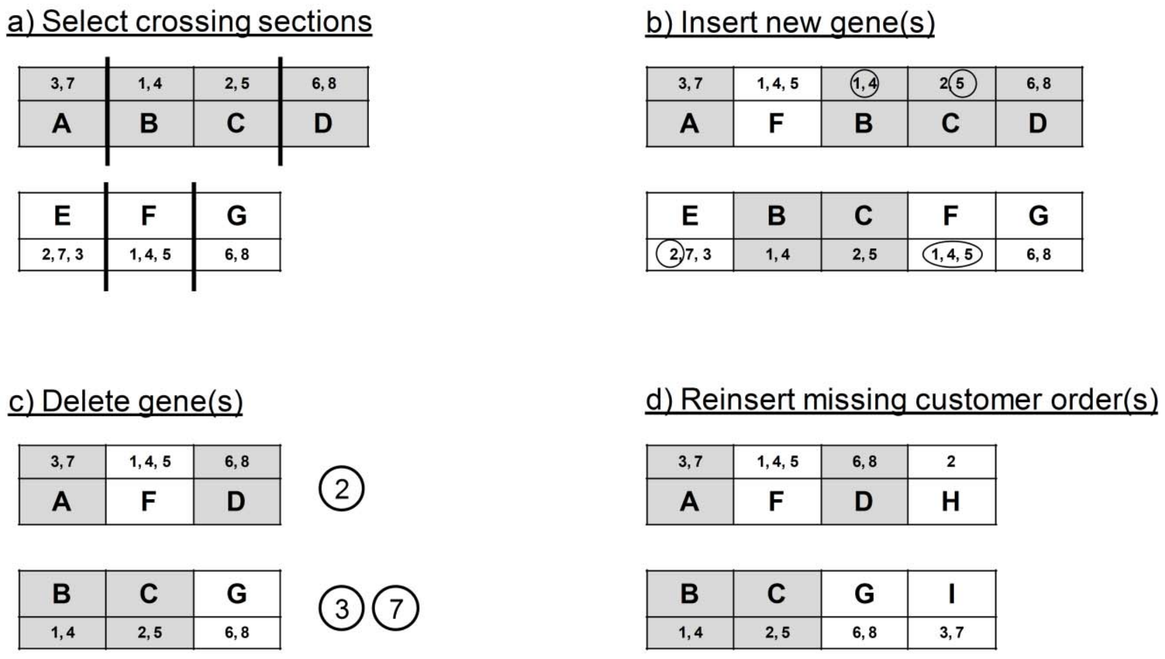

Fig. 7: *Crossover operator for grouping problems (in analogy to Falkenauer, 1995)*

The mutation operator included in the algorithm should be a group-oriented one as well. This is implemented by deleting existing genes (batches) from the chromosome and reinserting the affected customer orders in additional batches according to the Best-Fit-Rule.

For the transition to the next generation (cf. Fig. 8) an elitist strategy is applied (Goldberg, 1989). That means that a set of the best individuals (indicated by TOP in Fig. 8) is copied directly into the next generation. This procedure guarantees that the objective function value will improve monotonically. The remaining chromosomes of the new generation are generated by means of the crossover procedure described above, where one parental chromosome is chosen from the TOP part and the other from the non-TOP part.

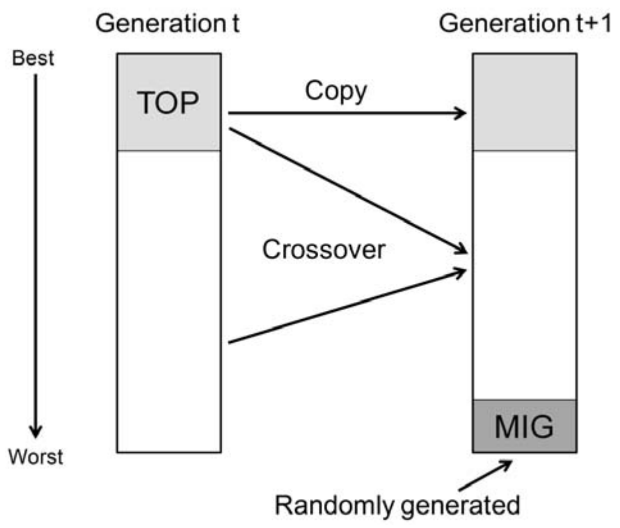

***Fig. 8:** Transition into the next generation (in analogy to Gonçalves, 2007)*

A mutation occurs to a chromosome with a certain probability. In the mutation operator two randomly chosen batches are deleted and the relevant customer orders, which have been included in the batches, are sorted according to their size in a non-increasing way. Applying the Best-Fit Rule the customer orders are then assigned to new batches. Afterwards, the non-TOP part is checked for identical chromosomes and duplicates are deleted. The population is completed by allowing the migration of some new randomly generated individuals (indicated by MIG in Fig. 8) into the population. Migration has a similar effect as mutation; it avoids premature convergence and ensures the diversity of the population.

After the algorithm has gone through all generations, it concludes with a local search procedure, which is applied to the individuals of the final population. It consists of two phases: a SWAP- and a SHIFT-phase. At first it is tried to improve the objective function value by interchanging one customer order from one batch with a customer order from another batch (SWAP). If no further improvement can be obtained it is tried to improve the objective function value by moving a customer order from one batch to another (SHIFT). The procedure terminates if no further improvements can be obtained by either SWAPs or SHIFTs. This local search procedure is identical to the one described in Henn et al. (2010).

4 Design of the Experiments

4.1 Warehouse Parameters

The warehouse layout considered in the numerical experiments is a single-block warehouse with two cross aisles and a depot located in the lower left corner. This layout type is identical to the one shown in Fig. 1 and is frequently used in numerical experiments (e.g. Henn et al., 2010; Petersen and Aase, 2004; de Koster et al., 1999a; Petersen and Schmenner, 1999). The warehouse consists of ten vertically orientated picking aisles. Each picking aisle has a capacity of 90 locations (45 on each side of the aisle), resulting in a total capacity of 900 storage location. Each storage location has a length of one length unit (LU). The necessary movement from the first and the last storage location of a picking aisle into the cross aisle amounts to one LU and the center-to-center distance between two aisles amounts to five LU. The depot is located 1.5 LU away from the first storage locations of the leftmost aisle.

In the warehouse, each article is located at one storage location only and the distribution of the articles within the warehouse follows a class-based storage assignment policy. For this purpose, the warehouse is grouped into three classes: Articles with the highest demand frequency (A) are located in the leftmost aisle, articles with a medium frequency (B) are located in the four subsequent aisles and articles with a low demand frequency (C) are located in the five rightmost aisles. Furthermore, it is assumed that 52% of the demand arises from articles in class A, 36% from B-articles and the remaining 12% from C-articles. Similar demand frequencies can be observed in real-world warehouses and have been considered in different numerical studies (de Koster et al., 1999a; Henn et al., 2010).

4.2 Problem Classes

For the numerical experiments, the number of customer orders is varied in five steps from 20 to 60 customer orders, where each step has a size of ten customer orders. The number of articles in a customer order is modeled as a random number which is uniformly distributed in {5, 6, …, 25}. The capacity of the picking device is defined by the maximum number of articles which can be placed on it. The values considered here are 30, 45, 60, 75. The underlying routing problem is solved by means of the S-Shape Heuristic. Table 1 summarizes the characteristics of the problem classes.

characteristic	specification
total number of customer orders (n)	20, 30, 40, 50, 60
capacity of the picking device (C) [no. of articles]	30, 45, 60, 75
routing strategy	S-Shape

Table 1: *Characteristics of the problem classes*

Combination of the different specifications results in 20 problem classes. For each problem class, 40 instances have been generated.

4.3 Benchmarks

The performance of the GGA will be compared to the performance of the IGA (i.e. the algorithm of Hsu et al. (2005)) with respect to solution quality and computing time. Both algorithms are evaluated in comparison to the CW(ii) heuristic, which serves as baseline for the solution quality. CW(ii) is chosen as baseline heuristic, since it has shown to provide good results in different warehouse settings (de Koster et al., 1999a). In order to allow for a fair comparison, the IGA is extended by the same local search procedure as it has been introduced for the GGA.

4.4 Algorithm Parameters

As explained in the previous chapter, genetic algorithms can be differentiated according to the realization of some structural aspects of genetic algorithms (cf. Section 3.1). The distinguishing features are summarized in Table 2.

feature	IGA	GGA
encoding scheme	real integers	real integers
orientation	item-oriented	group-oriented
decoding	direct representation	indirect representation
crossover	two-point crossover	two-point crossover
mutation	mutation	migration and mutation

Table 2: *Features of the genetic algorithms*

Similar to other experiments (e.g. Gonçalves, 2007) the size of the population will be adjusted proportionally to the size of the problem classes. In our case the number of individuals in the population will be set to four times the number of customer orders. All other parameters of the algorithms have been fixed across all problem classes. The number of generations has been set to 80 for both algorithms.

For the determination of the necessary parameter settings, a series of pre-tests has been carried out in order to determine a suitable combination of the parameters. For the IGA all combinations of the following parameters have been tested:

- Crossover probability $\in \{0.3, 0.4, 0.5\}$;
- Mutation probability $\in \{0.05, 0.1, 0.2\}$;

With respect to the GGA the following combinations were considered:

- TOP $\in \{0.10, 0.20, 0.30\}$;
- Mutation probability $\in \{0.10, 0.20, 0.30\}$;

The parameter configuration which provided the best objective function values is depicted in the following Table 3:

characteristic	IGA	GGA
size of the population	4*number of customer orders	
number of generations	80	
crossover probability	0.50	-*)
mutation probability	0.10	0.30
size of the TOP-part	-	0.10

*) The crossover probability for a certain chromosome cannot be defined explicitly, as the parental chromosomes are chosen randomly. However, if a chromosome is chosen as one of the parental chromosomes the crossover probability accounts for one.

Table 3: *Algorithm parameters*

4.5 Implementation and Hardware

The different algorithms have been implemented in C++ using the DEV Compiler Version 4.9.9.9.2. The numerical experiments have been carried out on a desktop PC with a 2.2 GHz Pentium processor and 2 GB RAM.

5 Results of the Experiments

5.1 Solution Quality

The results of the numerical experiments are summarized in Table 4. In comparison to the total tour length resulting from the application of C&W(ii), the IGA reduced the total length of the picker tours by 3.15% on average. The improvements ranged from 0.68% (no. of customer orders: n = 60; capacity of the picking device: C = 75) to 5.76% (n = 20; C = 75). By application of the GGA an average improvement of 3.75 % could be obtained. The improvements varied between 1.27% (n = 60; C = 75) and 6.47% (n = 20; C = 75). The group-oriented approach outperformed the item-oriented approach not only with respect to the average improvement, but also resulted in a superior average objective function value for every single problem class.

The superior results of the GGA can be explained by two aspects: On the one hand, in comparison to the IGA, the GGA generates solutions that incorporate – on average – a slightly smaller number of batches. A smaller number of batches tends to result in a smaller total tour length. On the other hand, the difference in the number of batches is not large enough to account completely for the differences in the solution quality. Therefore, it can be concluded that the main differences stem from the composition of the batches. The GGA is able to match customer orders better than the IGA does.

The improvements obtained by application of the two genetic algorithms are larger for problem classes that are characterized by a small or medium-size number of customer orders, or by a small or medium-size capacity of the picking device. For problem classes with a large number of customer orders and a large capacity of the picking device – e.g. problem class (n = 60; C = 60) or (n = 60; C = 75) – the improvements obtained from the two metaheuristics are small. This can be explained by the fact that the C&W(ii) heuristic performs best if a large number of customer orders is available that should be grouped into one batch. The good performance of C&W(ii) leaves little potential for improvements for the genetic algorithms.

n	C	C&W(ii)		IGA			GGA		
	[no. art.]	Ø TTL [LU]	Ø no. bat.	Ø TTL [LU]	Ø impr. [%]	Ø no. bat.	Ø TTL [LU]	Ø impr. [%]	Ø no. bat.
20	30	4360	11.3	4206	3.55	10.7	**4197**	3.77	10.7
	45	2855	7.1	2745	3.74	6.9	**2700**	5.32	6.8
	60	2283	5.3	2184	4.31	5.3	**2167**	5.02	5.3
	75	1878	4.3	1769	5.76	4.2	**1756**	6.47	4.2
30	30	6426	16.5	6225	3.09	15.8	**6195**	3.59	15.7
	45	4333	10.7	4152	4.11	10.3	**4119**	4.86	10.3
	60	3288	7.8	3149	4.17	7.7	**3125**	4.90	7.6
	75	2715	6.3	2604	3.95	6.1	**2582**	4.77	6.1
40	30	8219	21.2	7945	3.52	20.2	**7926**	3.60	20.2
	45	5504	13.8	5308	3.69	13.3	**5273**	4.17	13.1
	60	4282	10.4	4113	4.11	10.0	**4088**	4.44	9.9
	75	3479	8.2	3377	3.00	8.1	**3347**	3.75	8.0
50	30	10509	27.0	10186	3.24	25.9	**10124**	3.69	25.7
	45	6760	16.9	6608	2.32	16.6	**6580**	2.65	16.6
	60	5254	12.8	5154	1.92	12.7	**5120**	2.50	12.6
	75	4325	10.3	4254	1.70	10.1	**4219**	2.43	10.1
60	30	12145	31.6	11721	3.56	30.1	**11674**	3.82	30.0
	45	7930	20.1	7803	1.58	19.8	**7755**	2.13	19.7
	60	6147	15.1	6086	0.98	15.0	**6036**	1.75	14.8
	75	5029	12.1	4997	0.68	12.0	**4964**	1.27	11.9
Minimum			4.30		0.68	4.20		1.27	4.20
Average			13.44		3.15	13.04		3.75	12.97
Maximum			31.60		5.76	30.10		6.47	30.00

"n": number of customer orders; "C [no. art.]": capacity of the picking device in number of articles; "Ø TTL [LU]": average total tour length of picking tours in length units; "Ø impr. [%]": improvement obtained by the respective algorithm in comparison to the C&W(ii) solution in percent; "Ø no. bat.": average number of batches. For each problem class the best obtained average total tour length is highlighted bold.

Table 4: *Solution quality*

The development of the solution quality of the two genetic algorithms as a function of the number of generations is given in Fig. 9. The evolution is shown exemplarily for the four problem classes (n = 20; C = 30), (n = 30; C = 60), (n = 50; C = 30), and (n =

60; C = 75). The average deviation of the objective function value from the best objective function value obtained at the end of the local search phase is depicted on the y-axis. The number of generations (1-80) is presented on the x-axis; the last entry (i.e. 81) on the x-axis corresponds to the termination of the algorithm after the local search phase. The values in the graph represent the average deviation calculated across all 40 instances. Fig. 9 demonstrates two important differences between the two algorithms: The GGA needs fewer generations than the IGA in order to identify good solutions. Moreover the quality of the solutions resulting from the application of the “pure” genetic algorithm (i.e. before the local search is applied) is much better for the group-oriented approach than for the item-oriented approach. This is demonstrated by the different step-sizes after generation 80. Application of the local search phase to the final generation of the GGA resulted in an additional average improvement between 0.3 % (n = 20; C = 30) and 5 % (n = 60; C = 75) whereas the application of the local search phase in the IGA resulted in additional improvements between 1.1 % (n = 20; C = 30) and 9.8 % (n = 60; C =75).

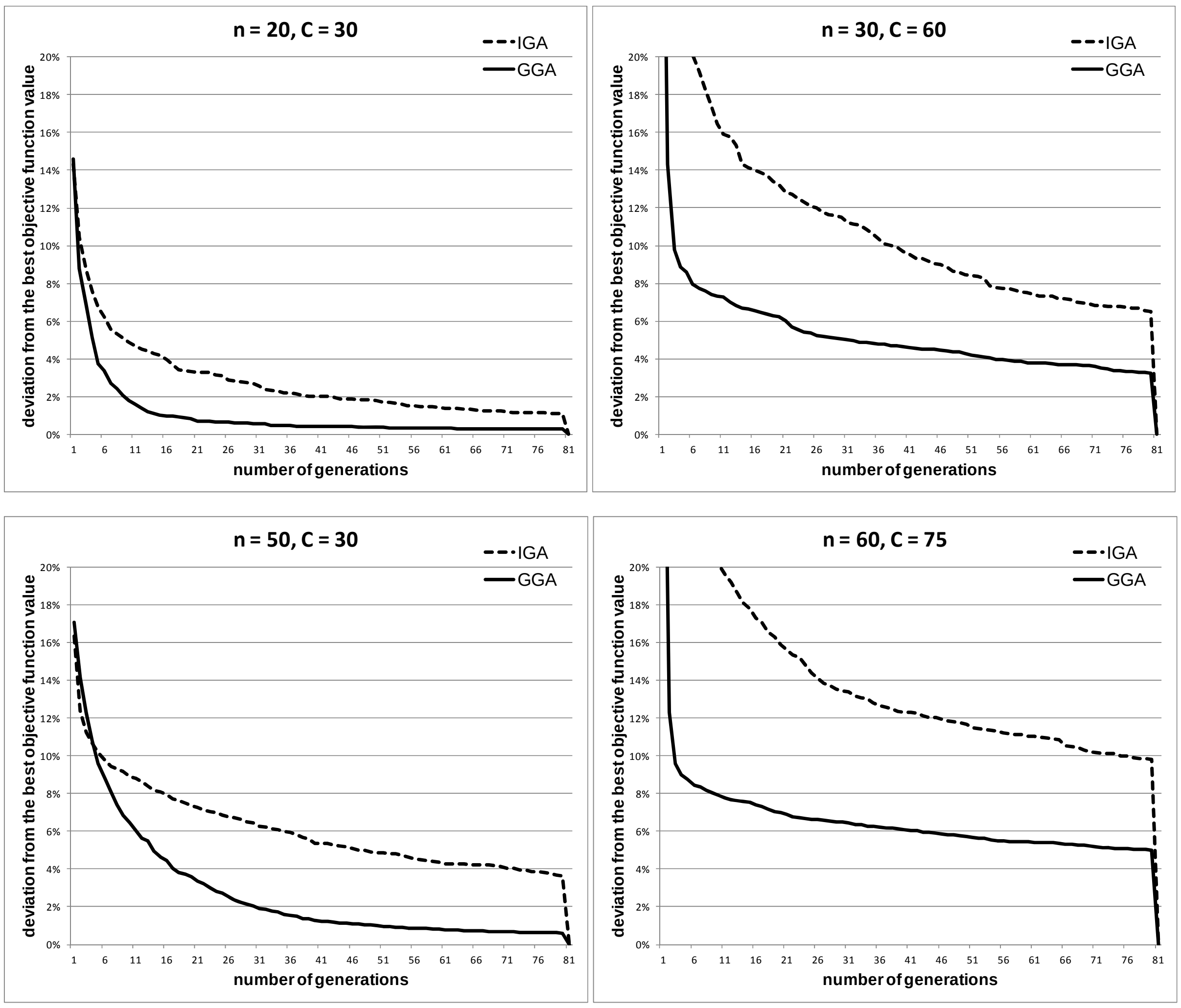

Fig. 9: Development of the solution quality over the generations for different problem classes

5.2 Computing Times

The computing times for the two genetic algorithms are depicted in Table 5. Those for the C&W(ii) algorithm are not listed here, since they can be considered as negligible (<1 sec.). Table 5 demonstrates that the computing times are determined by two parameters, namely by the number of customer orders and by the size of the capacity of the picking device; an increase of each of the two parameters leads to an increase of the computing times of the algorithms. Comparison of the computing times of the two algorithms reveals that the item-oriented approach requires less computing time for small and medium-size problems of up to 40 customer orders. For a larger number of customer orders and for larger capacity of the picking device, in particular, the group-oriented approach consumes less time.

n	C [no. art.]	IGA Ø time [sec.]	GGA Ø time [sec.]
20	30	3.1	7.0
	45	4.2	7.9
	60	4.5	9.0
	75	4.5	9.8
30	30	9.1	17.3
	45	15.0	20.8
	60	17.2	24.5
	75	16.5	27.1
40	30	22.5	34.6
	45	39.5	44.8
	60	48.9	53.7
	75	49.6	59.9
50	30	46.7	60.5
	45	87.4	86.2
	60	117.2	109.4
	75	109.2	116.5
60	30	94.8	99.6
	45	174.8	150.2
	60	230.3	187.2
	75	242.4	207.6

"n": number of customer orders; "C [no. art.]": capacity of the picking device in number of articles; "Ø time [sec.]": average computing time in seconds per instance.

Table 5: *Computing times*

In general the computing times of both algorithms are reasonable, with a maximum of four minutes for the IGA and three and a half minutes for the GGA, both for the most challenging problem class (n = 60; C = 75).

6 Summary and Outlook

The paper has dealt with the Order Batching Problem, a problem arising in the context of order picking in distribution warehouses. Order batching is of pivotal importance for the efficient management of warehouse processes, since improved order batching can result in a significant reduction of the delivery lead times and in a reduction of the operating cost of a warehouse. In this paper two population-based metaheuristics, an item-oriented genetic algorithm and a group-oriented genetic algorithm have been presented for the solution of the Order Batching Problem. The application of both algorithms results in a significant reduction of the total tour length of the order pickers with reasonable computing times. However, the application of the group-oriented genetic algorithm resulted in improvements that are larger than those of the item-oriented one. These results demonstrate that it is very important to choose a genetic algorithm that is suitable for the problem in order to obtain the best results possible. For the Order Batching Problem considered in this paper, it could be shown that a group-oriented approach is the best possibility to represent the problem structure.

Interesting aspects for further research could be the incorporation of due dates for customer orders or the extension of order batching to the wave picking concept.

References

Albareda-Sambola, M.; Alonso-Ayuso, A.; Molina, E.; de Bas, C.S. (2009):
Variable Neighborhood Search for Order Batching in a Warehouse. In: *Asia-Pacific Journal of Operational Research* 26(5), 655-683.

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

Choe, K.; Sharp, G. (1991):
Small Part Order Picking: Design and Operation. Available online at: http://www2.isye.gatech.edu/~mgoetsch/cali/Logistics%20Tutorial/order/article.htm

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

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

de Koster, R.; Roodbergen, K.J.; van Voorden, R. (1999b):
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.

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. In: *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. In: *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. In: *International Journal of Production Research* 27(7), 1097-1114.

Falkenauer, E. (1992):
The Grouping Genetic Algorithms: Widening the Scope of the GAs. In: *JORBEL – Belgian Journal of Operations Research, Statistics and Computer Science* 33(1,2), 79-102.

Falkenauer, E. (1995):
Tapping the Full Power of Genetic Algorithm through Suitable Representation and Local Optimization: Application to Bin Packing. In: Biethahn, J.; Nissen, V. (eds.): Evolutionary Algorithms in Management Applications. Springer, Berlin et al., 167-181.

Falkenauer, E. (1996):
A Hybrid Grouping Genetic Algorithm for Bin Packing. In: *Journal of Heuristics* 2(1), 5-30.

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

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

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

Goldberg, D.E. (1989):
Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Gonçalves, J.F. (2007):
A Hybrid Genetic Algorithm-heuristic for a Two-dimensional Orthogonal Packing Problem. In: *European Journal of Operational Research* 183(3), 1212-1229*.*

Henn, S.; Koch, S.; Doerner, K.; Strauss, C.; Wäscher, G. (2010):
Metaheuristics for the Order Batching Problem in Manual Order Picking Systems. In: *BuR - Business Research* 3(1), 82-105.

Henn, S.; Wäscher, G. (2010):
Tabu Search Heuristics for the Order Batching Problem in Manual Order Picking Systems. *Working Paper No. 07/2010*, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg.

Henn, S.; Koch, S.; Wäscher, G. (2011):
Order Batching in Order Picking Warehouses: A Survey of Solution Approaches. *Working Paper No. 01/2011*, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg.

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

Holland, J.H. (1975):
Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

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

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

Petersen, C.G.; Aase, G. (2004):
A Comparison of Picking, Storage, and Routing Policies in Manual Order Picking. In: *International Journal of Production Economics* 92(1), 11-19.

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

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

Roodbergen, K.J. (2001):
Layout and Routing Methods for Warehouses, Ph.D. thesis, Erasmus University Rotterdam.

Tompkins, J.A.; White, J.A.; Bozer, Y.A.; Frazelle, E.; Tanchoco, J.M.A. (2003): Facilities Planning. 3rd ed., Wiley, 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. In: *International Journal of Production Research* 46(22), 6533-6555.

van Nieuwenhuyse, I.; de Koster, R. (2009):
Evaluating Order Throughput Time in 2-block warehouses with Time Window Batching. In: *International Journal of Production Economics* 121(2), 654-664.

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, 324-370.

Weicker, K. (2007):
Evolutionäre Algorithmen. 2nd ed., Teubner, Wiesbaden.

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

Tel.: +49 (0) 3 91 / 67-1 85 84
Fax: +49 (0) 3 91 / 67-1 21 20

www.fww.ovgu.de/femm

ISSN 1615-4274