WORKING PAPER SERIES

Metaheuristics for Order Batching and Sequencing in Manual Order Picking Systems

Sebastian Henn/Verena Schmid

Working Paper No. 11/2011

Impressum (§ 5 TMG)

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

Verantwortlich für diese Ausgabe:
Sebastian Henn
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

Metaheuristics for Order Batching and Sequencing in Manual Order Picking Systems

Sebastian Henn[1], Verena Schmid[2]

June 2011

Abstract

Order picking deals with the retrieval of articles from their storage locations in order to satisfy customer requests. A major issue in manual order picking systems concerns of the transformation and consolidation of customer orders into picking orders (order batching). In practice, customer orders have to be completed by certain due dates in order to avoid delay in the shipment to customers or in production. The composition of the picking orders, their processing times and the sequence according to which they are released have a significant impact on whether and to which extent given due dates are violated. This paper presents how metaheuristics can be used in order to minimize the total tardiness for a given set of customer orders. The first heuristic is based on Iterated Local Search, the second one is inspired by the Attribute-Based Hill Climber, a heuristic based on a simple tabu search principle. In a series of extensive numerical experiments, the performance of these metaheuristics is analyzed for different classes of instances. We will show that the proposed methods provide solutions which may allow for operating order picking systems more efficiently. Solutions can be improved by 46% on average, compared to the ones obtained by standard constructive heuristics such as an application of the Earliest Due Date rule.

Keywords: Warehouse Management, Order Batching, Batch Sequencing, Due Dates, Iterated Local Search, Attribute-Based Hill Climber

[1] Faculty of Economics and Management, Otto-von-Guericke University Magdeburg, Germany

[2] Faculty of Business, Economics and Statistics, University of Vienna, Austria

Corresponding author:
Dipl.-Math. oec. Sebastian Henn
Otto-von-Guericke University Magdeburg, Faculty of Economics and Management
Postbox 4120, 39016 Magdeburg, Germany
Phone: +49 391 67 11841
Fax: +49 391 67 18223
Email: sebastian.henn@ovgu.de

Contents

1 Introduction

Order picking is a warehouse function dealing with the retrieval of articles from their storage locations in order to satisfy a given demand specified by customer requests (Petersen and Schmenner, 1999). Order picking arises because incoming articles are typically received and stored in (large-volume) unit loads while (internal or external) customers order small volumes (less-than-unit loads) of different articles.

Despite that there have been several attempts to automate the picking process, order picking systems still involve human operators on a large scale. Among such manual order picking systems, picker-to-parts systems can be considered as the most important ones. In these systems order pickers walk (or ride) through the warehouse and collect the requested articles (Wäscher, 2004). For an efficient organization of the corresponding picking operations, order batching, i.e. the grouping of customer orders into picking orders (batches), has been proven to be pivotal (de Koster et al., 2007).

In order picking systems customer orders often have to be completed by certain due dates. The composition of the batches, their processing times (mainly determined by the length of the picker tours) and the sequence according to which the batches are processed determine whether these due dates can be met. In order to avoid delays, the tardiness between due dates and the completion of customer orders should be as small as possible. For this purpose the Order Batching and Sequencing Problem (OBSP), in which the tardiness for a set of customer orders should be minimized, has to be solved.

Henn and Wäscher (2010) and Henn et al. (2010) have shown that minimization of the picking tour length by means of local search-based metaheuristics, namely Iterated Local Search (ILS) and Attribute-Based Hill Climber (ABHC), may allow for operating distribution warehouses substantially more efficient. This article presents how these metaheuristics can be adapted for the OBSP.

The remainder of this paper is organized as follows: In the following section, we give a brief introduction into order picking systems and describe the OBSP precisely. Moreover, we present a mathematical model for this problem. In Section 3, the related literature for the OBSP is reviewed. In the subsequent sections 4 and 5 the implementation for ILS and ABHC is described. Extensive numerical experiments have been carried out in order to evaluate the performance of ILS and ABHC. Section 6 is dedicated to the design of these experiments (including the description of warehouse parameters, algorithm parameters, and problem classes). The results of the experiments are presented and analyzed for different problem classes in Section 7. The article concludes with a summary and an outlook on further research directions.

2 Order Batching and Sequencing Problem

2.1 Problem Description

A customer order is composed of order lines, where each order line consists of a particular article (item type) and the corresponding requested quantity. Those order lines which should be processed together are summarized in a pick list. This list may enclose the order lines of a single customer order (pick-by-order) or of a combination of customer orders (pick-by-batch). Moreover, the list guides the order pickers through the picking area. The tours of the order pickers (who start at the depot, proceed to the respective storage locations, and return to the depot) are usually determined by routing strategies. Since order pickers seem to accept only straightforward and non-confusing routing schemes, tours provided by the S-Shape- or Largest Gap-Heuristic are prevalent in practice (de Koster et al., 1999). For a single-block warehouse layout these routing strategies are visualized in Figure 2.1 for a single-block warehouse layout. The black rectangles symbolize the locations (called pick locations) where items have to be picked on the respective tours through the picking area. Tours generated by the *S-Shape-Heuristic* are characterized as follows: The order picker enters an aisle if at least one requested item is located in that aisle and traverses it completely. Afterwards, the order picker proceeds to the next aisle which contains a requested item. An exception to that rule may arise in the last aisle containing an item to be picked: if the order picker is positioned in the front cross-aisle, he/she would pick the items of the last aisle and would return to the depot along the front aisle. The *Largest Gap-Heuristic* gives a tour in which the order picker always traverses the rightmost and leftmost aisle entirely which contains an item to be picked. All other aisles can be entered from the front and back aisle in such a way that the non-traversed distance is maximal. For the collection of the requested items, order

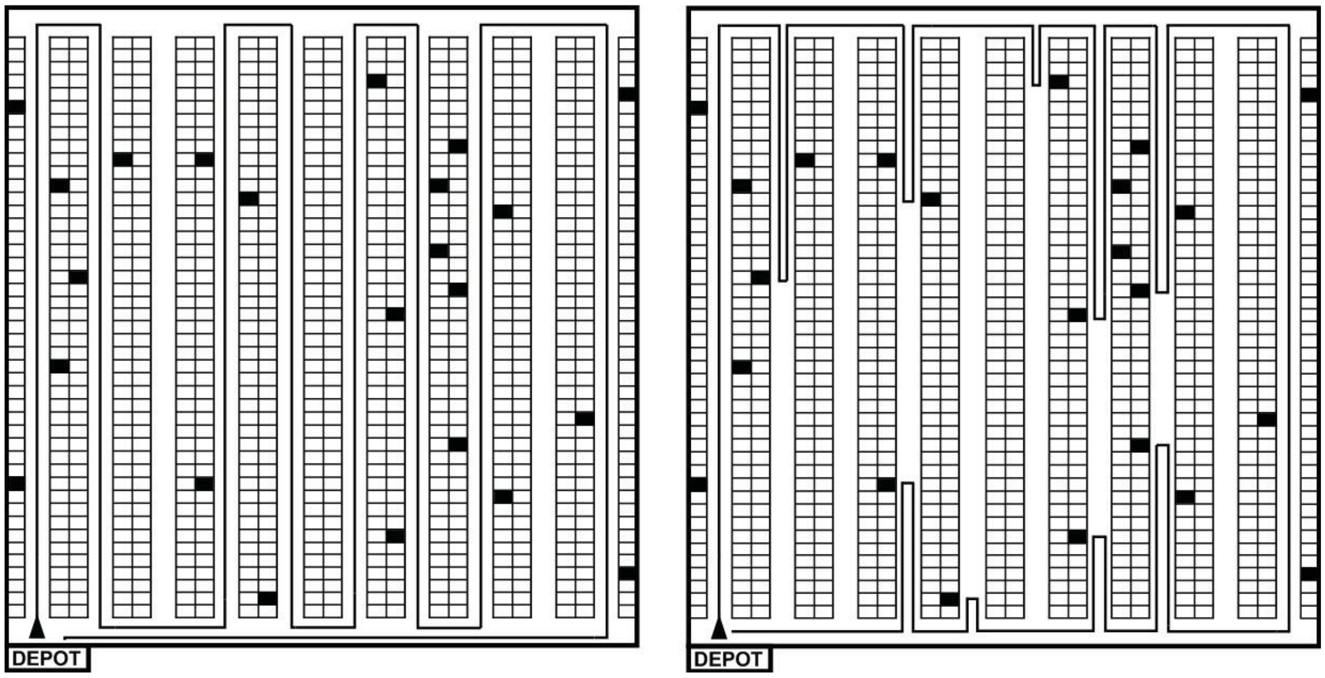


Figure 2.1: Example of S-Shape routing (left) and Largest Gap routing (right) in a single-block layout

pickers typically use a picking device (e.g. a cart or a roll pallet). When combining customer orders into batches one has to ensure that the capacity of the picking device is not exceeded. Splitting of customer orders is usually not permitted since it would result in an additional, not acceptable sorting effort. With respect to the availability of information for all customer orders, order batching can be distinguished into static and dynamic batching (Yu and de Koster, 2009). In the static case, which is discussed in this paper, all customer orders are known in advance. In the dynamic case customer orders may arrive at different points in time.

Given a particular picking order, the time the order picker requires to complete a tour is called *(batch) processing time*. This time consists of the following elements:

- *travel time*, i.e. the time period the order picker spends traveling from the depot to the first

pick location, between the pick locations and from the last pick location to the depot;
- *search time*, i.e. the time period required for the identification of items;
- *pick time*, i.e. the time period needed for moving items from the corresponding locations onto the picking device;
- *setup time*, i.e. the time period consumed by administrative and setup tasks at the beginning and end of each tour.

Among these components, the travel time is of outstanding importance, since it consumes the major proportion (at least 50%) of the total processing time (Tompkins et al., 2003).

In practice, the customer orders have to be completed by certain due dates. In distribution warehouses, e.g. due dates have to be met in order to guarantee the scheduled departure of trucks, which deliver the requested items to the (external) customers (Gademann et al., 2001). In material warehouses which provide the input to a production system (internal customers), on-time retrievals from the warehouse are vital in order to avoid production delays. In theses cases, instead of measuring the quality of a solution by means of the total processing time, the batching of customer orders into picking orders has to be evaluated with respect to the tardiness of the customer orders (Elsayed et al., 1993). The tardiness of a customer order is defined as the positive difference between the completion time of a customer order and its due date. The point in time when the order picker returns from his/her tour to the depot after having collected all items is called (batch) completion time. The completion time of a customer order is identical to the completion time of the batch the order is assigned to. The composition of the batches, the sequence according to which the batches are processed and the corresponding release times (i.e. the points in time when the various batches are started) determine whether and how due dates can be met. The described situation gives rise to the following problem (Order Batching and Sequencing Problem; OBSP): How, given a routing strategy, storage locations and a picking device of fixed capacity, can a given set of customer orders with known due dates be combined (batched) into picking orders such that the total tardiness of all customer orders is minimized?

In the following, we consider for our OBSP a picking area in which only one order picker operates. Therefore, all batches have to be started successively. A solution of the OBSP can therefore be interpreted as sequence of batches. For a solution, the order picker starts to collect the customer orders included in the first batch of the sequence (the batch at position #1). After the order picker returns to the depot he/she collects the customer orders of the second batch of the sequence (the batch at position #2) and so on.

The OBSP is closely related to the (static) Order Batching Problem (OBP), which has been discussed frequently in literature. This problem consists of the assignment of customer orders to batches in order to minimize the total processing time. In this case it is only necessary to determine an optimal composition of the batches. In the OBSP, however the determination of an optimal sequence according to which the batches are scheduled is also required.

2.2 Model Formulation

For the above defined OBSP, we propose the following mathematical optimization model based on the set partitioning problem. This model uses the following index sets and constants:

n:	number of customer orders;
J:	set of customer orders, where $J = \{1, \dots, n\}$;
C:	capacity of the picking device;

c_j:	capacity utilization required by customer order j ($j \in J$);
d_j:	due date of customer order j ($j \in J$);
I:	set of all feasible batches.

Let $a_i = (a_{i1}, \dots, a_{in})$ be a vector of binary indicators a_{ij}, stating whether customer order j ($j \in J$) is included in batch i ($a_{ij} = 1$) or not ($a_{ij} = 0$). Each feasible batch i is characterized by the fact that the capacity constraint is not violated, therefore

$$\sum_{j \in J} c_j \cdot a_{ij} \le C \tag{2.1}$$

holds. Additionally, the following sets and constants are introduced:

$\overline{m}$:	upper bound on the number of batches, which have to be scheduled; this results in at most $\overline{m}$ positions at which the order picker can start a batch;
K:	set of positions at which a batch can be scheduled, where $K = \{1, \dots, \overline{m}\}$;
M:	a sufficiently large (positive) number;
pt_i :	processing time of batch i ($i \in I$).

Furthermore, the following variables are introduced:

x_{ik} :	binary decision variable which is equal to 1 if and only if batch i ($i \in I$) is scheduled at position k ($k \in K$);
y_{jk} :	binary decision variable which is equal to 1 if and only if order j ($j \in J$) is assigned to the batch which is scheduled at position k ($k \in K$);
ct_k :	completion time of the batch scheduled at position k ($k \in K$);
ta_{jk} :	tardiness of customer order j ($j \in J$) if included in the batch scheduled at position k ($k \in K$).

The optimization model can then be stated as follows:

$$\min \sum_{j \in J} \sum_{k \in K} ta_{jk} \tag{2.2}$$

$$\text{s.t.} \sum_{i \in I} x_{ik} \le 1, \forall k \in K; \tag{2.3}$$

$$\sum_{k \in K} \sum_{i \in I} a_{ij} x_{ik} = 1, \forall j \in J; \tag{2.4}$$

$$\sum_{i \in I} a_{ij} x_{ik} = y_{jk}, \forall j \in J, \forall k \in K; \tag{2.5}$$

$$ct_{k-1} + \sum_{i \in I} pt_i x_{ik} \le ct_k, \forall k \in K \setminus \{1\}; \tag{2.6}$$

$$\sum_{i \in I} pt_i x_{i1} \le ct_1; \tag{2.7}$$

$$ct_k - ta_{jk} \le d_j + M(1 - y_{jk}), \forall j \in J, \forall k \in K; \tag{2.8}$$

$$ta_{jk} \ge 0, \forall j \in J, \forall k \in K; \tag{2.9}$$

$$x_{ik} \in \{0, 1\}, \forall i \in I, \forall k \in K; \tag{2.10}$$

$$y_{jk} \in \{0, 1\}, \forall j \in J, \forall k \in K. \tag{2.11}$$

The objective function (2.2) represents the total tardiness of all customer orders. It is guaranteed by constraints (2.3) that at most one batch is assigned to a particular position. Equations (2.4) ensure that each customer order is assigned to exactly one batch. Equations (2.5) ensure consistency among batches and orders scheduled at any particular position. Conditions (2.6) ensure feasibility with respect to time and guarantee that two batches do not overlap. Inequality (2.7) is necessary to determine the completion time of the batch at position 1. Constraints (2.8) restrict the tardiness of each customer order j ($j \in J$), which is defined by the number of time units between its due date and the resulting completion time. Constraints (2.9) impose non-negativity on all decision variables modeling tardiness.

In this model, the number of possible batches and, therefore, the number of binary variables grows exponentially with the number of orders. For a small-sized problem instance, e.g. with $n = 20$ and in which at most five orders can be assigned to one batch, the number of feasible batches amounts to 21,699. In this instance, $\overline{m}$ is equal to 20 and the model consists of 860 constraints and 434,800 variables, of which 434,380 are binary.

Du and Leung (1990) have shown that the problem of minimizing the total tardiness for a set of independent jobs on a single machine is $\mathcal{NP}$-hard. This problem can be interpret as a special case of the OBSP, in which the capacity of the picking device is equal to only one customer order. Therefore, we can conclude that the OBSP is also $\mathcal{NP}$-hard and the use of heuristics becomes unavoidable.

3 Literature Review

For the OBP a variety of heuristic solution approaches exists. They can be partitioned into four groups: priority rule-based algorithms, seed algorithms, savings algorithms, and metaheuristics. The first three groups contain approaches which are constructive by nature, whereas metaheuristic focus on improving given solutions. For metaheuristics either approaches based on the local search principle (Iterated Local Search, Tabu Search, Variable Neighborhood Search) or population-based metaheuristics (Genetic Algorithms, Ant Colony Optimization) have been suggested. In several numerical experiments it was shown that metaheuristics can improve the results obtained by constructive algorithms significantly (Henn et al., 2012).

For the solution of the OBSP in an automated storage and retrieval system (AS/AR-system), Elsayed and Lee (1996) proposed a two-step solution approach for the construction of batches and the sequence according to which these batches are released. At first, the customer orders are sorted according to their due dates and the times which would be needed if each customer order is processed in a single batch. A first upper bound on the optimal total tardiness can be obtained by assuming that the items of each customer order would be collected on a single tour and by scheduling the tours according to the position of the order in the sequence. In order to improve this bound, decision rules for the generation of batches are proposed and evaluated. The Nearest-Schedule Rule determines the first customer order of the obtained sequence as the seed order and assigns it to the first batch. Additional customer orders of the sequence are assigned to the seed if the inclusion does not increase the total tardiness and does not violate the capacity restriction. When no further customer order can be added, the first unassigned customer order in the sequence serves as seed order for the next batch. In numerical experiments the Nearest-Schedule Rule outperformed all other proposed rules. Additionally, the authors showed how storage orders (items which have to be carried to the storage locations) can be included in the obtained sequence.

Elsayed et al. (1993) suggested a solution method for the problem of minimizing the total tardiness and earliness of all customer orders in an AS/AR-system. Their approach consists of three steps. First, priorities are determined for the customer orders. Each customer order is considered as a single batch and its priority value is defined by the weighted sum of its due date and the corresponding processing time. Customer orders are ranked in ascending order of their priority index. Based on this sequence the objective function value is determined. This sequence is updated, if a pairwise exchange of two adjacent customer orders can be identified which would improve the objective function value. In step 2, customer orders are combined into batches. For each customer order it is determined whether separate picking or picking in combination with other customer orders is favorable. In the latter case the customer order is assigned to the first combination which results in a smaller objective function value. For each of the obtained batches, release times are determined (Step 3). It has to be noted that an idle time may occur between the start time of a batch and the completion time of its predecessor. As a consequence the algorithm shifts batches along the time axes in order to obtain improved objective function values by reducing the total earliness, since earliness is penalized as well.

The solution approaches proposed by Elsayed and Lee (1996) and Elsayed et al. (1993) are of limited applicability to manual picker-to-parts systems. In contrast to the situation in AS/AR-systems, here the processing time of a batch containing only one customer order differs significantly from the processing times of a batch to which more than one customer order are assigned. Both solution approaches use the single processing times in order to determine a sequence of batches. Therefore, they will not yield competitive results in the situation described in this article.

Tsai et al. (2008) considered an order batching and sequencing problem where the total costs (depending on the total travel time) have to be minimized and both earliness and tardiness are penalized. The batching problem is solved by means of a genetic algorithm. In their implementation, a solution is represented by a sequence of integers. This sequence contains the indices of the batches to which the items requested in the customer order have been assigned. In order to determine the fitness of the generated solutions, a Traveling Salesman Problem is solved by another genetic algorithm. Unlike in the approaches previously discussed, the authors permit the splitting of orders; therefore, the items required by a single customer order may be collected on different tours. Due to this specific condition under which the algorithm can be applied, the approach will not be considered here, any further.

4 Iterated Local Search

4.1 General Principle

Iterated Local Search (ILS) has been applied successfully to a variety of different optimization problems, such as the Traveling Salesman Problem (Katayama and Narihisa, 1999) and the OBP (Henn et al., 2010). The main principle can be described as follows (Lorenço et al., 2010): 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 the heuristic starts from an initial solution $s_0 \in S$ and determines a sequence of l feasible solutions $s_0, s_1, \ldots, s_l = \hat{s}$. Each element s_j ($j \in \{1, \ldots, l\}$) is a neighbor of its predecessor and s_j improves the objective function with respect to s_{j-1}. Provided the problem is bounded, $\hat{s}$ is a local optimum.

ILS explores the vicinity of the obtained local optimum $\hat{s}$ (used as an incumbent solution) more closely in order to identify a solution with an improved objective function value. Therefore, during the perturbation phase, the incumbent solution is partially modified (perturbed) and a further local search phase is applied to this modified solution s. The new solution $\hat{s}$ stemming from the local search phase has to pass an acceptance criterion in order to become the new incumbent solution, otherwise the previous solution remains the incumbent solution for a further iteration. These two phases are repeated until a termination condition is met. Algorithm 4.1 summarizes the general principle of ILS.

Algorithm 4.1 Iterated Local Search: General Principle

```
Input: initial solution $s_0$;
1: $s_{\text{incumbent}}$ := local_search($s_0$);
2: $s^\star$ := incumbent;
3: repeat
4:     $s$ := perturbation($s_{\text{incumbent}}$);
5:     $\hat{s}$ := local_search($s$);
6:     $s_{\text{incumbent}}$ := acceptance_criterion($s_{\text{incumbent}}$, $\hat{s}$, history);
7:     update best solution $s^\star$ found so far;
8: until termination condition is met;
```

4.2 Adaption for the OBSP

Based on the adaption of ILS to the OBP by Henn et al. (2010), this general principle has been modified to the OBSP as follows: An initial solution is generated by means of the Earliest Due Date (EDD) rule. In this rule all orders are sorted according to their due dates in a non-increasing order. According to this sequence, the orders are assigned to batches successively, ensuring that the capacity of the picking device is not violated.

For the local search phase two operators have been implemented: Two solutions are called neighbors if both solutions can be achieved from each other by interchanging two orders from different batches (*swap move*) or by assigning one order to a different batch (*shift move*). For the execution of these transformations a list of the customer orders which have been assigned to a batch is considered. The sequence in which the customer orders appear on this list is set up in the chronological order in which the customer orders were assigned to the batch. A swap move is applied in a way that the exchanged orders are taking exactly the same positions in the new batches as their counterparts occupied before. In the case of a shift move the chosen customer order is assigned to the end of the new batch.

In the local search phase, a systematic first improvement strategy is used. Solutions are tried to to be improved by a sequence of swap moves. If an improvement can be obtained, one proceeds from the new solution and searches for another improvement by a swap move. In case that no swap move can be identified which improves the objective function value, it is tried to achieve an improvement by a sequence of shift moves. If no further improvement by shift moves can be identified, the algorithm searches again for a swap move that leads to an improved solution. These steps are repeated until no further improvements by swap moves and shift moves are possible.

In the perturbation phase two different batches k and l are selected randomly and a random number q of orders from batch k is moved to batch l, and vice versa (q is bounded by at most 50% of the number of orders currently contained in k and l, respectively). Should this rearrangement violate the capacity constraint, then all remaining orders will be assigned to a new batch. The number of rearrangements is an important issue: If the number of exchanges is too small, the algorithm would probably fall back into the same local minimum. On the other hand, if this number is too high, it would not be able to intensify the search in a good solution subspace. Therefore, the number of rearrangements is limited by $m^\star \cdot \lambda + 1$, where $m^\star$ corresponds to the number of batches in the best known solution and the constant λ is called rearrangement factor.

A new solution will always be accepted as an incumbent solution if its total tardiness is smaller than the best currently known one. Moreover, we also accept deteriorating solutions, if the incumbent solution did not improve the global best solution within a predefined number of iterations. The solution obtained in the last local search phase will be chosen as new incumbent solution, if the difference between the total tardiness of the last solution $tar(s)$ and total tardiness of the best known solution $tar(s^\star)$ is smaller than a threshold $\mu \cdot tar(s^\star)$, where the threshold factor μ is in $[0, 1]$. Otherwise the incumbent solution will be perturbated again.

5 Attribute-Based Hill Climber

5.1 General Principle

The Attribute-Based Hill Climber (ABHC) is an almost parameter-free heuristic based on a simple tabu search principle. It was developed by Whittley and Smith (2004), who applied it successfully to the Traveling Salesman Problem and to the Quadratic Assignment Problem. Derigs and Kaiser (2007) proposed an application of ABHC to the Vehicle Routing Problem and showed that the quality of the generated solutions is competitive with that of solutions generated by other state-of-the-art heuristics. Also for the Open Vehicle Routing Problem ABHC generated high-quality solutions (Derigs and Reuter, 2009). Henn and Wäscher (2010) have applied ABHC to the OBP and obtained solutions with improved tour lengths compared to other local search-based heuristics.

The main principle of ABHC can be described as follows: For each problem a set of attributes $\mathbf{A} = \{A_1, \ldots, A_q\}$ is introduced. An attribute can be any feature to be found in a solution, e.g. the traversed edges in a solution of a Traveling Salesman Problem. In order to describe whether a solution 'contains' an attribute A_k ($k \in \{1, \ldots, q\}$) or not, a Boolean function $a_k : s \mapsto$ {'true', 'false'} is defined where

$$a_k(s) = \begin{cases} \text{true} & \text{if } s \text{ contains attribute } A_k, \\ \text{false} & \text{else.} \end{cases} \tag{5.1}$$

For each solution s ($s \in S$) and each attribute A_k ($k \in \{1, \ldots, q\}$), $tar_k(s)$ can be defined as

$$tar_k(s) = \begin{cases} tar(s) & \text{if } a_k(s) = \text{'true'}, \\ \infty & \text{else,} \end{cases} \tag{5.2}$$

where $tar(s)$ represents the objective function value of solution s. ABHC performs a local search, in which solutions $s_0, s_1, s_2, \ldots$ are visited. Let s_i be the solution visited in iteration i, then for each attribute k the best objective function value $\text{best}(i,k)$ for a solution containing this attribute is updated,

$$\text{best}(i,k) := \begin{cases} tar_k(s_0) & \text{for } i = 0, \\ \min\{\text{best}(i-1,k), tar_k(s_i)\} & \text{for } i \geq 1. \end{cases} \tag{5.3}$$

A solution can be accepted if and only if it represents the best solution found so far for at least one attribute that is 'contained' in the solution. In each iteration i the set of acceptable solutions can be defined as

$$S_i := \{s | \exists k \in \{1, \ldots, q\} \text{ where } a_k(s) = \text{'true' and } tar(s) < \text{best}(i,k)\}. \tag{5.4}$$

The algorithm starts with an initial solution s_0 and searches for a new incumbent solution in each iteration i. For this solution only neighbors of the incumbent solution (elements of $\mathcal{N}(s_i)$) are considered which possess for at least one containing attribute the best found objective function value (elements of S_i). Among this set $\mathcal{N}(s_i) \cap S_i$ an element with the smallest objective function value is chosen as new incumbent solution s_{i+1}, and the values $\text{best}(i,k)$ ($\forall k \in \{1,\ldots,q\}$) are updated according to (5.3).

The algorithm stops if the current neighborhood does not contain a solution that represents a best solution for at least one containing attribute, i.e. if $\mathcal{N}(s_i) \cap S_i = \emptyset$. Since only solutions which represent the best solution found so far for at least one attribute that is 'contained' in the solution can be accepted during the search, an incumbent solution can never be accepted as incumbent solution in a later iteration. Since the number of solutions and the number of attributes are finite, ABHC terminates after a finite number of iterations. The general principle of ABHC is summarized in Algorithm 5.1.

Algorithm 5.1 Attribute-Based Hill Climber: General Principle

```
Input: initial solution $s_0$;
1: $i := 0$;
2: $s^\star := s_0$;
3: determine $\text{best}(0,k)$;
4: while $N(s_i) \cap S_i \neq \emptyset$ do
5:     $s_{i+1} := \arg\min\{tar(s) | s \in N(s_i) \cap S_i\}$;
6:     $i := i + 1$;
7:     update $\text{best}(i,k)$;
8:     update best solution $s^\star$ found so far;
9: end while
```

5.2 Adaption for the OBSP

Only three design decisions have to be made: the choice of the initial solution, the neighborhood structure, and the set of attributes. Further parameter settings which have to be derived in a probably lengthy procedure are not required, what can be seen as a major advantage of ABHC.

The specification for the OBSP is similar to the one suggested by Henn and Wäscher (2010) for the OBP. The initial solution is generated by means of the EDD rule. For the neighborhood structure all solutions which can be obtained by a swap move or a shift move are considered. With respect to the attributes the following two sets are used: The attribute set $\mathbf{A}_{o,o} := \{(i_1, i_2) | 1 \leq i_1 < i_2 \leq n\}$ (Order, Order) describes the solution by pairs (i_1, i_2) of customer orders i_1, i_2 which are assigned to the same batch. The resulting number of attributes amounts to $q_{o,o} = n(n-1)/2$. In the following this configuration of ABHC will be denoted by ABHC$_{o,o}$. The second set of attributes (Batch, Order) captures the assignment of customer orders to batches, i.e. $\mathbf{A}^{b,o} := \{(j, i) | 1 \leq j \leq \overline{m}, 1 \leq i \leq n\}$. This results in $q_{b,o} = \overline{m} \cdot n$ combinations. This configuration will be abbreviated with ABHC$_{b,o}$.

6 Test Design

6.1 Warehouse Parameters and Problem Classes

Numerical experiments have been carried out in order to determine the performance of the suggested heuristics. In these experiments, a single-block layout with two cross aisles, one in the front and one in the back of the picking area, is considered. All aisles are vertically orientated and the depot is located in front of the leftmost aisle. An exemplary visualization of this type of layout is depicted in Figure 2.1; this layout is frequently used in literature (de Koster et al., 1999; Henn et al., 2010). The picking area consists of 900 storage locations and 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 on both sides of each aisle). The aisles are numbered from 1 to 10, where aisle #1 is the leftmost aisle and aisle #10 is the rightmost aisle. Within each aisle two-sided picking is assumed, i.e. being positioned in the center of an aisle, the order picker can pick items from storage locations on the right as well as from storage locations on the left without additional movements. The length of each storage location amounts to one length unit (LU). When picking an item, the order picker is positioned in the middle of the storage location. 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. 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.

A class-based storage assignment of articles to storage locations is assumed: Articles are grouped into three classes (A, B, and C) depending on their demand, where A contains articles with high, B with medium and C with low demand frequencies. Articles of class A are stored in aisle #1, articles of class B in aisles #2, #3 and #4, and articles of class C in the remaining six aisles. Furthermore, 52 percent of the demand belongs to articles in class A, 36 percent to articles in class B and 12 percent to articles in class C. Within a class, the location of each article is determined randomly. The considered demand frequencies represent a typical environment for a real-world order picking system and have been used in various numerical experiments (de Koster et al., 1999).

As described in Section 2, the processing time is composed of the travel time, the search time, the pick time and the setup time. For the travel time we assume that an order picker walks 10 length units in 30 seconds. Moreover, he/she needs 10 seconds to search and pick an item from a storage location. Each tour requires a setup time of 3 minutes.

In our numerical experiments the number of orders (n) has been chosen from (20, 40, 60, 80), where the number of articles per order is uniformly distributed in $\{5, 6, \ldots, 25\}$. The capacity of the picking device (defining the maximal number of items that can be collected in a tour) (C) amounts to 45 or 75 items. The due date for each order is chosen from the interval

$$[\min_{j\in J} pt_j^s, 2\cdot(1-\mathrm{MTCR})\sum_{j\in J} pt_j^s + \min_{j\in J} pt_j^s],\ \text{with } 0\leq \mathrm{MTCR}\leq 1, \tag{6.1}$$

where pt_j^s is the processing time of order j ($j\in J$) if the order is processed in a batch without any other customer order (Elsayed and Lee, 1996). MTCR is called *modified traffic congestion rate*. This rate describes the tightness of the problem with respect to the due dates. A high MTCR value indicates that the due dates are very close to each other and therefore difficult or even impossible to meet. In instances with a low MTCR value it is possible that the majority of the customer orders can be completed before their due dates. For the MTCR the values 0.50, 0.55, 0.60, 0.65, 0.70, and 0.75 have been considered. In combination with the two routing strategies (S-Shape, Largest Gap) we obtain 96 problem classes; for each of these classes 50 instances have been generated.

6.2 Benchmarks, Algorithm Parameters and Implementation

Application of EDD represents a straightforward way to generate a solution to the OBSP. For each problem instance the total tardiness obtained by application of EDD serves as a reference value. The solution quality of all other strategies is measured in relation to that of EDD: the improvement of the total tardiness of a solution s compared to the total tardiness of a solution s^{EDD} provided by EDD is calculated as $100\cdot(tar(s^{\mathrm{EDD}})-tar(s))/tar(s^{\mathrm{EDD}})$.

In order to provide a classic local search method against which ILS and ABHC can be benchmarked, we combine EDD with the local search procedure described in Section 4 (EDD+LS).

The parameter settings of ILS have been determined in a series of pre-tests: The rearrangement factor λ has been set to 0.3. A maximal deterioration of $\mu = 0.05$ (threshold factor) is allowed after 40 calls of the perturbation phase in which no improvement of the best found solution has been obtained. ILS terminates after having executed 10 deteriorations.

The computations for all 4,800 instances have been carried out on a desktop PC with a 2.21 GHz Pentium processor with 2.0 GB RAM. The algorithms have been encoded in C++.

7 Test Results

7.1 Solution Quality

Table 7.1 and 7.2 depict the solution qualities of the evaluated heuristics if routes are determined by the S-Shape-Heuristic and Largest Gap-Heuristic, respectively. For each problem class the average total tardiness (tar) in minutes obtained by each solution approach is presented. Furthermore, the tables show the corresponding (average) improvements compared to the total tardiness of EDD (imp). The smallest average total tardiness observed for each problem class is highlighted bold. In the appendix the tables A.1 and A.2 give further insights into the solution qualities of the proposed heuristics by showing the average tardiness per customer order (tar_av), the number of delayed costumer orders (tar_no) and the time necessary to complete all customer orders, i.e. the

n	C	**MTCR**	**EDD**	**EDD+LS**		**ILS**		**ABHC$_{o,o}$**		**ABHC$_{b,o}$**	
			tar [min]	tar [min]	imp [%]	tar [min]	imp [%]	tar [min]	imp [%]	tar [min]	imp [%]
20	45	0.50	20.1	17.9	10	**15.1**	32	**15.1**	32	15.7	29
		0.55	22.7	20.1	8	**15.2**	26	15.9	25	17.1	23
		0.60	51.6	42.1	27	**33.3**	43	33.9	41	34.5	40
		0.65	56.9	46.6	20	**33.9**	41	34.4	37	34.9	36
		0.70	107.5	86.8	24	**60.6**	47	61.2	47	64.3	43
		0.75	291.0	245.7	15	170.1	42	**169.5**	42	178.2	39
	75	0.50	14.7	11.9	22	**7.4**	48	8.7	41	9.0	39
		0.55	20.4	17.1	23	**11.3**	49	12.0	45	12.8	42
		0.60	22.1	19.1	13	**12.2**	47	13.1	42	13.3	42
		0.65	38.5	34.5	11	**22.7**	50	27.4	36	27.7	34
		0.70	64.2	55.8	19	**43.9**	41	47.8	33	49.2	31
		0.75	106.7	90.3	16	**75.4**	32	77.7	28	81.1	25
40	45	0.50	25.1	21.2	19	**16.5**	32	**16.5**	33	17.0	31
		0.55	43.1	36.3	14	**24.9**	39	25.1	40	25.5	38
		0.60	45.8	37.7	22	27.5	40	**27.3**	44	28.0	37
		0.65	96.4	79.1	19	**51.3**	40	52.1	40	54.6	35
		0.70	327.1	268.6	23	**137.8**	59	138.8	59	150.8	55
		0.75	1043.6	894.8	14	**479.9**	55	480.6	56	505.7	53
	75	0.50	23.5	20.3	18	**12.7**	49	13.4	46	13.7	44
		0.55	26.4	22.4	20	**15.2**	49	15.8	49	16.2	47
		0.60	27.2	23.3	19	**16.0**	50	16.4	49	16.6	48
		0.65	53.7	46.2	22	**34.9**	49	36.0	45	37.5	43
		0.70	92.8	81.4	15	**61.0**	40	62.3	38	66.5	34
		0.75	245.8	203.9	17	**142.5**	41	144.3	38	153.8	34
60	45	0.50	16.7	14.6	13	**11.9**	28	12.0	28	12.3	26
		0.55	45.7	38.2	12	28.7	30	**28.5**	30	29.4	30
		0.60	68.8	57.4	21	**34.9**	42	**34.9**	44	38.4	39
		0.65	212.1	172.7	17	97.1	38	**95.5**	39	102.0	37
		0.70	633.8	495.6	26	199.9	71	**198.9**	72	211.9	69
		0.75	2230.2	1953.9	13	899.0	62	**893.1**	62	939.5	60
	75	0.50	20.1	17.1	21	**11.3**	51	**11.3**	51	11.4	51
		0.55	24.0	20.2	22	**13.5**	50	13.9	49	13.7	50
		0.60	43.3	36.1	22	27.7	52	**27.4**	52	28.1	50
		0.65	54.0	45.0	19	**33.8**	52	34.4	50	35.4	49
		0.70	97.2	81.4	13	**58.5**	41	**58.5**	40	61.1	36
		0.75	231.7	174.8	20	119.1	48	**118.2**	46	126.8	40
80	45	0.50	18.1	16.1	19	11.9	37	**11.7**	38	11.9	37
		0.55	32.0	26.7	25	20.3	43	**20.2**	43	20.4	40
		0.60	57.1	44.3	24	34.8	41	**34.0**	42	35.7	38
		0.65	269.3	225.9	18	**109.6**	47	111.1	47	118.1	44
		0.70	984.0	767.6	24	**254.9**	73	263.4	73	278.7	71
		0.75	3582.6	3163.9	12	1241.9	67	**1225.7**	68	1284.5	66
	75	0.50	12.3	10.0	19	4.9	50	**4.7**	52	**4.7**	51
		0.55	27.9	23.0	18	16.0	53	15.9	54	**15.8**	54
		0.60	30.1	25.7	19	**18.3**	49	18.5	47	18.6	48
		0.65	49.6	41.8	16	**31.1**	40	**31.1**	40	31.7	40
		0.70	115.8	96.1	18	73.1	41	**72.8**	41	76.6	38
		0.75	442.3	352.4	21	214.0	52	**207.7**	53	226.9	49
Average					18		46		45		42
Maximum					27		73		73		71
Minimum					8		26		25		23

Table 7.1: Solution quality for S-Shape routing

total processing time or makespan (mak). All entries in the tables 7.1, 7.2, A.1 and A.2 represent average values over the instances generated for each problem class.

Analyzing the results for S-Shape routing, the following can be observed: For EDD+LS the average improvement amounts to 18%; the improvements range from 8% (S-Shape; $n = 20$; $C = 45$; MTCR = 0.55) to 27% (S-Shape; $n = 20$; $C = 45$; MTCR = 0.6). The total tardiness obtained by ILS improves the tardiness of EDD by 46%, whereas a minimal improvement of 26% (20; 45; 0.55) and a maximal improvement of 73% (80; 45; 0.70) can be observed. Application of ABHC

n	*C*	**MTCR**	**EDD**	**EDD+LS**		**ILS**		**ABHC**$_{o,o}$		**ABHC**$_{b,o}$	
			tar [min]	tar [min]	imp [%]	tar [min]	imp [%]	tar [min]	imp [%]	tar [min]	imp [%]
20	45	0.50	17.4	15.0	18	**11.5**	40	11.7	39	11.9	38
		0.55	21.2	18.4	17	**14.2**	39	14.3	37	15.1	33
		0.60	30.3	25.5	17	20.7	34	**20.6**	33	21.6	32
		0.65	64.7	54.6	23	**41.2**	44	41.4	43	43.1	40
		0.70	163.8	138.9	18	**102.8**	40	103.1	40	108.2	36
		0.75	311.3	269.4	14	**194.9**	38	195.0	38	202.0	36
	75	0.50	19.2	16.7	19	**10.7**	49	12.8	42	13.6	37
		0.55	22.3	18.9	18	**11.9**	57	13.5	49	14.0	47
		0.60	43.4	38.0	14	**28.3**	46	30.7	38	31.0	36
		0.65	43.9	38.9	16	**27.7**	52	31.6	41	32.9	38
		0.70	69.1	60.5	13	**48.9**	41	52.7	32	53.6	29
		0.75	141.6	122.2	12	**105.1**	25	107.0	22	110.4	20
40	45	0.50	27.8	24.4	16	**18.0**	42	**18.0**	42	18.3	39
		0.55	35.2	30.2	14	24.0	34	**23.7**	34	24.6	32
		0.60	54.2	46.9	14	32.6	36	**32.4**	36	33.5	33
		0.65	105.5	86.9	19	55.6	45	**55.1**	45	59.4	42
		0.70	432.8	359.7	20	**192.6**	59	193.6	60	208.5	56
		0.75	1056.4	927.9	12	538.1	50	**536.9**	51	563.7	48
	75	0.50	19.6	16.3	18	**9.9**	47	10.5	45	11.0	40
		0.55	24.0	21.1	20	**14.1**	50	14.7	47	15.0	46
		0.60	32.7	28.2	23	**21.0**	52	21.7	49	23.1	44
		0.65	54.9	48.6	11	**37.5**	45	38.6	41	39.6	39
		0.70	102.9	91.2	11	**73.2**	36	74.9	34	77.9	28
		0.75	275.4	237.6	14	**176.2**	34	176.8	34	182.6	31
60	45	0.50	33.1	29.0	15	**20.1**	37	20.5	37	21.1	33
		0.55	33.7	30.1	15	24.6	36	**24.2**	37	25.0	35
		0.60	142.6	116.9	17	**75.8**	40	76.3	40	80.0	38
		0.65	207.1	175.0	20	88.7	49	**88.0**	50	93.8	47
		0.70	906.4	767.1	19	**341.8**	64	349.3	64	368.9	62
		0.75	2278.6	2029.9	12	1016.7	57	**1016.4**	57	1060.6	56
	75	0.50	21.7	18.4	18	12.1	50	**12.0**	50	12.0	50
		0.55	26.5	22.7	21	**15.6**	53	16.3	49	17.3	47
		0.60	29.1	25.8	11	**17.2**	46	17.4	44	18.4	41
		0.65	81.6	72.4	14	56.5	40	**56.4**	41	57.4	40
		0.70	120.9	103.0	13	78.6	37	**78.5**	37	82.5	33
		0.75	394.4	338.1	16	216.5	43	**213.7**	44	223.6	41
80	45	0.50	21.8	18.1	22	13.6	37	**13.4**	38	14.1	36
		0.55	68.8	59.2	16	**41.7**	34	42.0	34	43.3	32
		0.60	65.1	54.4	19	**36.9**	38	37.7	39	39.0	36
		0.65	176.9	142.7	26	80.9	54	**79.6**	54	83.7	50
		0.70	984.0	819.6	22	**281.3**	72	289.3	72	304.8	70
		0.75	4073.5	3682.3	10	1656.5	60	**1646.8**	61	1731.1	59
	75	0.50	25.0	22.1	12	16.0	36	**15.9**	36	**15.9**	37
		0.55	31.1	28.2	16	**20.4**	45	20.8	43	20.6	45
		0.60	37.2	31.2	17	**22.5**	44	**22.5**	44	22.6	44
		0.65	50.8	45.1	16	30.9	49	**30.6**	49	31.5	47
		0.70	124.7	109.2	14	82.5	44	**82.2**	44	84.4	41
		0.75	584.9	481.1	19	272.4	55	**266.0**	56	289.1	52
Average					16		45		44		41
Maximum					26		72		72		70
Minimum					10		25		22		20

Table 7.2: Solution quality for Largest Gap routing

improves the solutions generated by EDD on average by 45% (ABHC$_{o,o}$) and 42% (ABHC$_{b,o}$). For ABHC$_{o,o}$ improvements are contained in the interval [25%; 73%] where the limits originate from the problem classes (20; 45; 0.55) and (80; 45; 0.70). These two problem classes also determine the limits for the improvements obtained by ABHC$_{b,o}$ (23% and 71%, respectively). In general, ILS finds the smallest tardiness for 32 (out of 48) problem classes, ABHC$_{o,o}$ for 21 problem classes and ABHC$_{b,o}$ for two problem classes.

For Largest Gap routing, the results and improvements of the heuristics are similar. Application of EDD+LS can improve the total tardiness by 16 % on average (ranging from 10% (80; 45; 0.75) to 26% (80; 45; 0.65)). ILS improves the total tardiness by 45 % with a minimal improvement of 25% (20; 75; 0.75) and a maximal one of 72% (80; 45; 0.70). The application of ABHC$_{o,o}$ results in an average improvement of 44%, a minimal improvement of 22% (20; 75; 0.75) and a maximal improvement of 72% (80; 45; 0.70). The corresponding values for ABHC$_{b,o}$ are 41%, 20% (20; 75; 0.75), and 70% (80; 45; 0.70). Again, ILS turns out to be the best performing approach since it finds the smallest tardiness for 29 (out of 48) problem classes, whereas ABHC$_{o,o}$ does so for 21 problem classes and ABHC$_{b,o}$ for one problem class.

The results obtained by EDD+LS demonstrate that the application of a single local search procedure can already improve the total tardiness significantly. Moreover, escaping from local minima by means of deterioration steps – as implemented by ILS and ABHC – contributes to a further reduction of the total tardiness. The large reductions of the total tardiness resulting from the application of the metaheuristics – in comparison to EDD and to EDD+LS – can be explained as follows: ILS and ABHC are able to reduce the total processing time and the number of batches substantially. This advantage over constructive methods has already been observed in previous numerical studies (Henn et al., 2010; Henn and Wäscher, 2010) and can also be identified by analyzing the total processing time. Based on the smaller total processing time, total tardiness can also be reduced. The reduction of the total processing time does not only decrease the tardiness of a single customer order but may also have an impact on the tardiness for more than one customer order. With respect to the total processing times obtained by ILS, it has to be noted that these values exceed the total processing times for the remaining solution approaches significantly for some problem classes. These peculiarities are caused by batches which contain only a single customer order and which are released at the end of the planning period. For the customer orders in those batches due dates can be met without further combinations of batches, since sufficient time has been saved at the beginning of the planning period. In addition to the reduction of the tardiness by minimizing the total processing time, the tardiness can further be improved by other exchanges. ILS and ABHC are able to postpone a customer order with an early due date to a later point in time, accepting that the due date of this order is violated, if the later release can be used to complete other customer orders before their due dates.

The configuration of ABHC$_{o,o}$, which uses the attribute set $\mathbf{A}_{o,o}$, provides a smaller total tardiness than the configuration ABHC$_{b,o}$, which uses $\mathbf{A}_{b,o}$. This small difference can be explained by the different numbers of attributes and therefore the different sizes of the set of acceptable solutions during the search. Whereas ABHC$_{o,o}$ makes use of $n \cdot (n-1)/2$ attributes, ABHC$_{b,o}$ considers only $\overline{m} \cdot n$ attributes which is smaller than $n \cdot (n-1)/2$. This becomes also evident from an analysis of the different capacities: for a small capacity $\overline{m}$ is larger than in case of a higher capacity, since fewer batches are required. For a small capacity (i.e. a larger $\overline{m}$) the differences between the two configurations are smaller than in case of a larger capacity (i.e. a smaller $\overline{m}$). However, the small differences between both variants show that the general principle of ABHC provides – independent from the attribute set – an appropriate framework to obtain high-quality solutions for the OBSP. Comparing the results of ILS on the one hand and both ABHC variants on the other, the application of ILS results in the smallest tardiness on average. Nevertheless, some problem classes exist (characterized by a large number of orders and a large MTCR) in which ILS is outperformed by ABHC$_{o,o}$.

n	C	MTCR	S-Shape routing					Largest Gap routing				
			EDD	EDD+LS	ILS	ABHC$_{o,o}$	ABHC$_{b,o}$	EDD	EDD+LS	ILS	ABHC$_{o,o}$	ABHC$_{b,o}$
20	45	0.50	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1	< 1	< 1
		0.55	< 1	< 1	< 1	< 1	< 1	< 1	< 1	2	< 1	< 1
		0.60	< 1	< 1	1	< 1	< 1	< 1	< 1	2	< 1	< 1
		0.65	< 1	< 1	1	< 1	< 1	< 1	< 1	2	< 1	< 1
		0.70	< 1	< 1	2	< 1	< 1	< 1	< 1	3	< 1	< 1
		0.75	< 1	< 1	1	< 1	< 1	< 1	< 1	3	< 1	< 1
	75	0.50	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1	< 1	< 1
		0.55	< 1	< 1	< 1	< 1	< 1	< 1	< 1	2	< 1	< 1
		0.60	< 1	< 1	< 1	< 1	< 1	< 1	< 1	2	< 1	< 1
		0.65	< 1	< 1	1	< 1	< 1	< 1	< 1	3	< 1	< 1
		0.70	< 1	< 1	1	< 1	< 1	< 1	< 1	3	< 1	< 1
		0.75	< 1	< 1	1	< 1	< 1	< 1	< 1	3	< 1	< 1
40	45	0.50	< 1	< 1	6	9	11	< 1	< 1	15	14	19
		0.55	< 1	< 1	8	9	10	< 1	< 1	16	19	20
		0.60	< 1	< 1	10	10	10	< 1	< 1	22	17	21
		0.65	< 1	< 1	14	12	10	< 1	< 1	31	23	19
		0.70	< 1	< 1	22	16	9	< 1	< 1	41	26	17
		0.75	< 1	< 1	20	11	5	< 1	< 1	35	16	9
	75	0.50	< 1	< 1	4	6	7	< 1	< 1	9	11	13
		0.55	< 1	< 1	5	5	7	< 1	< 1	11	12	16
		0.60	< 1	< 1	5	7	8	< 1	< 1	14	13	16
		0.65	< 1	< 1	9	7	8	< 1	< 1	21	18	16
		0.70	< 1	< 1	17	12	9	< 1	< 1	29	23	16
		0.75	< 1	< 1	24	13	7	< 1	< 1	44	29	12
60	45	0.50	< 1	< 1	21	36	49	< 1	< 1	43	69	87
		0.55	< 1	< 1	29	48	55	< 1	< 1	54	76	94
		0.60	< 1	< 1	44	57	62	< 1	< 1	102	103	108
		0.65	< 1	< 1	77	76	37	< 1	< 1	402	490	450
		0.70	< 1	< 1	98	105	68	< 1	< 1	196	172	114
		0.75	< 1	< 1	81	64	33	< 1	< 1	143	102	75
	75	0.50	< 1	< 1	11	23	37	< 1	< 1	32	53	77
		0.55	< 1	< 1	17	30	44	< 1	< 1	44	53	85
		0.60	< 1	< 1	22	36	49	< 1	< 1	67	72	94
		0.65	< 1	< 1	31	37	58	< 1	< 1	93	87	104
		0.70	< 1	< 1	52	57	52	< 1	< 1	151	127	98
		0.75	< 1	< 1	104	74	49	< 1	< 1	243	184	84
80	45	0.50	< 1	< 1	54	135	219	< 1	< 1	104	285	349
		0.55	< 1	< 1	70	139	182	< 1	< 1	169	329	394
		0.60	< 1	< 1	125	185	209	< 1	< 1	237	326	383
		0.65	< 1	< 1	214	273	250	< 1	< 1	437	435	485
		0.70	< 1	< 1	303	362	284	< 1	< 1	504	608	457
		0.75	< 1	< 1	240	233	141	< 1	< 1	417	362	253
	75	0.50	< 1	< 1	27	91	126	< 1	< 1	76	204	290
		0.55	< 1	< 1	40	95	157	< 1	< 1	112	205	338
		0.60	< 1	< 1	47	100	157	< 1	< 1	135	229	303
		0.65	< 1	< 1	78	131	159	< 1	< 1	212	259	378
		0.70	< 1	< 1	143	175	186	< 1	< 1	418	376	320
		0.75	< 1	< 1	349	346	183	< 1	< 1	490	279	160

Table 7.3: Computing times in seconds

7.2 Computing Times

The computing times required by the proposed heuristics are presented in Table 7.3. Computing times of EDD and EDD+LS are smaller than one second and can, therefore, be considered as negligible. The computing times increase for all algorithms with an increasing number of customer orders. For a small MTCR the computing times are small compared to a larger MTCR. This can be attributed to the fact that the problem classes are not very challenging from a computational point of view. As an exception we have to consider the problem classes with a MTCR of 0.75. For some problem classes the computing times required by the metaheuristics are smaller than the ones for the corresponding problem classes with a MTCR of 0.70. These particularities reveal that the problem structure has changed. For problem classes with a MTCR of 0.70 it has to be decided for the first time *which* customer order should be delayed. For problem classes with a MTCR of 0.75 the amount of the tardiness has to be considered only, since for a large number of customer orders the due date cannot be met at all. Moreover, the computing times required for those problem instances in which tours are generated by the Largest Gap-Heuristic are larger than in case of S-Shape routing, since the calculations for the tour length require a larger amount of time by the application of the Largest Gap-Heuristic than in case of S-Shape routing.

Computing times of $\text{ABHC}_{o,o}$ are larger than those of $\text{ABHC}_{b,o}$, which can be explained by an increased number of attributes incorporated in $\text{ABHC}_{o,o}$. The increase in computing times of both ABHC variants – in comparison to the ones of ILS – can be attributed to the fact that ABHC evaluates a large number of solutions although the best solution has been determined. After having identified the best solution, ILS terminates after a fixed number of iterations. Furthermore, the larger computing times of ABHC are caused by the computational effort, which is necessary for deciding whether a solution represents the best found solution for an attribute and for updating $\text{best}(i, k)$ in each iteration.

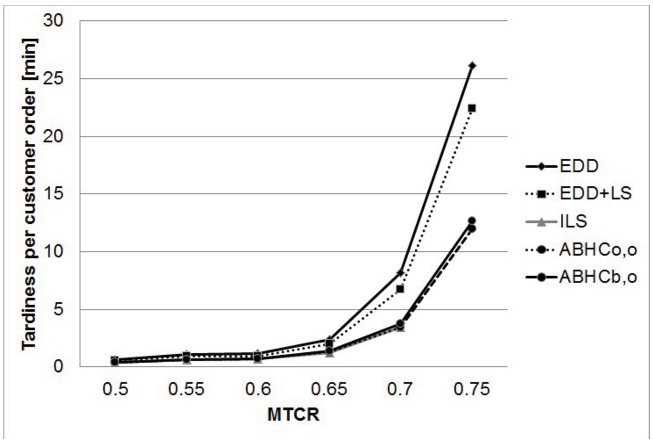
7.3 Effects of Instance Characteristics Variation

In this subsection it will be analyzed how the parameters n, C, MTCR, and the routing strategy influence the total tardiness. As expected, the total tardiness increases with an increasing number of customer orders. Considering the average tardiness per customer order (cf. tables A.1 and A.2) the following can be observed: Only for those problem classes which are characterized by a large MTCR (0.75), the average tardiness increases with n. For small values of MTCR – and with respect to the application of the metaheuristics in the first place – a reduction of the average tardiness can be observed when n is increased. This can be explained as follows: If the number of customer orders is large, similar customer orders exist which can be combined favorably in order to obtain small processing times and, thus, a smaller tardiness.

A comparison of the tardiness for problem classes with $C = 45$ to the tardiness obtained for the corresponding problem classes with $C = 75$ reveals, that the total tardiness is significantly smaller for larger capacities. When a picking device with a larger capacity is used, more customer orders can be collected on one tour than in case of a smaller capacity. Therefore, a reduction of the total processing time (as also indicated in tables A.1 and A.2) can be achieved, and customer orders can be completed earlier than in case of a small capacity. Consequently, this results in a smaller total tardiness.

Furthermore, for small value of MTCR (e.g. 0.50) a small – nearly negligible – tardiness can be noted. The total tardiness for S-Shape routing, $n = 20$, $C = 45$ and MTCR $= 0.50$ amounts for EDD to 20.1 minutes. For this problem class, application of ILS leads to a total tardiness of 15.1 minutes, which represents an absolute improvement of 5 minutes, whereas the relative improvement amounts to 32%. Therefore for problem classes with a small MTCR conclusions and interpretations based on relative improvements have to be drawn carefully. In general, it can be concluded that a larger MTCR results in a larger total tardiness. This can also be seen from the fact that for a small MTCR nearly all customer orders can be completed before their due dates, whereas for a large MTCR it is not possible to meet 50% of the due dates. Figure 7.1 presents the development of the tardiness per customer order for different values of MTCR, 40 customer orders and a capacity of 45 items. As indicated by the figure, the total tardiness does not increase linearly with MTCR, but rather exponentially.

For the majority of the problem classes (mainly those classes with $C = 75$) the results obtained by S-Shape routing are slightly smaller than in the corresponding problem classes for which the picking tours are determined by means of the Largest Gap routing strategy. This small advantage is caused by the fact that for problem classes with a large capacity S-Shape routing is able to provide smaller tour lengths than Largest Gap routing (as also observed by Henn et al. (2010)) which results in smaller processing times and in a smaller total tardiness.



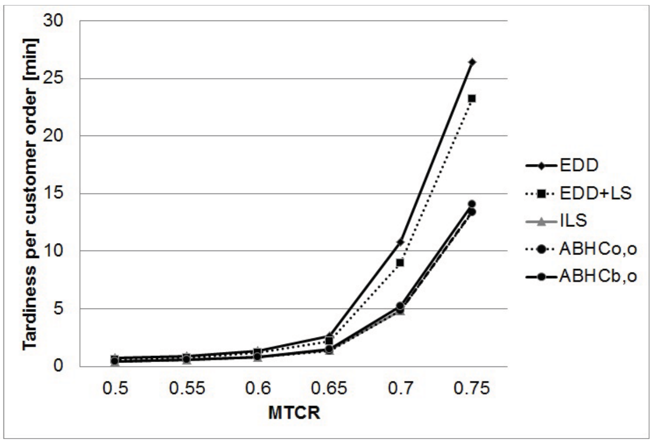


Figure 7.1: Average tardiness per customer order for $n = 40$, $C = 45$, S-Shape routing (left) and Largest Gap routing (right)

8 Conclusion

This paper dealt with the joint OBSP, where batches have to be constructed and released in order to minimize the total tardiness for a given set of customer orders. The problem is pivotal for efficient management and control of manual picker-to-parts order picking systems. For the solution of this problem we proposed the application of two local search-based metaheuristics, namely Iterated Local Search and the Attribute-Based Hill Climber heuristic.

By means of numerical experiments it was shown that both approaches generate solutions superior to those from a priority-rule based heuristic. Implementing these solutions can improve customer service by delivering the customers on time and by avoiding delay in other production stages. In total, the algorithms can improve the warehouse performance significantly. Since both algorithms required small computation times, both variants appear very suitable for being implemented in software systems for practical purposes.

Further research directions may deal with an extension of the OBSP, in which order pickers can work in parallel. Moreover, priority values can be assigned to the due dates of the customer orders, resulting in a weighted tardiness. Furthermore, we conclude that the interaction of order batching and batch sequencing with other related planning issues (layout design, item location, picker routing) has not been considered sufficiently in the literature. Thus, it might be worthwhile to provide a "global" solution approach which integrates decisions on these planning issues in order to minimize total tardiness. In dynamic situations, customer orders arrive continuously and batching decisions have to be made under a rolling planning horizon. Therefore, it would be meaningful to consider a dynamic variant of the OBSP.

References

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

de Koster, R.; Le-Duc, T. and Roodbergen, K.J. (2007). Design and Control of Warehouse Order Picking: A Literature Review. *European Journal of Operational Research 182*(2), 481–501.

Derigs, U. and Kaiser, R. (2007). Applying the Attribute Based Hill Climber Heuristic to the Vehicle Routing Problem. *European Journal of Operational Research 177*(2), 719–732.

Derigs, U. and Reuter, K. (2009). A Simple and Efficient Tabu Search Heuristic for Solving the Open Vehicle Routing Problem. *Journal of the Operational Research Society 60*(12), 1658–1669.

Du, J. and Leung, J.Y.-T. (1990). Minimizing Total Tardiness on one Machine is NP-hard. *Mathematics of Operations Research 15*(3), 483–495.

Elsayed, E.A. and Lee, M.-K. (1996). Order Processing in Automated Storage/Retrieval Systems with Due Dates. *IIE Transactions 28*(7), 567–577.

Elsayed, E.A.; Lee, M.-K.; Kim, S. and Scherer, E. (1993). Sequencing and Batching Procedures for Minimizing Earliness and Tardiness Penalty of Order Retrievals. *International Journal on Productions Research 31*(3), 727–738.

Gademann, A.J.R.M.; van den Berg, J.P. and van der Hoff, H.H. (2001). An Order Batching Algorithm for Wave Picking in a Parallel-Aisle Warehouse. *IIE Transactions 33*(5), 385–398.

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

Henn, S.; Koch, S. and Wäscher, G. (2012). Order Batching in Order Picking Warehouses: A Survey of Solution Approaches. In Manzini, R. (Ed.), *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems.* Heidelberg: Springer. to appear.

Henn, S. and Wäscher, G. (2010). Tabu Search Heuristics for the Order Batching Problem in Manual Order Picking Systems. Working Paper 07/2010, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg.

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

Lorenço, H.R.; Martin, O.C. and Stützle, T. (2010). Iterated Local Search:Framework and Applications In Gendreau, M. and Potvin, J.-Y. (Eds.), *Handbook of Metaheuristics*, 2nd ed., pp. 363–397. Norwell: Kluwer Academic Publishers.

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

Tompkins, J.A.; White, J.A.; Bozer, Y.A.; Frazelle, E.H. and Tanchoco, J.M.A. (2003). *Facilities planning*, 2nd ed., New York: John Wiley & Sons.

Tsai, C.-Y.; Liou, J.J.H. and 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. and Reese, J. (Eds.), *Supply Chain Management and Reverse Logistics*, pp. 323–347. Berlin et al.: Springer.

Whittley, I.M. and Smith, G.D. (2004). The Attribute Based Hill Climber. *Journal of Mathematical Modelling and Algorithms 3*(2), 167–178.

Yu, M. and de Koster, R. (2009). The Impact of Order Batching and Picking Area Zoning on Order Picking System Performance. *European Journal of Operational Research 198*(2), 480–490.

A Appendix

n	C	**MTCR**	**EDD**			**EDD+LS**			**ILS**			**ABHC$_{o,o}$**			**ABHC$_{b,o}$**		
			tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]
20	45	0.50	1.0	2.5	131	0.9	2.3	130	0.8	2.0	136	0.8	2.0	130	0.8	2.0	130
		0.55	1.1	2.7	134	1.0	2.6	134	0.8	2.1	138	0.8	2.1	134	0.9	2.2	133
		0.60	2.6	4.7	133	2.1	4.2	132	1.7	3.5	137	1.7	3.5	132	1.7	3.7	132
		0.65	2.8	6.4	131	2.3	5.5	130	1.7	4.4	136	1.7	4.2	129	1.7	4.3	129
		0.70	5.4	9.7	130	4.3	8.0	128	3.0	6.2	128	3.1	6.2	125	3.2	6.3	125
		0.75	14.5	16.5	133	12.3	15.0	131	8.5	11.1	124	8.5	11.1	124	8.9	11.7	125
	75	0.50	0.7	1.7	103	0.6	1.5	102	0.4	1.3	111	0.4	1.4	102	0.4	1.3	102
		0.55	1.0	2.2	103	0.9	1.9	103	0.6	1.5	111	0.6	1.6	102	0.6	1.6	102
		0.60	1.1	2.6	103	1.0	2.4	102	0.6	1.8	112	0.7	2.0	102	0.7	2.0	102
		0.65	1.9	4.0	99	1.7	3.7	99	1.1	3.0	109	1.4	3.6	98	1.4	3.5	98
		0.70	3.2	6.3	99	2.8	5.5	98	2.2	4.7	108	2.4	5.0	98	2.5	5.0	98
		0.75	5.3	10.0	104	4.5	8.8	102	3.8	8.0	108	3.9	7.5	101	4.1	7.9	102
40	45	0.50	0.6	2.9	263	0.5	2.3	263	0.4	2.0	267	0.4	2.0	261	0.4	2.0	262
		0.55	1.1	4.3	262	0.9	3.8	262	0.6	2.9	268	0.6	2.9	260	0.6	3.0	259
		0.60	1.1	5.1	265	0.9	4.5	264	0.7	3.3	272	0.7	3.2	260	0.7	3.3	260
		0.65	2.4	8.1	262	2.0	7.2	261	1.3	5.0	268	1.3	5.0	257	1.4	5.5	256
		0.70	8.2	22.1	263	6.7	19.0	259	3.4	10.7	254	3.5	10.2	248	3.8	11.9	249
		0.75	26.1	34.0	263	22.4	30.8	258	12.0	19.9	239	12.0	20.4	237	12.6	21.5	237
	75	0.50	0.6	2.4	204	0.5	2.2	203	0.3	1.7	215	0.3	1.7	202	0.3	1.9	201
		0.55	0.7	3.0	201	0.6	2.7	200	0.4	2.2	214	0.4	2.1	199	0.4	2.2	198
		0.60	0.7	2.6	204	0.6	2.5	203	0.4	2.0	216	0.4	2.0	202	0.4	2.0	201
		0.65	1.3	5.2	201	1.2	4.5	200	0.9	3.8	216	0.9	4.1	199	0.9	3.8	198
		0.70	2.3	8.7	203	2.0	7.6	202	1.5	6.7	223	1.6	6.6	200	1.7	6.5	200
		0.75	6.1	19.3	204	5.1	16.6	202	3.6	12.4	211	3.6	12.4	198	3.8	12.7	198
60	45	0.50	0.3	2.0	394	0.2	1.9	394	0.2	1.6	398	0.2	1.6	393	0.2	1.6	390
		0.55	0.8	4.2	392	0.6	3.7	392	0.5	3.0	401	0.5	3.1	389	0.5	3.1	384
		0.60	1.1	5.9	394	1.0	5.0	393	0.6	3.8	401	0.6	3.7	390	0.6	3.9	387
		0.65	3.5	13.9	393	2.9	12.1	391	1.6	7.0	402	1.6	6.8	382	1.7	7.6	381
		0.70	10.6	34.7	394	8.3	29.2	389	3.3	13.3	381	3.3	13.5	371	3.5	13.9	372
		0.75	37.2	53.2	393	32.6	49.2	388	15.0	29.2	354	14.9	29.9	350	15.7	33.5	352
	75	0.50	0.3	2.2	302	0.3	2.0	302	0.2	1.5	310	0.2	1.5	301	0.2	1.5	300
		0.55	0.4	2.6	303	0.3	2.4	302	0.2	1.8	312	0.2	1.9	301	0.2	1.9	300
		0.60	0.7	4.4	304	0.6	3.9	304	0.5	3.1	317	0.5	3.3	302	0.5	3.2	300
		0.65	0.9	5.3	303	0.8	4.6	302	0.6	3.8	315	0.6	3.9	300	0.6	3.9	298
		0.70	1.6	8.5	306	1.4	7.3	304	1.0	6.0	323	1.0	5.9	301	1.0	6.1	300
		0.75	3.9	18.2	301	2.9	14.5	298	2.0	10.3	314	2.0	9.9	292	2.1	10.4	292
80	45	0.50	0.2	2.6	514	0.2	2.3	514	0.1	2.0	519	0.1	1.9	512	0.1	2.0	506
		0.55	0.4	3.7	515	0.3	3.0	515	0.3	2.6	518	0.3	2.7	510	0.3	2.6	506
		0.60	0.7	5.6	516	0.6	4.8	514	0.4	3.7	526	0.4	3.7	507	0.4	3.7	501
		0.65	3.4	14.4	517	2.8	13.1	515	1.4	7.7	535	1.4	7.8	504	1.5	8.0	499
		0.70	12.3	45.4	517	9.6	38.7	512	3.2	15.0	504	3.3	14.9	484	3.5	15.7	486
		0.75	44.8	71.0	518	39.5	66.8	512	15.5	34.9	464	15.3	36.7	456	16.1	41.2	458
	75	0.50	0.2	1.5	403	0.1	1.4	402	0.1	0.9	414	0.1	1.0	401	0.1	0.9	399
		0.55	0.3	3.0	402	0.3	2.6	402	0.2	2.0	412	0.2	2.0	401	0.2	1.9	397
		0.60	0.4	3.3	401	0.3	3.0	401	0.2	2.4	409	0.2	2.4	399	0.2	2.3	396
		0.65	0.6	4.9	399	0.5	4.4	399	0.4	3.4	413	0.4	3.6	398	0.4	3.7	394
		0.70	1.4	8.9	402	1.2	7.4	400	0.9	6.2	420	0.9	6.3	396	1.0	6.3	393
		0.75	5.5	27.1	401	4.4	22.2	398	2.7	14.1	422	2.6	13.9	388	2.8	14.6	387

Table A.1: Average tardiness, number of delayed customer orders, and makespan for S-Shape routing

n	C	**MTCR**	**EDD**			**EDD+LS**			**ILS**			**ABHC$_{o,o}$**			**ABHC$_{b,o}$**		
			tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]	tar_av [min]	no_tar	mak [min]
20	45	0.50	0.9	2.3	132	0.8	2.1	132	0.6	1.8	137	0.6	1.9	131	0.6	1.8	131
		0.55	1.1	2.7	127	0.9	2.4	127	0.7	2.2	132	0.7	2.3	126	0.8	2.2	126
		0.60	1.5	3.6	129	1.3	3.4	129	1.0	2.7	135	1.0	3.0	128	1.1	2.9	128
		0.65	3.2	6.0	129	2.7	5.3	128	2.1	4.2	130	2.1	4.3	126	2.2	4.4	127
		0.70	8.2	13.1	126	6.9	11.7	124	5.1	9.1	123	5.2	9.0	121	5.4	9.6	121
		0.75	15.6	16.8	131	13.5	15.4	129	9.7	11.6	124	9.8	11.6	123	10.1	12.2	123
	75	0.50	1.0	2.4	103	0.8	2.1	103	0.5	1.8	111	0.6	2.0	103	0.7	1.9	103
		0.55	1.1	2.3	103	0.9	2.2	103	0.6	1.8	114	0.7	2.1	103	0.7	2.0	103
		0.60	2.2	4.5	103	1.9	3.9	103	1.4	3.3	112	1.5	3.5	103	1.5	3.4	103
		0.65	2.2	4.7	102	1.9	4.3	102	1.4	3.7	113	1.6	4.0	102	1.6	4.0	102
		0.70	3.5	6.9	100	3.0	6.4	99	2.4	5.4	110	2.6	5.8	99	2.7	5.9	99
		0.75	7.1	12.9	104	6.1	11.5	103	5.3	10.6	106	5.4	10.5	102	5.5	10.5	103
40	45	0.50	0.7	3.3	254	0.6	3.0	253	0.5	2.4	259	0.5	2.4	252	0.5	2.5	252
		0.55	0.9	4.2	253	0.8	3.7	253	0.6	3.1	260	0.6	3.1	252	0.6	3.1	251
		0.60	1.4	5.3	257	1.2	4.9	256	0.8	4.0	266	0.8	3.7	254	0.8	3.8	254
		0.65	2.6	9.2	256	2.2	8.1	254	1.4	5.6	262	1.4	5.5	252	1.5	5.8	251
		0.70	10.8	24.7	257	9.0	22.4	254	4.8	12.9	246	4.8	12.7	243	5.2	13.8	243
		0.75	26.4	34.0	257	23.2	31.3	254	13.5	21.1	238	13.4	21.7	235	14.1	22.9	237
	75	0.50	0.5	2.1	200	0.4	2.0	200	0.2	1.5	210	0.3	1.7	199	0.3	1.6	199
		0.55	0.6	2.8	201	0.5	2.6	201	0.4	2.4	212	0.4	2.3	201	0.4	2.4	200
		0.60	0.8	3.4	201	0.7	3.1	200	0.5	2.6	214	0.5	2.7	200	0.6	2.6	200
		0.65	1.4	5.4	201	1.2	4.8	201	0.9	4.1	216	1.0	4.2	200	1.0	4.1	200
		0.70	2.6	9.5	200	2.3	8.5	200	1.8	7.0	216	1.9	7.1	198	1.9	7.5	198
		0.75	6.9	21.9	201	5.9	19.0	199	4.4	14.8	204	4.4	14.6	196	4.6	15.3	197
60	45	0.50	0.6	3.1	379	0.5	2.8	379	0.3	2.3	385	0.3	2.4	378	0.4	2.3	375
		0.55	0.6	3.5	379	0.5	3.3	378	0.4	2.6	385	0.4	2.5	377	0.4	2.7	374
		0.60	2.4	8.6	385	1.9	7.4	384	1.3	5.2	394	1.3	5.4	380	1.3	5.5	377
		0.65	3.5	14.5	508	2.9	12.6	506	1.5	7.4	521	1.5	6.8	496	1.6	7.2	493
		0.70	15.1	39.2	382	12.8	35.1	377	5.7	17.8	365	5.8	17.5	359	6.1	19.5	361
		0.75	38.0	54.4	382	33.8	50.2	377	16.9	30.8	347	16.9	32.0	344	17.7	34.3	345
	75	0.50	0.4	2.3	301	0.3	2.1	301	0.2	1.6	311	0.2	1.7	300	0.2	1.7	298
		0.55	0.4	2.7	299	0.4	2.5	299	0.3	2.1	311	0.3	2.2	298	0.3	2.2	297
		0.60	0.5	3.7	302	0.4	3.6	302	0.3	3.0	320	0.3	3.0	302	0.3	3.1	301
		0.65	1.4	7.0	301	1.2	6.4	301	0.9	5.3	315	0.9	5.2	300	1.0	5.2	298
		0.70	2.0	11.0	301	1.7	9.6	300	1.3	7.2	323	1.3	7.4	298	1.4	7.3	299
		0.75	6.6	27.2	302	5.6	24.0	300	3.6	15.6	306	3.6	15.0	295	3.7	16.1	295
80	45	0.50	0.3	2.5	508	0.2	2.2	508	0.2	1.8	515	0.2	1.8	505	0.2	1.8	501
		0.55	0.9	5.8	499	0.7	5.3	498	0.5	3.9	512	0.5	3.9	493	0.5	3.9	488
		0.60	0.8	6.2	508	0.7	5.3	507	0.5	4.0	518	0.5	4.0	502	0.5	4.0	499
		0.65	2.2	13.5	514	1.8	11.7	512	1.0	6.5	527	1.0	6.4	503	1.0	7.1	501
		0.70	12.3	44.6	508	10.2	40.0	503	3.5	16.3	485	3.6	16.4	479	3.8	17.4	480
		0.75	50.9	73.2	512	46.0	69.2	506	20.7	39.2	465	20.6	41.7	457	21.6	46.8	459
	75	0.50	0.3	2.6	400	0.3	2.3	400	0.2	2.1	412	0.2	2.1	399	0.2	2.0	397
		0.55	0.4	3.5	403	0.4	3.2	403	0.3	2.7	418	0.3	2.8	402	0.3	2.8	399
		0.60	0.5	3.9	393	0.4	3.5	392	0.3	2.9	407	0.3	3.0	391	0.3	3.0	388
		0.65	0.6	5.2	400	0.6	4.9	400	0.4	4.2	417	0.4	4.2	398	0.4	4.2	396
		0.70	1.6	9.2	400	1.4	8.4	399	1.0	6.3	424	1.0	6.2	396	1.1	6.3	393
		0.75	7.3	38.8	401	6.0	33.4	398	3.4	19.8	408	3.3	18.2	390	3.6	20.2	389

Table A.2: Average tardiness, number of delayed customer orders, and makespan for Largest Gap routing

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