WORKING PAPER SERIES

Variable Neighborhood Search for the Order Batching and Sequencing Problem with Multiple Pickers

Sebastian Henn

Working Paper No. 4/2012

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

FACULTY OF ECONOMICS
AND MANAGEMENT

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

Variable Neighborhood Search for the Order Batching and Sequencing Problem with Multiple Pickers

Sebastian Henn

February 2012

Abstract

Order picking deals with the retrieval of articles from their storage locations in order to satisfy customer requests. The transformation and consolidation of customer orders into picking orders (batches) is pivotal for the performance of order picking systems. Typically, customer orders have to be completed by certain due dates in order to avoid delays in production or in the shipment to customers. The composition of the batches, their processing times, their assignment to order pickers and the sequence according to which they are scheduled determine whether and the extent to which the due dates are missed. This article shows how Variable Neighborhood Descent and Variable Neighborhood Search can be applied in order to minimize the total tardiness of a given set of customer orders. In a series of extensive numerical experiments, the performance of the two approaches is analyzed for different problem classes. It is shown that the proposed methods provide solutions which may allow order picking systems to operate more efficiently.

Keywords: Warehouse Management; Order Batching; Batch Sequencing; Due Dates; Variable Neighborhood Descent; Variable Neighborhood Search

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 the warehouse function dealing with the retrieval of articles from a picking area in order to satisfy a given customer demand. Order picking is necessary because incoming articles are received and stored in (large-volume) unit loads while (internal or external) customers order small volumes (less-than-unit loads) of different articles. Among manual order picking systems, picker-to-parts systems can be considered as the most important. In these systems, order pickers walk (or drive) through the warehouse and collect the requested articles from the different storage locations (Wäscher, 2004). Order batching, i.e. the grouping of customer orders into picking orders (batches), has a significant impact on the efficient organization of the corresponding picking operations (de Koster et al., 2007).

In the majority of research it is focused on the minimization of pick times for a set of customer orders. However, nowadays on-time retrievals of customer orders have become important. For instance, this requirement can be identified in the following two situations: In distribution systems, mail order companies offer their customers to deliver requested items in a small time interval (e.g., 24 hours). Those items are typically carried by trucks to the customers. For those trucks fixed departure times from the distribution warehouse are determined, in order to guarantee that the items are delivered in the required time period (Gademann et al., 2001). Dependent on the particular distances to the customer different departure times arise. Due dates for the customer orders originate from the corresponding departure times.

Also in material warehouses on-time retrievals of components or material from the warehouse to the production stages are necessary since production delays should be avoided. Therefore due dates are determined for each order (Elsayed et al., 1993).

Whether and how these due dates can be met depends on the composition of the batches, their processing times (mainly determined by the length of the order pickers' tours), and by the sequence according to which the batches are processed by the different order pickers. Since it is sometimes not possible to meet all due dates, an assignment has to be determined which minimizes the total tardiness of the customer orders (Henn and Schmid, 2011). Previous research on this topic has neglected the assignment of customer orders to different order pickers. In practice, several order pickers are used in order to satisfy the customer orders. Therefore, the performance of the order picking system can be improved if an efficient assignment of batches to pickers is taken into account.

A local search-based approach is suggested for the solution of the corresponding optimization problem. This kind of approach has provided high-quality solutions for related problems (Henn et al., 2010; Henn and Schmid, 2011). The performance of local search-based methods depends significantly on the structure of the neighborhoods used. Modern heuristics change the structure of neighborhoods considered during the search. This paper shows how different neighborhood structures can be applied to minimize the tardiness for a set of customer orders. In detail, two approaches, a Variable Neighborhood Descent method and a Variable Neighborhood Search, are presented.

The remainder of this paper is organized as follows: In the next section, the Order Batching and Sequencing Problem with Multiple Pickers (OBSPMP) is defined and a mathematical model is presented. Section 3 contains an overview of the relevant literature. Section 4 summarizes the foundations of local search methods with multiple neighborhood structures. The specific applications of these methods to the proposed problem are presented in Section 5. A series of numerical experiments has been carried out in order to evaluate the suggested algorithms. Section 6 describes 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 in Section 7. The article concludes with a summary and an outlook on possible further research.

2 Order Batching and Sequencing Problem with Multiple Pickers

2.1 Problem Description

Customer orders are composed of order lines, where each order line consists of a particular article and the corresponding quantity requested. Order lines which should be processed together are summarized in a pick list. This list may include 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 picker 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. Tours provided by the *S-shape* or the *largest gap heuristic* are prevalent in practice, since order pickers seem to accept only straightforward and non-confusing routing schemes. Furthermore, in comparison to an optimal routing scheme, the tours provided by S-shape and largest gap routing seem to reduce the number of in-aisle congestions (de Koster et al., 1999, 2007). These routing strategies are visualized in Figure 2.1 for a single-block warehouse layout. The black rectangles symbolize the locations where items (article units) have to be picked (known as pick locations). 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 in the last aisle and would return to the depot along the front aisle. Applying the largest gap heuristic results in a tour in which the order picker always entirely traverses the rightmost and leftmost aisles which contain an item to be picked. All other

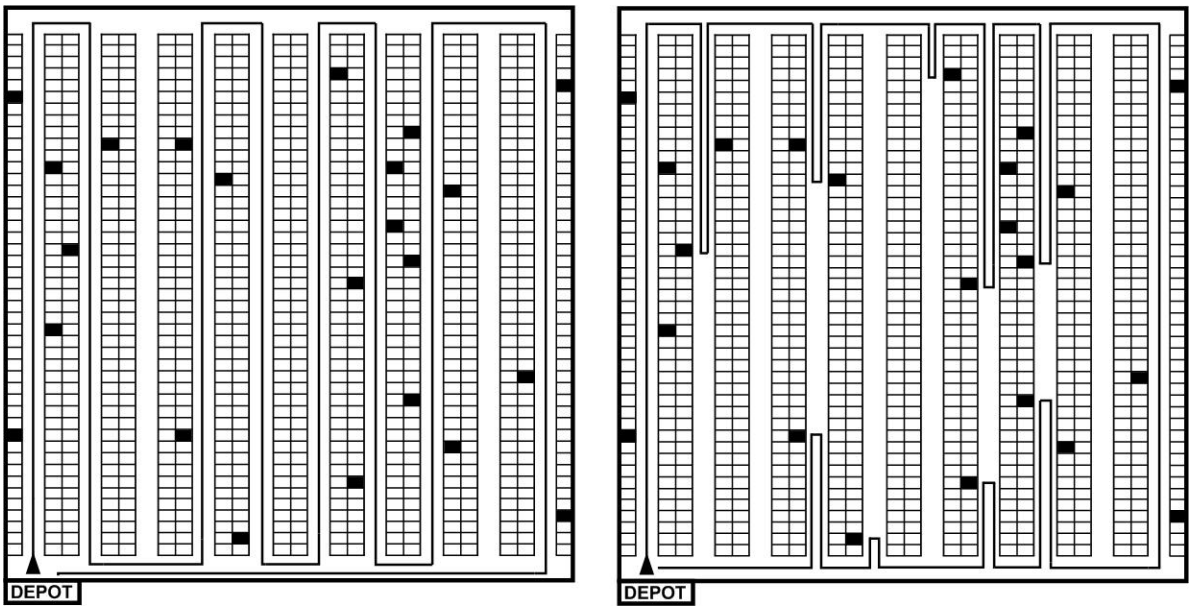

Figure 2.1: Example of S-shape routing (left) and largest gap routing (right) in a single-block layout

aisles are entered from the front or back aisle in such a way that the non-traversed distance is maximal.

Order pickers use picking devices (e.g. carts or roll pallets) for the collection of the requested items. The capacity of the picking device cannot be exceeded when combining customer orders into batches. The items requested in each customer order have to be collected on a single tour (order integrity condition), since splitting customer orders would result in an additional, unacceptable 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 here, all customer orders are known in advance. In the dynamic case customer orders may arrive at different points in time. The arrival time and the composition of the orders is not known in advance.

Given a particular picking order, the *(batch) processing time* (i.e. the time which is required to complete a batch) consists of the following components:

- *travel time*, i.e. the total length of the time periods 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 length of time required for the identification of items;
- *pick time*, i.e. the length of time needed for moving items from the corresponding locations onto the picking device;
- *setup time*, i.e. the length of time period consumed by administrative and setup tasks for each tour.

The point in time when an order picker starts his/her tour from the depot in order to collect all items is called the (batch) start date. The start date of a customer order is identical to the start date of the batch the order is assigned to. Analogously, the point in time when the order picker returns to the depot is called the (batch) completion time and the completion time of a customer order is identical to the completion time of the batch to which the order is assigned. In order picking environments, a particular number of order pickers have to process the customer orders by certain due dates. In distribution warehouses, 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 for production delays to be avoided.

Instead of measuring the quality of a solution in terms of the time which is necessary to complete all customer orders, the batching of customer 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 composition of the batches, the assignment of batches to pickers, the sequence according to which the batches are processed by an order picker and the corresponding start dates of the batches determine the way in which due dates are met.

The Order Batching and Sequencing Problem with Multiple Pickers (OBSPMP) can be formulated as follows: How can a given set of customer orders be grouped into batches, how should these batches be assigned to a limited number of order pickers, and how should the batches be scheduled such that total tardiness is minimized?

A solution to the OBSPMP can be interpreted as a set of batches and corresponding sequences (a sequence for each order picker). A particular order picker starts all batches successively. He/she collects the customer orders included in the first batch of his/her sequence (the batch at position #1). After returning to the depot he/she collects the customer orders in the second batch of his/her sequence (the batch at position #2) and so on.

The width of picking aisles usually allows for overtaking maneuvers. Therefore, the possibility of aisles being blocked by an order picker or traffic jams caused by order pickers having to collect an item from the same storage location is not considered in the following.

2.2 Model Formulation

The following mathematical optimization model is proposed for the OBSPMP defined above. In this model, all feasible batches (i.e. all batches which do not exceed the capacity of the picking device) are considered explicitly. The model requires the following index sets:

K: set of positions (for each order picker) where batches can be scheduled, where $K = \{1, \ldots, \overline{m}\}$; $\overline{m}$ is the upper bound on the number of batches which have to be scheduled by each order picker; this results in at most $\overline{m}$ positions;
J: set of customer orders, where $J = \{1, \ldots, n\}$;
I: set of all feasible batches;
P: set of order pickers, where $P := \{1, \ldots, p_{\max}\}$.

Furthermore, the model requires the following constants:

a_{ij}: coefficient which indicates if customer order j ($j \in J$) is included in batch i ($a_{ij} = 1$) or not ($a_{ij} = 0$);
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 ($j \in J$);
M: a sufficiently large (positive) number;
pt_i: processing time of batch i ($i \in I$).

Since the composition of each batch is known, the tour length (depending on the underlying routing strategy), the search time and pick time can be calculated beforehand. Based on these factors the processing time can be calculated beforehand. Then, a feasible batch i is characterized by the fact that the capacity constraint is not exceeded, i.e. $\sum_{j \in J} c_j a_{ij} \leq C$. Furthermore, the following real-value variables are used:

ct_{pk} : completion time of the batch completed by order picker p ($p \in P$) at position k ($k \in K$);
ta_j : tardiness of customer order j ($j \in J$).

Additionally, two types of binary variables have to be defined:

x_{ipk} : binary decision variable which describes whether batch i ($i \in I$) is completed by order picker p ($p \in P$) at position k ($k \in K$) ($x_{ipk} = 1$) or not ($x_{ipk} = 0$);
y_{jpk} : binary decision variable which describes whether order j ($j \in J$) is assigned to a batch to be completed by order picker p ($p \in P$) at position k ($k \in K$) ($y_{jpk} = 1$) or not ($y_{jpk} = 0$).

The optimization model can be stated as follows:

$$\min \sum_{j \in J} ta_j \tag{2.1}$$

$$\text{s.t.} \sum_{i \in I} x_{ipk} \leq 1, \forall p \in P, \forall k \in K; \tag{2.2}$$

$$\sum_{i \in I} \sum_{p \in P} \sum_{k \in K} a_{ij} x_{ipk} = 1, \forall j \in J; \tag{2.3}$$

$$\sum_{i \in I} a_{ij} x_{ipk} = y_{jpk}, \forall j \in J, \forall p \in P, \forall k \in K; \tag{2.4}$$

$$ct_{pk-1} + \sum_{i \in I} pt_i x_{ipk} \leq ct_{pk}, \forall p \in P, \forall k \in K \setminus \{1\}; \tag{2.5}$$

$$\sum_{i \in I} pt_i x_{ip1} \leq ct_{p1}, \forall p \in P; \tag{2.6}$$

$$ct_{pk} - ta_j \leq d_j + M(1 - y_{jpk}), \forall j \in J, \forall p \in P, \forall k \in K; \tag{2.7}$$

$$ct_{pk} \geq 0, \forall p \in P, \forall k \in K; \tag{2.8}$$

$$ta_j \geq 0, \forall j \in J; \tag{2.9}$$

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

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

The objective function (2.1) minimizes the total tardiness of all orders. The constraints (2.2) guarantee that at most one batch is assigned to a particular position of an order picker. Equations (2.3) make sure that each customer order is assigned to exactly one batch. The consistency among batches and orders scheduled at any particular position by each order picker is expressed by constraints (2.4). The conditions (2.5) guarantee that two batches do not overlap and ensure feasibility with respect to time. Inequalities (2.6) determine the completion time of the first batch of each order picker. Constraints (2.7) restrict the tardiness of each order. Constraints (2.8) and (2.9) impose non-negativity on all variables modeling completion times and tardiness, respectively.

The model can be solved in two different ways: the first consists of generating all batches beforehand and a solution procedure afterwards. The second is a column generation approach where batches are added successively to the optimization problem which is solved several times.

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 interpreted as a special case of the OBSPMP, in which the capacity of the picking device is equal to only one customer order. Therefore, the OBSPMP is also $\mathcal{NP}$-hard.

3 Literature Review

The OBSPMP is closely related to the Order Batching Problem (OBP), which has been discussed frequently in the literature. It consists of the assignment of customer orders to batches in order to minimize the total processing time. In this case, it is sufficient to determine an optimal composition of the batches. In the OBSPMP, however, it is also necessary to determine the sequences according to which the batches are scheduled. A variety of heuristic approaches has been suggested for the OBP. These approaches can be partitioned into priority rule-based algorithms, seed algorithms, savings algorithms, and metaheuristics (Henn et al., 2012). The first three groups contain approaches which are constructive by nature, whereas metaheuristics focus on improving given solutions. For metaheuristics approaches based on the local search principle (Henn et al. (2010): Iterated Local Search, Albareda-Sambola et al. (2009): Variable Neighborhood Search) or population-based metaheuristics (Tsai et al. (2008): Genetic Algorithms, Henn et al. (2010): Ant Colony Optimization) are suggested. For a detailed survey on variants and solution approaches for order batching it is referred to de Koster et al. (2007) and Henn et al. (2012).

In manual order picking systems, three planning issues can be distinguished at the operative level (Caron et al., 1998): the assignment of articles to storage locations (article location), the transformation of customer orders into picking orders (order batching) and the routing of pick-

ers through the warehouse (picker routing). The simultaneous solution of these problems, which would represent a global optimum, is not a very realistic approach (Wäscher, 2004). In practice, the decisions for the planning issues are made sequentially. Existing order batching methods consider the routing policy as given (de Koster et al., 2007). Because of this, only a few integrated approaches have been proposed to date that solve the batching and routing problems simultaneously in order to minimize the total processing time (e.g., Kulak et al. (2011)). However, in numerical experiments it was shown that the impact of efficient solutions to order batching is larger than the impact of the routes through the warehouse on the total warehouse performance (Hsieh and Huang, 2011).

Henn and Schmid (2011) suggested two heuristics (Iterated Local Search (ILS) and Attribute-Based Hill Climber (ABHC)) for the special case of the OBSPMP in which only a single order picker is available (called OBSP). ILS consists of two phases, a perturbation and a local search phase. In the perturbation phase, an incumbent solution is partially modified. In the local search phase, proceeding from this solution, an improved solution is sought. ABHC is an almost parameter-free heuristic based on a simple Tabu Search principle: A set of attributes is introduced for each problem. An attribute can be any specific solution feature. During a local search phase a solution can become an incumbent solution if and only if it represents the smallest objective function value found so far for at least one attribute. The algorithm terminates if the current neighborhood does not contain a solution representing a best solution for at least one attribute. In numerical experiments, the proposed methods provided solutions which improved a constructive heuristic by 46% on average.

For the solution of the OBSP in an automated storage and retrieval system (AS/RS), Elsayed and Lee (1996) proposed an approach for the construction of batches and the sequence according to which these batches are scheduled. At first, the customer orders are sorted according to their due dates and the times are calculated which would be needed if each customer order were processed in a single batch. An initial solution consists of batches in which each order is collected on a single tour and by scheduling the orders according to its position in the sorted list. In order to improve this solution, rules are proposed for the generation of batches. The Nearest-Schedule Rule determines the first customer order of the obtained sequence to be the seed order and assigns it to the first batch. Additional customer orders in the sequence are assigned to the seed if their inclusion does not increase the total tardiness and does not violate the capacity restriction. When no further customer orders can be added, the first unassigned customer order in the sequence serves as the seed order for the next batch.

Elsayed et al. (1993) suggested a method for minimizing the total tardiness and earliness of all customer orders in an AS/RS. Their solution 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 according to their priority index. This sequence is updated if it can be identified that swapping two adjacent customer orders 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 an improved solution. Start dates are determined for each of the batches obtained (step 3).

These two approaches only consider a single AS/RS machine and are of limited applicability to picker-to-parts systems. In contrast to the situation in AS/AR systems, here the processing time of a batch containing only one order (called single processing time) differs significantly from the processing time of a batch to which more than one customer order is assigned. Both approaches

use the single processing times in order to determine a sequence of batches. Therefore, they will not yield competitive results for the OBSPMP.

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 earliness and tardiness are penalized. A genetic algorithm is introduced. Unlike in the approaches discussed previously, the order integrity condition is not required, i.e. the items required for a single customer order may be assigned to different batches. This approach will not be considered here further because of this specific condition.

4 Variable Neighborhood Search: General Principle

4.1 Fundamentals

In combinatorial optimization problems, a solution s has to be found in the set of all feasible solutions S which has a minimal (or maximal) value for an objective function $f(\cdot)$. In the following section, the focus is on minimization problems, which can be described as $\min\{f(s)|s \in S\}$. A simple and widely-used solution method for combinatorial optimization problems is local search (LS). Local search-based heuristics explore the neighborhood of a solution in order to identify a new solution with a smaller objective function value. For a solution s, a solution is called neighbor solution of s if it can be obtained by applying a single local transformation ("move") to s. Classic local search selects one element of the neighborhood of the incumbent solution to be the next incumbent solution in each iteration. Either the first element generated with a smaller objective function value (first improvement strategy) is selected and becomes the next incumbent solution or all elements in the current neighborhood are evaluated and the one with the smallest objective function value is choosen (best improvement strategy). This results in a sequence of solutions $s_0, s_1, s_2, \ldots$, where each member of the sequence is a neighbor of its predecessor and possesses a smaller objective function value than the previous one. Classic local search stops at a local minimum, i.e. when no neighbor solution can be identified that has a smaller objective function value than the incumbent solution. The objective function value of the local minimum may be far from the optimal objective function value.

The quality of the solution provided by local search depends on the choice of the neighborhood structure. A variety of different neighborhood structures can be defined for each optimization problem. A local minimum with respect to one neighborhood structure is not necessarily a local minimum with respect to a different neighborhood structure. However, a global minimum represents a local minimum for all possible neighborhood structures. In local search selecting a solution from a neighborhood as the next incumbent solution can be done in a deterministic or in a stochastic way (Hansen et al., 2010). In the following section, two general principles are introduced which incorporate several neighborhood structures.

4.2 Variable Neighborhood Descent

A straightforward deterministic local search approach dealing with different neighborhood structures is the Variable Neighborhood Descent (VND) approach (Hansen and Mladenović, 2001). VND requires a sequence of neighborhood structures $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{l_{\max}^{\text{VND}}}$. VND starts with an incumbent solution s_{inc} and explores $\mathcal{N}_1$. In each iteration the algorithm identifies the element of the current neighborhood structure with the smallest objective function value. If a solution with an improved objective function value can be identified, this solution becomes the next incumbent solution and the algorithm continues to explore $\mathcal{N}_1$. Otherwise (i.e. a local minimum with respect

to the current neighborhood structure l has been found), the algorithm explores the neighborhood structure $l + 1$. The algorithm terminates if the incumbent solution represents a local minimum for all neighborhood structures. The general principle of VND is summarized in Algorithm 4.1.

Algorithm 4.1 Variable Neighborhood Descent: General Principle

Input: neighborhood structures $\mathcal{N}_l, l = 1, \ldots, l_{\max}^{\text{VND}}$;
1: generate initial solution s_{ini};
2: $s_{inc} := s_{ini}$;
3: $l := 1$;
4: **repeat**
5: $s' := \arg\min\{f(s)|s \in \mathcal{N}_l(s_{inc})\}$;
6: **if** $f(s') < f(s_{inc})$ **then**
7: $s_{inc} := s'$;
8: $l := 1$;
9: **else**
10: $l := l + 1$;
11: **end if**
12: **until** $l > l_{\max}^{\text{VND}}$.

4.3 Variable Neighborhood Search

The general principle of Variable Neighborhood (VNS) was developed and introduced by Mladenović and Hansen (1997). VNS explores the different neighborhoods in a stochastic and deterministic way. It is composed of two phases, a (stochastic) shaking and a (deterministic) local search phase. For the shaking phase the $l_{\max}^{\text{VNS}}$ neighborhood structures of a solution s are notated with $\mathcal{N}_l(s)$, whereas l denotes the l-th neighborhood structure ($l \in \{1, \ldots, l_{\max}^{\text{VNS}}\}$). VNS starts with an initial solution s_{ini} and $l = 1$. The initial solution is improved by a local search phase. The solution stemming from this phase becomes the incumbent solution s_{inc}. Starting with the first neighborhood structure, in each iteration a solution s' is randomly chosen from the l-th neighborhood structure of the incumbent solution (shaking phase). This solution s' is then improved by a deterministic local search procedure. Several neighborhood structures can be used for this local search phase. The local search phase generates a local minimum s''. If s'' has a smaller objective function value than the incumbent solution it becomes the next incumbent solution and l is reset to 1. Otherwise, l is set to $l + 1$ (if $l + 1 > l_{\max}^{\text{VNS}}$, the parameter l is set to 1) and a solution from the next neighborhood of the incumbent solution is randomly selected and locally optimized. This procedure is repeated until a given termination criterion is met. The incumbent solution s_{inc} represents the best solution found by the algorithm. VNS is summarized in Algorithm 4.2.

5 Variable Neighborhood Search: Application to the OBSPMP

5.1 Initial Solution

VND as well as VNS require the generation of an initial solution. In the following a constructive solution approach, called *Earliest Start Date* (ESD) rule, is proposed. This is a priority rule-based algorithm consisting of two steps. In the first step (sequencing step), priorities are assigned to the customer orders. In the second step (dispatching step), in accordance with the previously

Algorithm 4.2 Variable Neighborhood Search: General Principle

Input: neighborhood structures $\mathcal{N}_l, l = 1, \ldots, l_{\max}^{\text{VNS}}$;
1: generate initial solution s_{ini};
2: s_{inc} :=local_search(s_{ini});
3: **repeat**
4: $l := 1$;
5: **repeat**
6: generate a solution s' at random from $\mathcal{N}_l(s_{inc})$ (shaking);
7: s'' :=local_search(s');
8: **if** $f(s'') < f(s_{inc})$ **then**
9: $s_{inc} := s''$;
10: $l := 1$;
11: **else**
12: $l := l + 1$;
13: **end if**
14: **until** $l > l_{\max}^{\text{VNS}}$.
15: **until** termination condition is met

assigned priorities, these customer orders are assigned successively to the batches ensuring that the the capacity constraint is not violated.

In the first step, all orders are sorted according to their due dates in an ascending order. In the second step, a customer order is assigned to a batch with minimal start date on the currently last position of an order picker. It is checked for each order picker if the order can be assigned to the batch currently on his/her last position. In case that the capacity remaining for this batch is not sufficient, the start date of a new batch is calculated for each order picker. The order is then assigned to the order picker and the corresponding batch with the earliest start date. The algorithm is summarized in Algorithm 5.1.

Algorithm 5.1 Earliest Start Date

1: sort the set of orders according to their due dates in an ascending order;
2: **while** there are unassigned orders **do**
3: let j be the first order in the sorted list;
4: **for all** order pickers $p \in P$ **do**
5: **if** j can be assigned to the last batch of picker p **then**
6: $sd_p :=$ start date of the last batch of picker p (CASE 1);
7: **else**
8: $sd_p :=$ completion time of the last batch of picker p (CASE 2);
9: **end if**
10: **end for**
11: $p^\star := \arg\min_{p \in P} sd_p$;
12: assign j to the last batch of picker $p^\star$ (CASE 1) or open a new batch for picker $p^\star$ and assign j to it (CASE 2);
13: remove j from the list of unassigned orders;
14: **end while**

5.2 Neighborhood Structures

A variety of possible neighborhood structures can be identified for the problem defined above. Two classes of neighborhoods are considered. The first class contains neighborhood structures

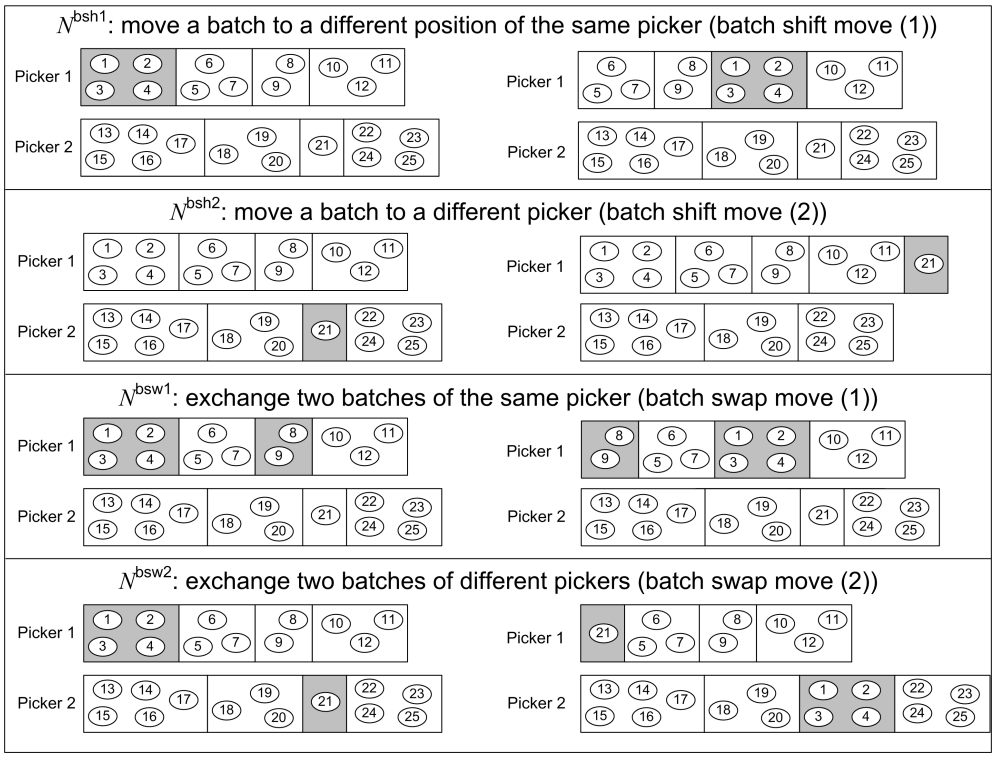

Figure 5.1: Examples of the proposed neighborhood structures considering batches

which incorporates transformations (moves) that change the position of a complete batch. The second class of neighborhood structures changes the position of a customer order. In the first class of neighborhood structures, the move can either be restricted to a single order picker or to two order pickers. A batch can either be shifted (a batch is moved to a different position) or two batches can be exchanged/swapped. To sum up, the following neighborhood structures are considered:

$\mathcal{N}^{bsh1}$: move a batch to a different position in the batch sequence of the same picker (batch shift move (1));
$\mathcal{N}^{bsh2}$: move a batch to a position in the batch sequence of a different picker (batch shift move (2));
$\mathcal{N}^{bsw1}$: exchange two batches of the same picker (batch swap move (1));
$\mathcal{N}^{bsw2}$: exchange two batches of different pickers (batch swap move (2));

Examples of the neighborhood structures are given in Figure 5.1. Neighborhood structures of the second class, which change the position of a customer order, are also proposed analogous to these structures. Again, the move can be restricted to one or to several order pickers. Also, a customer order can be shifted or two orders can be swapped. This results in the following structures:

$\mathcal{N}^{osh1}$: move an order to a different batch of the same picker (order shift move (1));
$\mathcal{N}^{osh2}$: move an order to a different picker (order shift move (2));
$\mathcal{N}^{osw1}$: exchange two orders of the same picker (order swap move (1));
$\mathcal{N}^{osw2}$: exchange two orders of different pickers (order swap move (2));

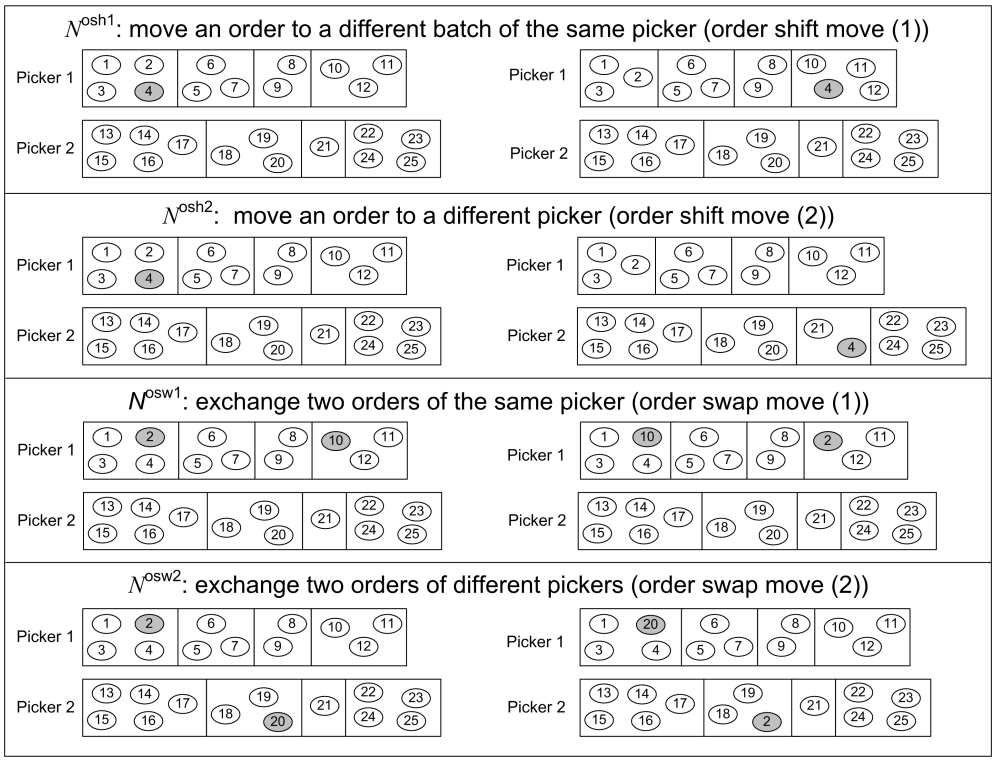

Figure 5.2: Examples of the proposed neighborhood structures considering customer orders

Examples of the neighborhood structures are shown in Figure 5.2. In the approach presented here, only neighbors which represents feasible solutions are considered, i.e. transformations are considered which do not violate the capacity constraint. It should be noted that a variety of other moves can be proposed for the OBSPMP (cf. Albareda-Sambola et al. (2009)).

5.3 Variable Neighborhood Descent

When applying VND, it is only necessary to specify the initial solution and the type and sequence of the neighborhood structures. For the generation of the initial solution, the ESD-rule has been implemented. Here, numerical pre-tests have shown that the sequence $\mathcal{N}_1 := \mathcal{N}^{\text{bsw2}}, \mathcal{N}_2 := \mathcal{N}^{\text{osh1}}, \mathcal{N}_3 := \mathcal{N}^{\text{osh2}}, \mathcal{N}_4 := \mathcal{N}^{\text{osw1}}, \mathcal{N}_5 := \mathcal{N}^{\text{osw2}}$ results in the smallest total tardiness.

5.4 Variable Neighborhood Search

VNS requires the specification of the following components: type of the initial solution, type and sequence of the neighborhood structures used in the shaking phase, the design of the local search phase, and the termination condition. For the shaking phase, implemented here, the sequence $\mathcal{N}_1 := \mathcal{N}^{\text{osw2}}, \mathcal{N}_2 := \mathcal{N}^{\text{osh2}}, \mathcal{N}_3 := \mathcal{N}^{\text{bsw2}}$ and $\mathcal{N}_4 := \mathcal{N}^{\text{bsh2}}$ is used which results in $l_{\max}^{\text{VNS}} = 4$. As described above, only feasible solutions are considered. In order to guarantee feasibility the order shift move and the order swap move are designed as follows: An order and a picker as well as a batch to which the order should be assigned are selected in the order shift move. In case that the order cannot be included in this batch the order will be assigned to a new batch scheduled after the last position of the picker. In the order swap move two customer orders are selected. If the exchange would violate the capacity restriction of one batch a new batch will be opened which is scheduled after the last position of the selected picker.

Two neighborhoods are incorporated in the local search phase, namely $\mathcal{N}^{\text{osh1}}$ and $\mathcal{N}^{\text{osw1}}$. Therefore, the assignment of the customer orders to order pickers remains unchanged during the local search phase. An improvement to the incumbent solution is sought by a sequence of order shift moves (1). The best move of all possible improvements is identified and performed. If an improvement can be obtained, one proceeds from the new solution and searches for another improvement by an order shift move (1). In cases where no order shift move (1) can be identified which reduces the total tardiness, an improvement is sought by a sequence of order swap moves (1). If no further improvement can be identified by order swap moves (1), the algorithm searches again for an order shift move (1) that leads to an improved solution. These steps are repeated until no further improvement by either order shift moves (1) or order swap moves (1) is possible.

Two different variants are proposed for the VNS termination condition. The first (VNS-1) allows for the same computing time which is required for VND. The second (VNS-2) terminates after n calls of the outer loop (lines 3 - 15) of the algorithm (variant VNS-2), where n is the number of orders.

6 Test Design

6.1 Warehouse Parameters

The performance of the proposed approaches are evaluated in a series of numerical experiments. A single-block layout with 900 storage locations and two cross aisles, one at the front and one at the back of the picking area, is considered for all experiments. The aisles are vertically orientated and the depot is located in front of the leftmost aisle (cf. Figure 2.1). The picking area is partitioned into 10 (picking) aisles with 90 storage locations each (45 on each side of every aisle). The aisles are numbered from 1 to 10, where aisle #1 is the leftmost aisle and aisle #10 the rightmost aisle. The length of each storage location amounts to one length unit (LU). When picking an item, the order picker is positioned in the in the center of the storage location. Whenever an order picker leaves an aisle, he/she has to move one LU in a 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, an order picker has to move 5 LU in a horizontal direction. The distance between the depot and the first storage location of the leftmost aisle amounts to 1.5 LU.

The S-shape and the largest gap heuristic are considered for the routing strategy. When the S-shape heuristic is applied, an aisle has to be traversed entirely if an article has to be picked. In largest gap routing, an order picker enters and leaves a picking aisle from the same cross aisle (cf. Figure 2.1).

In the experiments a class-based storage assignment of articles to storage locations is assumed: The articles are grouped into three classes A, B and C according to their demand frequency. Class A contains articles with a high, B with a medium and C with a low demand frequency. Articles of class A are only stored in aisle #1, articles of class B in aisles #2, #3 and #4, and articles of class C only in the remaining six aisles. Within a class, the location of each article is determined randomly. Furthermore, it is assumed that 52 percent of the demand is for articles in class A, 36 percent for articles in B and 12 percent for articles in C. These frequencies have been used in different numerical experiments, since they represent a typical real-world order picking system (de Koster et al., 1999). Petersen and Schmenner (1999) have shown that this kind of assignment should be preferred for the two routing strategies considered, S-shape and largest gap.

As described in Section 2, the processing time is composed of the travel time, the search time, the pick time and the setup time. As for the travel time, an order picker walks 20 length units in one minute. Moreover, he/she needs 10 seconds to search and pick an item from a storage

location. Each batch requires a setup time of 3 minutes.

6.2 Problem Sets

In the numerical experiments the number of customer orders (n) is set to 100 and 200. For each order, the number of articles is uniformly distributed in $\{5, 6, \ldots, 25\}$. The capacity of the picking device (C) is 45 and 75 items (defining the maximal number of items that can be collected on a tour). For the number of order pickers $p_{\max}$, we consider problem classes with 2, 3, and 5 order pickers. The due date for each order is chosen from the interval $[\min_{j \in J} pt_j^s, (2 \cdot (1 - \text{MTCR}) \sum_{j \in J} pt_j^s + \min_{j \in J} pt_j^s)/p_{\max}]$, where pt_j^s is the processing time of order j ($j \in J$), if the order is processed as a single order in a batch without any other customer orders (Elsayed and Lee, 1996). MTCR stands for *modified traffic congestion rate*. This rate describes the tightness of the problem with respect to the due dates. A high value indicates that the due dates are very close to each other and therefore cannot be met. For the MTCR, the values 0.6, 0.7, 0.8 (low, medium, high) are investigated. In combination with the two routing strategies (S-shape, largest gap) we obtain 72 problem classes, and for each of these classes 50 instances are generated.

6.3 Reference Values and Implementation

Application of the ESD-rule (presented in Section 5.1) represents a straightforward way to generate a solution to the OBSPMP. For each problem instance, the total tardiness obtained by application of ESD serves as a reference value. The solution quality of all other strategies is measured in relation to that of ESD. The improvement of the total tardiness tar of a solution s compared to the total tardiness of a solution s_{ESD} provided by ESD is calculated as $100 \cdot (tar(s_{\text{ESD}}) - tar(s))/tar(s_{\text{ESD}})$.

The computations for all 3,600 instances are 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 Total Tardiness

Tables 7.1, 7.2, and 7.3 depict the solution qualities of the evaluated heuristics when 2, 3 or 5 order pickers operate in the order picking environment, respectively. The average total tardiness (tar) in minutes obtained by each approach is presented for each problem class. Furthermore, the tables show the corresponding (average) improvements compared to the total tardiness of ESD (impr) and the (average) number of tardy orders (no_tar). The smallest total tardiness observed for each problem class is shown in bold.

For VND, the average improvement (obtained over all problem classes) amounts to 39%; the improvements range from 23% ($p_{\max} = 2$; S-shape; $n = 100$; $C = 45$; MTCR = 0.6) to 64% (2; S-shape; 200; 75; 0.8). The total tardiness obtained by VNS within the same computing times required by VND (called VNS-1) improves the tardiness of ESD by 20%, whereas a minimal improvement of 3% (3; S-shape; 100; 45; 0.6) and a maximal improvement of 53% (2; S-shape; 200; 75; 0.8) can be observed. In cases where the termination condition of VNS is set to n iterations (VNS-2) the solutions generated by ESD are improved by 43% on average. For VNS-2 improvements are contained in the interval [21%; 65%] where the limits originate from the problem classes (5; S-shape; 100; 45; 0.7) and (2; S-shape; 200; 75; 0.8). In general, VND finds the smallest tardiness for 41 (out of 72) problem classes and VNS-2 for 34 classes.

It has to be noted that small – almost negligible – tardiness can be observed for small MTCR values. For example, the total tardiness for problem class (2; S-shape; 100; 45; 0.6) amounts to 49 minutes for ESD. For this problem class, applying VND leads to a total tardiness of 35 minutes, which represents an absolute improvement of 14 minutes (0.14 min per order), whereas the relative improvement amounts to 23%. Therefore, conclusions and interpretations based on relative improvements have to be drawn carefully for problem classes with a small MTCR. The results obtained by VND and VNS demonstrate that the application of a local search-based heuristic can improve the total tardiness significantly. The large reductions resulting from the application of the metaheuristics – in comparison to ESD – can be explained as follows: VND and VNS are able to reduce the total processing time and the number of batches substantially. This advantage can be identified by analyzing the total processing time. The shorter total processing time means that the total tardiness can also be reduced. The reduction of the total processing time does not only decrease the tardiness of a single order but may also have an impact on the tardiness of more than one order. In addition to the reduction of the tardiness by minimizing the total processing time, the tardiness can further be improved by other exchanges. VNS and VND are able to postpone an order with an early due date to a later point in time, accepting that the due date of this order will be missed, if the later release of this order can be used to complete other orders before their due dates.

The results show that VNS is only able to generate solutions with a smaller tardiness than those of VND if the computing time is longer (variant VNS-2). Therefore, it can be stated that VND outperforms VNS-1. This result was expected since VND evaluates all elements in the neighborhood and proceeds with an element in the neighborhood with the smallest total tardiness. In the shaking phase, VNS-1 selects an element in the neighborhood randomly. Since the computing times are identical for both heuristics, it can be concluded that a large number of moves is selected by VNS-1 which are not useful for improving the best found solution.

Also, in cases where more shaking steps can be applied (VNS-2), VNS is not able to obtain significantly improved solutions in comparison to VND. Whereas the average improvements for VND are nearly identical independent of the different numbers of order pickers, the improvements are smaller for VNS-2. In detail, the average improvements for two order pickers amount to 50% and 47%. For $p_{\max} = 3$, only improvements of 44% and 41% can be observed, and for five order pickers VNS-2 is only able to improve the results of ESD by 39% and 36%. The results imply that additional computing time cannot be used to identify the moves which result in a smaller total tardiness. A further disadvantage of VNS becomes evident: The assignment of orders to pickers is only done by random exchanges, whereas the moves which change the assignment of orders to pickers in VND are done in a deterministic way. The results show that the random exchanges are inappropriate (too small) to successfully assign orders to pickers. To sum up, the deterministic (and more systematic) search outperforms the stochastic variant if the proposed neighborhood structures are used.

In the following paragraph, the way in which the total tardiness is influenced by the parameters n, C, MTCR, the routing strategy and $p_{\max}$ will be analyzed. For n, C, MTCR and the routing strategies, the effects are in line with the findings of Henn and Schmid (2011), where there is only one order picker. In detail, the total tardiness increases with an increasing number of orders. In problem classes in which a large capacity is used, more orders can be collected on one tour than in cases with the small capacity. This results in a shorter processing time and therefore a smaller tardiness. It can also be observed that a larger MTCR results in a larger total tardiness. This can also be seen from the fact that for small MTCR values, nearly all orders can be completed before their due date, whereas for an MTCR of 0.8 it is not possible to meet even 50% of the due dates. For the majority of the problem classes, the tardiness obtained in the problem classes for

which routes are determined by the S-shape heuristic is smaller than in the corresponding problem classes to which the largest gap heuristic is applied. This small advantage is caused by the fact that for problem classes with a large capacity, S-shape routing is able to provide shorter tour lengths than largest gap routing which results in shorter processing times and in a smaller total tardiness. When analyzing the different number of order pickers it can be noted that the total tardiness is larger for the problem classes with two order pickers compared to the corresponding problem classes for which 3 or 5 order pickers are assumed.

Table 7.1: Solution quality for $p_{\max} = 2$

Routing	n	C	**MTCR**	**ESD**		**VND**			**VNS**					
									VNS-1			VNS-2		
				tar	no_tar	tar	impr	no_tar	tar	impr	no_tar	tar	impr	no_tar
S-shape	100	45	0.6	49	7	35	23%	5	42	8%	6	**34**	39%	5
	100	45	0.7	488	43	**226**	51%	22	306	32%	27	233	49%	21
	100	45	0.8	5087	98	2656	48%	63	2969	42%	67	**2597**	49%	60
	100	75	0.6	57	6	43	29%	5	53	8%	6	**32**	51%	4
	100	75	0.7	164	16	113	27%	12	141	8%	14	**107**	40%	11
	100	75	0.8	2065	90	1021	51%	57	1250	39%	62	**1003**	52%	55
	200	45	0.6	71	10	**43**	31%	7	55	13%	8	**43**	40%	7
	200	45	0.7	930	69	**302**	61%	25	421	47%	33	317	59%	25
	200	45	0.8	19947	198	9443	53%	120	10334	48%	131	**9203**	54%	118
	200	75	0.6	71	7	50	33%	6	66	8%	7	**37**	60%	5
	200	75	0.7	157	16	106	31%	11	138	9%	13	**101**	43%	11
	200	75	0.8	6824	187	2551	64%	97	3229	53%	111	**2504**	65%	98
	Average						42%			26%			50%	
l. gap	100	45	0.6	62	9	**46**	29%	7	57	11%	8	**46**	37%	8
	100	45	0.7	595	53	269	58%	27	354	42%	34	**267**	56%	27
	100	45	0.8	5308	98	2984	44%	67	3294	38%	70	**2915**	45%	63
	100	75	0.6	54	6	44	24%	5	52	6%	6	**29**	61%	5
	100	75	0.7	145	16	109	25%	13	131	8%	14	**105**	34%	12
	100	75	0.8	2463	95	1438	42%	65	1667	32%	68	**1407**	43%	60
	200	45	0.6	79	10	**53**	29%	7	65	13%	8	54	38%	7
	200	45	0.7	1686	100	594	61%	40	758	49%	49	**580**	60%	40
	200	45	0.8	20975	198	10933	48%	127	11898	43%	136	**10690**	49%	126
	200	75	0.6	67	7	50	28%	6	63	7%	7	**38**	54%	6
	200	75	0.7	181	17	132	27%	13	155	9%	15	**128**	33%	13
	200	75	0.8	8658	192	4181	52%	118	4883	44%	126	**4079**	53%	113
	Average						40%			25%			47%	

legend: 'tar': average total tardiness in minutes; 'impr': (average) improvements compared to the total tardiness of ESD; 'no_tar': (average) number of tardy orders;

Table 7.2: Solution quality for $p_{\max} = 3$

Routing	n	C	**MTCR**	**ESD**		**VND**			**VNS**					
									VNS-1			VNS-2		
				tar	no_tar	tar	impr	no_tar	tar	impr	no_tar	tar	impr	no_tar
S-shape	100	45	0.6	51	8	**35**	32%	7	47	7%	8	**35**	38%	7
	100	45	0.7	416	46	**208**	44%	26	313	17%	35	251	30%	28
	100	45	0.8	3356	97	**1846**	45%	66	2170	35%	70	1888	44%	63
	100	75	0.6	78	171	56	32%	7	72	9%	8	**38**	60%	6
	100	75	0.7	142	17	95	31%	13	132	5%	16	**93**	39%	12
	100	75	0.8	1500	92	**823**	45%	61	1073	28%	67	838	44%	58
	200	45	0.6	77	11	**50**	34%	8	70	5%	10	55	30%	8
	200	45	0.7	891	81	**348**	58%	34	551	33%	48	432	46%	38
	200	45	0.8	12939	198	**6347**	51%	120	7260	44%	132	6502	50%	121
	200	75	0.6	78	8	56	31%	7	74	6%	8	**38**	61%	6
	200	75	0.7	164	18	**112**	30%	14	154	4%	17	115	33%	14
	200	75	0.8	4840	189	**1967**	60%	105	2710	44%	119	2113	57%	107
	Average						41%			20%			44%	
l. gap	100	45	0.6	67	11	**49**	29%	9	64	4%	11	52	33%	9
	100	45	0.7	502	54	**273**	46%	32	378	19%	41	317	32%	35
	100	45	0.8	3595	98	**2095**	42%	68	2416	33%	72	2132	41%	65
	100	75	0.6	79	172	58	31%	8	75	7%	9	**41**	58%	7
	100	75	0.7	169	20	121	27%	16	160	4%	19	**119**	33%	16
	100	75	0.8	1823	94	**1123**	38%	67	1344	26%	69	1145	37%	62
	200	45	0.6	66	10	**45**	32%	8	62	5%	10	50	34%	8
	200	45	0.7	1026	92	**426**	57%	40	651	32%	59	515	45%	47
	200	45	0.8	13856	199	**7513**	46%	131	8382	40%	136	7561	45%	129
	200	75	0.6	75	9	55	31%	8	72	7%	9	**39**	56%	7
	200	75	0.7	198	23	**143**	30%	17	187	5%	22	152	29%	19
	200	75	0.8	6162	195	**3186**	49%	125	3917	36%	131	3298	47%	117
	Average						38%			18%			41%	

Table 7.3: Solution quality for $p_{\max} = 5$

Routing	n	C	**MTCR**	**ESD**		**VND**			**VNS**					
									VNS-1			VNS-2		
				tar	no_tar	tar	impr	no_tar	tar	impr	no_tar	tar	impr	no_tar
S-shape	100	45	0.6	69	11	**47**	31%	9	66	3%	11	52	31%	10
	100	45	0.7	331	52	**176**	47%	31	290	9%	44	249	21%	38
	100	45	0.8	2142	98	**1231**	43%	67	1503	30%	69	1314	39%	65
	100	75	0.6	121	13	82	33%	11	110	11%	13	**54**	60%	10
	100	75	0.7	189	24	128	32%	18	181	4%	23	**120**	38%	19
	100	75	0.8	1177	93	**699**	41%	68	923	21%	70	743	37%	64
	200	45	0.6	76	13	**52**	35%	10	73	4%	12	57	36%	11
	200	45	0.7	491	72	**218**	50%	34	391	13%	56	334	23%	47
	200	45	0.8	7780	197	**3985**	49%	125	4853	38%	130	4302	45%	123
	200	75	0.6	132	14	92	31%	11	127	6%	14	**65**	57%	11
	200	75	0.7	217	27	**140**	34%	19	207	3%	26	158	29%	21
	200	75	0.8	3493	190	**1625**	54%	119	2330	33%	128	1860	47%	115
	Average						40%			15%			39%	
l. gap	100	45	0.6	73	13	**53**	30%	11	70	4%	13	56	29%	11
	100	45	0.7	372	58	**208**	46%	37	334	8%	50	284	22%	43
	100	45	0.8	2194	98	**1336**	39%	71	1595	27%	72	1427	35%	68
	100	75	0.6	131	15	92	31%	12	117	12%	14	**59**	60%	11
	100	75	0.7	249	31	178	29%	24	233	6%	29	**173**	35%	25
	100	75	0.8	1432	95	**963**	33%	72	1138	20%	72	974	32%	68
	200	45	0.6	82	15	**57**	34%	11	80	5%	14	68	30%	13
	200	45	0.7	716	98	**320**	55%	47	573	16%	76	482	28%	64
	200	45	0.8	8395	197	**4711**	44%	134	5507	34%	138	4985	41%	129
	200	75	0.6	138	16	102	27%	13	132	5%	16	**74**	52%	12
	200	75	0.7	243	29	**172**	30%	22	236	3%	29	185	29%	25
	200	75	0.8	4155	192	**2299**	45%	128	2908	30%	133	2488	40%	121
	Average						40%			14%			36%	

7.2 Computing Times

The computing times required by the proposed heuristics are presented in Table 7.4. By construction, the computing times required by VNS-1 are identical to those of VND. The computing times required by ESD are shorter than one second and are, therefore, omitted. Computing times increase for all algorithms with an increasing number of orders. The computing times required for instances with a large capacity are longer than those where the capacity of the picking device is small. For instances with a small MTCR, the computing times are shorter than for those with a larger MTCR. This can be attributed to the fact that the problem classes are not very challenging from a computational point of view. Moreover, the computing times required for those problem instances in which tours are generated by the largest gap heuristic are longer than in cases where the S-shape routing is applied, since the computations of the tour lengths require a larger amount of time for the largest gap heuristic than for S-shape routing. When comparing the number of order pickers, it can be identified that a smaller $p_{\max}$ results in the longest computing times if the other instance characteristics remain constant.

A further advantage of VND becomes evident when comparing the computing times for VND and VNS-2. The required times of VND are significantly shorter than those for VNS-2. These long times can be attributed to the fact that VNS-2 has to conduct a time consuming local search phase after each shaking step. Since VNS-2 is not able to provide a significantly superior solution than VND does in a longer computing time, the VND approach outperforms VND in terms of solution quality and computing time.

Figure 7.1 shows the development of the solution quality for VNS-2 for selected problem classes. Each of the functions represents the development of the solution quality of VNS-2 for one problem class. The x-axis depicts the computing time of VNS-2, with 100% representing the termination time. The y-axis shows the extent to which the best solution which has been found by a certain point in time deviates from the best found solution. For this representation, the deviation from the best found total tardiness is measured in 10 intervals of identical size. Each point on the graph represents the average deviation computed over all 50 instances in a problem class.

The functions for an MTCR of 0.8 are very similar to each other, independent of the number of order pickers under consideration. In contrast to an MTCR of 0.7, the functions differ for every problem class. After 30% of the computing time, the deviation in the problem class with $p_{\max} = 3$ is greater than 12.5% whereas for the problem class with 2 or 5 order pickers a deviation of less than 10% can be observed. For the problem class with an MTCR of 0.8, only 30% of the computing time is required to obtain a deviation of 5%. Also the behaviour of the functions is more asymptotical to the x-axis than in cases when the MTCR is equal to 0.7. It can be concluded from both plots that the entire computing time needs to be exploited in order to identify the best solution. A similar analysis for VND is omitted because of the very short computing times.

Table 7.4: Computing times [in sec.]

$p_{\max}$	n	C	MTCR	S-shape		largest gap	
				VND	VNS-2	VND	VNS-2
2	100	45	0.6	0.4	122.2	0.2	68.1
2	100	45	0.7	1.8	211.7	1.2	108.5
2	100	45	0.8	5.7	128.0	2.5	64.5
2	100	75	0.6	0.6	133.3	0.3	78.9
2	100	75	0.7	0.7	231.1	0.4	129.6
2	100	75	0.8	4.8	259.5	2.3	138.7
2	200	45	0.6	0.7	237.9	1.1	490.6
2	200	45	0.7	3.4	563.7	9.3	1229.6
2	200	45	0.8	15.7	300.6	25.2	580.3
2	200	75	0.6	0.7	237.9	1.3	532.3
2	200	75	0.7	0.9	429.4	2.0	987.2
2	200	75	0.8	11.2	660.4	25.4	1341.6
3	100	45	0.6	0.5	76.7	0.3	43.6
3	100	45	0.7	1.6	106.2	0.9	53.9
3	100	45	0.8	5.0	77.6	2.2	41.1
3	100	75	0.6	0.8	106.7	0.4	59.6
3	100	75	0.7	0.9	154.6	0.5	82.0
3	100	75	0.8	3.4	149.4	2.5	82.6
3	200	45	0.6	0.6	161.9	1.2	325.2
3	200	45	0.7	3.1	326.4	6.2	658.6
3	200	45	0.8	13.2	179.1	23.2	350.6
3	200	75	0.6	0.8	177.0	2.0	374.2
3	200	75	0.7	1.0	308.4	2.4	714.9
3	200	75	0.8	10.3	366.0	22.9	797.1
5	100	45	0.6	0.6	40.8	0.3	21.9
5	100	45	0.7	1.6	42.2	0.8	23.1
5	100	45	0.8	3.9	43.1	2.1	22.4
5	100	75	0.6	1.1	66.1	0.6	36.5
5	100	75	0.7	1.1	75.3	0.7	40.5
5	100	75	0.8	3.1	77.0	1.9	44.1
5	200	45	0.6	0.7	90.5	1.5	187.1
5	200	45	0.7	2.1	127.8	5.7	253.4
5	200	45	0.8	10.2	97.5	18.9	196.4
5	200	75	0.6	1.2	118.3	2.6	272.9
5	200	75	0.7	1.3	167.3	3.3	375.8
5	200	75	0.8	8.0	188.9	19.0	414.2

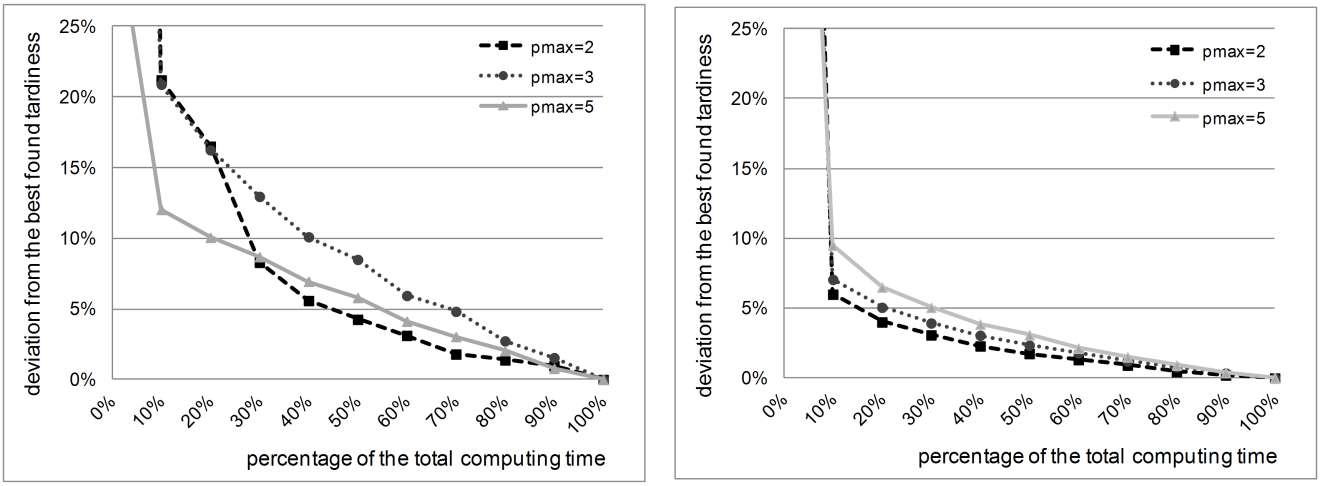

Figure 7.1: Development of solution quality over time for VNS-2 for problem classes with S-shape routing, $n = 200$, $C = 45$ and MTCR = 0.7 (left) and 0.8 (right)

7.3 Influence of the Neighborhood Structures

Table 11 depicts the impact on VND of the different neighborhood structures for the problem classes with $n = 200$, $C = 45$ and for which S-shape routing is applied. The number of times the best solution was improved on is shown (no. best solution). Furthermore, the remaining columns contain the number of improvements made to the best solution found by a move as a proportion of the total number of all improvements. As indicated in the table, the number of improved solutions increases with a rising MTCR. The impact of a neighborhood structure depends on the instance characteristics. For example, the proportion of improved solutions found in $\mathcal{N}^{\text{bsw2}}$ is small for MTCR = 0.6 and is large for MTCR = 0.8, whereas the behavior is opposite for $\mathcal{N}^{\text{osh1}}$.

Similary, Table 7.6 shows the impact of the different neighborhood structures on the solution quality for VNS-2. The table depicts how often an improved solution can be identified after a complete local search phase and which kind of neighborhood generates the initial solution for this local search phase. As has already been observed for VND the number of improved solutions increases with MTCR and the impact of the different neighborhood structures differs with respect to the instance characteristics.

Table 7.5: Impact of the neighborhood structures for VND for problem classes with S-shape routing, $n = 200$, and $C = 45$

$p_{\max}$	MTCR	no. best. sol.	$\mathcal{N}^{\text{bsw2}}$	$\mathcal{N}^{\text{osh1}}$	$\mathcal{N}^{\text{osh2}}$	$\mathcal{N}^{\text{osw1}}$	$\mathcal{N}^{\text{osw2}}$
2	0.6	8.3	14%	63%	14%	5%	4%
2	0.7	77.1	50%	39%	7%	2%	3%
2	0.8	551.3	74%	22%	3%	1%	1%
3	0.6	9.0	24%	64%	6%	6%	1%
3	0.7	88.7	57%	37%	3%	2%	1%
3	0.8	456.9	72%	24%	2%	1%	1%
5	0.6	11.8	20%	70%	2%	7%	1%
5	0.7	73.2	59%	38%	1%	2%	0%
5	0.8	396.0	72%	26%	1%	1%	0%

Table 7.6: Impact of the neighborhood structures for VNS-2 for problem classes with S-shape routing, $n = 200$, and $C = 45$

$p_{\max}$	MTCR	no. sol.	$\mathcal{N}^{\text{osw2}}$	$\mathcal{N}^{\text{osh2}}$	$\mathcal{N}^{\text{bsw2}}$	$\mathcal{N}^{\text{bsh2}}$
2	0.6	4.2	25%	39%	8%	29%
2	0.7	16.4	35%	41%	15%	10%
2	0.8	61.3	43%	23%	30%	5%
3	0.6	5.7	31%	41%	9%	20%
3	0.7	18.8	36%	40%	20%	4%
3	0.8	67.6	39%	23%	32%	5%
5	0.6	8.8	30%	45%	10%	15%
5	0.7	11.5	30%	36%	32%	1%
5	0.8	77.7	37%	27%	32%	4%

8 Conclusion

This paper proposes solution approaches for an order batching and sequencing problem, where batches have to be constructed and scheduled to different order pickers in order to minimize the total tardiness for a given set of orders. The problem is pivotal for the efficient management and control of manual picker-to-parts order picking systems. The paper has shown how different types of neighborhoods can be incorporated in local search based heuristics in order to obtain high-quality solutions to the problem described. In detail, a Variable Neighborhood Descent approach and a Variable Neighborhood Search were introduced.

By means of numerical experiments it was shown that both approaches generate solutions superior to those created by a priority-rule based heuristic. Furthermore, it was shown that a deterministic VND algorithm outperforms a stochastic VNS approach. The VNS approach is only able to generate high quality solutions if the computing time increases. Nevertheless, the implementation of these approaches can improve customer service by delivering to customers on time and by avoiding delays in other stages of production. The algorithms can improve warehouse performance significantly. Since both algorithms require short computation times, both variants appear very suitable for implementation in software systems for practical purposes.

For further research, priority values could be assigned to certain orders that originate from important customers. This would result in minimizing a weighted tardiness. The interaction of order batching and batch sequencing with other related planning issues (layout design, article location, picker routing) has not been considered to date. Thus, it might be interesting to develop an approach which integrates decisions on the article location, order batching and picker routing in order to obtain solutions with a small total tardiness.

References

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

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

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

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

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

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

Elsayed, E., 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, N., van den Berg, J., and van der Hoff, H. (2001). An Order Batching Algorithm for Wave Picking in a Parallel-Aisle Warehouse. *IIE Transactions*, 33(5):385–398.

Hansen, P. and Mladenović, N. (2001). Variable Neighborhood Search: Principles and Applications. *European Journal of Operational Research*, 130(3):449–467.

Hansen, P., Mladenović, N., Brimberg, J., and Moreno-Pérez, J. (2010). Variable Neighborhood Search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, pages 61–86. Springer, New York, 2nd edition.

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., editor, *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*, pages 105–137. Springer, London.

Henn, S. and Schmid, V. (2011). Metaheuristics for Order Batching and Sequencing in Manual Order Picking Systems. Working Paper 11/2011, Faculty of Economics and Management, Otto-von-Guericke University Magdeburg.

Hsieh, L.-F. and Huang, Y.-C. (2011). New Bacht Constructions Heuristics to Optimise the Performance of Order Picking Systems. *International Journal of Production Economics*, 131:618–630.

Kulak, O., Sahin, Y., and Taner, M. (2011). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, to be published.

Mladenović, N. and Hansen, P. (1997). Variable Neighborhood Search. *Computers and Operations Research*, 24(11):1097–1100.

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

Tsai, C.-Y., Liou, J., 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., editors, *Supply Chain Management and Reverse Logistics*, pages 323–347. Springer, Berlin et al.

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.

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