# Variable Neighborhood Search for Order Batching in a Warehouse[1]

**Maria Albareda-Sambola**
Department of Statistics and Operations Research, Technical University of Catalonia, Spain

**Antonio Alonso-Ayuso (corresponding author)**
Department of Statistics and Operations Research, University Rey Juan Carlos, Spain
e-mail: antonio.alonso@urjc.es

**Elisenda Molina**
Department of Statistics, University Carlos III de Madrid, Spain

**Clara Simón de Blas**
Department of Statistics and Operations Research, University Rey Juan Carlos, Spain

**Abstract**

In this paper we address the problem of batching orders in a warehouse, with the objective of minimizing the total travel time. Order batching is an NP-hard optimization problem that is very difficult to solve exactly in practice. Thus, most implemented solutions are based on elementary heuristic methods that perform a relatively limited exploration of the solution space. As an alternative, we propose a heuristic based on variable neighborhood search, where the emphasis is placed on performing an intensive exploration of the most promising regions of the solution space. Simulations are conducted to study the performance of the method with different warehouse configurations, and an exhaustive comparative analysis, which considers all the best known heuristics, is carried out. The results obtained show that the proposed heuristic is competitive and that it provides a suitable method which can be used in practice. Additionally, since the performance of the algorithms depends heavily on factors such as storage policy, routing strategies, or the structure of the orders, we have developed an ANOVA study in order to consider the effect of all the above factors on the different methods tested.

**Keywords:** Warehousing, Order Batching, Metaheuristics, VNS.

## 1 Introduction

In recent years, excess global capacity in most sectors of industry has created an intense competition. As Ghiani *et al.* [9] and Shamlaty [27] point out, the growth of e-commerce and the availability of a wide range of alternatives for most products has given rise to a very demanding type of customer, who expects to be served much faster than in the past. So, logistic activities

[1]Partially supported by the grants MTM2006-14961-C05-01/05 from MCyT, Spain, and CCG06-UC3M/ESP-0767 from CAM-UC3M, Madrid.

have become of significant competitive concern to many companies. In this respect, warehousing is one of the most important activities for many companies in the supply chain. Thus, optimizing activities in a warehouse has become an objective of vital importance. Several policies have been proposed in the literature directed towards the improvement of warehouse operations. Complete surveys can be found in [11, 17]. On a strategic level, warehouse layout design is one of the most important decisions to be made (see [9], among others). From the tactical point of view, the main decision is to decide where each product should be located. Finally, operational policies refer to *order picking*, i.e., to the process of retrieving items from their storage location to retrieve customers' orders.

In this paper we concern ourselves with order picking operations, which can account for up to 65% of the operating costs of a warehouse (see Coyle *et al.* [4]). Therefore, efficient order-picking policies can lead to great reductions in warehouse management costs. In particular, we focus on situations where the order-picking strategy in use is *order batching*, *i.e.*, where orders are grouped into batches which are then collected at the same time so that more efficient routes can be used. In this context, De Koster, Roodbergen and Van Voorden show, in [5], that it is possible to make up to 35% savings in travel time, simply by optimally designing the routes of the order pickers. Moreover, if the two decisions concerning the batching of orders are considered simultaneously, the benefits increase substantially, which provides the motivation for the study of problems where batching and routing are combined. However, obtaining optimal solutions to these problems is very difficult and time consuming, essentially because the implications of assigning a specific order to a batch depend on the other orders that are also assigned to the same batch. The objective considered in this paper is to minimize the total distance traveled in order to collect all the orders. However, alternative objectives have been addressed in the literature. For instance, the minimization of the maximum lead time among the batches has been tackled by Gademann *et al.* in [7]. In their work, the authors propose a branch-and-bound algorithm to solve this problem. However, large instances are difficult to solve. Gademann and van de Velde in [8] propose an enumeration scheme as well as a heuristic algorithm to solve the problem of order batching in a parallel aisle warehouse where the objective is to minimize the total travel time. They prove the NP-hardness of the problem except for the case where no batch contains more than two orders, which can be solved in polynomial time. The authors propose a branch-and-price algorithm based on a generalised set partition formulation of the problem that allows to solve moderate size instances to optimality. Obtaining the optimal solution is time consuming for large instances and the authors have noted that for practical purposes a truncated branch-and-bound algorithm might be a better option.

In this paper we propose a new metaheuristic to build the order batches, designed to favor efficient order picking. Our proposal is tested and compared to previously existing order batching heuristics. The results obtained show that the method we propose is either comparable to, or

even better than the previously published heuristic methods, in terms of solution quality. The remainder of this paper is organized as follows. Section 2 describes the problem. Section 3 reviews the existing order batching methods in the literature and describes the existing heuristic methods with which our method is compared. In Section 4 the new metaheuristic for order batching is presented. Computational results obtained on a testbed of randomly generated instances are presented in Section 5, which includes an exhaustive comparative analysis of all the batching methods considered in the paper. Additionally, since the performance of the algorithms depends heavily on factors such as storage policy, routing strategies, or the structure of the orders, an Analysis of Variance study has been developed in order to consider the effect of all the above factors on the different methods tested. Finally, Section 6 summarizes the paper and outlines the conclusions of this work.

## 2 Problem description

Several orders are received in a warehouse, on a daily basis. Each order consists of a number of order lines. Each order line is the code of an item that has to be delivered to the customer. In an order picking oriented policy, all the items in the same order are picked by the same picker, to preserve the order integrity. In this situation, two basic order-picking strategies can be considered: *strict-order picking*, and *order batching*. In strict-order picking, each picker works on one order at a time, and only when all items included in an order have been retrieved, a new order is assigned to the picker. In order batching, it is assumed that several orders are known before batching decisions are made so that the requirements of different orders can be aggregated. Under such assumptions, a picker can work with a batch of various orders simultaneously and, using an appropriate pick-device, items can be sorted by order during the picking process. All the items that make up a particular order are picked in the same batch. Thus, no consolidation is needed at the end of the route. Moreover, since several orders are picked simultaneously, efficient routes can be devised and the travel times can be reduced, as compared to the strict-order policy.

As mentioned above, this paper is concerned with warehouse management at an operational level. Thus, decisions regarding the layout of the warehouse and the location of the different items are considered fixed throughout the process. In particular, we consider the order batching problem for a warehouse or distribution center with parallel aisles of equal length, two possibilities for entering/exiting the aisles, at the front and back of each aisle, and a depot. The depot, which is referred to as the I/O point (input/output), is the point where pickers start their routes and where they deposit the items at the end of each route. The depot can be located either in one of the corners or in the center of one of the cross aisles. As an example, Figure 1 shows a warehouse with 7 parallel aisles. Additionally, it is assumed that each picker has a load-capacity and the total load for any given order is below this limit. If not, the order has to be split into

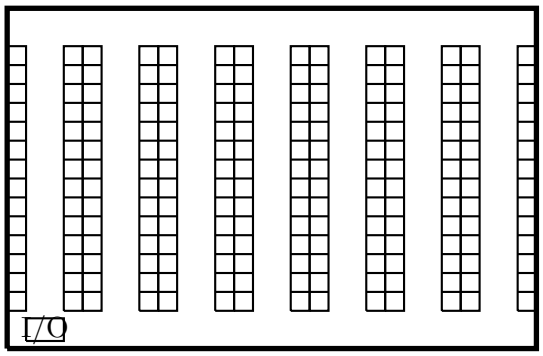



Figure 1: Warehouse layout

two or more smaller orders, so that each smaller order has a load that does not exceed the picking load-capacity. To be more exact, we deal with the problem of distributing the orders among batches, each with a total load below the load-capacity of a picker, and determining a batch picking route for each batch, in order to minimize the total travel time. So, the problem has two natural components, clustering of orders into feasible batches and actual batch routing, which consists in deciding the sequence used by each picker to retrieve the items in his batch so as to minimize the total picking time. If we assume that the orders are already divided into batches, then the route design for each batch is a problem that has been studied extensively and is polynomially solvable for the types of warehouses considered in this work [23].

## 3 Order Batching methods: A literature review

Batching problems are NP-hard, as proved in [8], and realistically sized instances cannot, in general, be solved in a reasonable amount of time. However, a few papers have proposed exact algorithms for different variants, where alternative objectives have been considered (see [11]). Armstrong *et al.* in [1] present a mixed-integer programming formulation for batching in a semi-automated order-picking system with the objective of minimizing the total picking time. The problem is solved using Benders' decomposition. Gademann *et al.* in [7] consider the order batching problem with the objective of minimizing the maximum lead time for the batches, and they solve the problem optimally using a branch-and-bound algorithm. In a more recent paper, [8], Gademann and van de Velde consider the total order picking time as the objective to minimize, and they model the problem as a generalized set partitioning problem, which they solve using a branch-and-price algorithm. In all cases, only small and some moderately sized instances (warehouses with 300 picking points and up to 40 orders) can be solved in a reasonable amount of time (less than 30 minutes).

In real life applications, large instances of such problems need to be solved quite quickly. Thus, warehouse managers need to resort to heuristic methods to assist in their batching deci-

sions. This fact has given rise to extensive research on heuristic methods for batching problems, as can be seen in the early survey by Van den Berg [28] and the more recent studies by Gu *et al.*, [11], and Koster *et al.*, [17]. Some of these works perform comparative studies of different heuristics, as Rosenwein in [26] and Koster *et al.* in [16], where the authors perform the most exhaustive comparative study published to date of different heuristics that are applied to manual picking systems.

According to Koster *et al.* in [16], most order batching heuristics can be classified into one of three categories: *Basic methods*, *Seed methods* and *Savings methods*.

**Basic methods** are simple constructive methods, based on different greedy rules. For instance, under the First-Come-First-Served (FCFS) strategy, orders are added to batches in the sequence they arrive. When an order cannot be added to the current batch, a new batch is added to the list. These are the most commonly implemented methods in practice because they are the easiest to implement, although they are not very effective (see Section 5).

**Seed methods** ([10, 14, 16, 20, 29]) build one batch at a time, following two steps. First, a *seed order* is selected from the set of orders not yet selected, which will serve as the reference order for the new batch. In a second step, *order addition*, the batch is enlarged by sequentially adding new orders to it, as long as the picker capacity is not exceeded. New orders are selected using some measure of *proximity* to the seed order. Different seed methods can be defined by choosing different rules for each of the two steps, namely, seed selection and proximity criterion (see [14, 16]). Moreover, all seed selection rules can be applied either in *single* mode or in *cumulative* mode. In *single* mode, the seed is selected once only, whereas in the *cumulative* mode the seed order is updated each time an order is added to the batch. For a detailed description of the most relevant seed methods, the reader is referred to [14, 16]. A complete review can be found in [29].

**Savings methods** ([16, 26]) consist in, starting with one batch per order, sequentially reducing the number of batches by combining two at a time. The criterion used to select the batches to be merged is the cost saving achieved by doing this. These methods are inspired by one of the best known heuristics for the Vehicle Routing Problem: The Clarke and Wright savings algorithm [3]. This method is usually applied to problems where the number of vehicles is modeled as a decision variable, and it works equally well for both directed and undirected problems. Let $t_B$ be the time spent collecting orders in batch $B$. A savings matrix can be defined, where element $(p, q)$ is the saving attained by merging batches $B_p$ and $B_q$ into a single unit, and is computed as $saving(B_p, B_q) = t_{B_p} + t_{B_q} - t_{B_p \cup B_q}$. If this value is positive and the picker capacity is not exceeded, then the batches $B_p$ and $B_q$ are merged into a single batch. Two variants of this algorithm have been proposed in the literature. They differ in the information they manage throughout the process and in the type of merges that are considered. The first variant, Clarke and Wright I, uses the original savings matrix during the whole process. In this

variant, only merges where at least one batch is formed by a single order are considered. In the second variant, Clarke and Wright II, the savings matrix is updated at each iteration and all pairs of batches are considered for merging. This variant is more complex than Clarke and Wright I. However, several experiments show that this variant usually outperforms the previous one (see Section 5 and [16]). It requires more computing time for the updating of the savings matrix but a few seconds are still enough to solve problems with several hundred orders.

As mentioned before, the above classification includes most of the batching algorithms considered in the literature. However, some algorithms have been developed that do not fit into any of the three categories. For instance, Elsayed and Unal in [6] developed several methods based on time savings which are a combination of savings and seed methods. In this work, we focus on the *EQUAL algorithm*, which is a seed algorithm that uses time savings as a proximity measure. To be specific, the algorithm selects the pair of orders with largest time saving as the seed for building a batch and then, the batch is enlarged by adding one order at a time, using the time saved as a proximity measure. That is, as a seed method, it is performed in a *cumulative mode*. Moreover, its seed order addition rule coincides with the *Largest Time Saving (LTS) Rule*, used in [16]. The order with the largest potential saving in travel time is added to the batch and the process is repeated until the picker is loaded to capacity. Then, a new pair of orders is selected as the seed of a new batch and the procedure is repeated. When, at the end of the procedure, there are some orders left over, they are put into individual batches.

More recently proposed alternatives incorporate innovative ideas in batching that are distinct. For example, Chen and Wu, in [2] present a batching approach based on data mining and integer programming, and Hsu, Chen and Chen, in [15], propose a genetic algorithm for a general version of the problem.

In any case, one of the features that contributes most to the difficulty of the order batching and routing problem is that even slight modifications in the batches lead to dramatic changes in the total time necessary to collect them. For instance, reassigning one order from one batch to another implies changing several references (picking sites) from one route to another that, in general, will have no geographic pattern in common. Moreover, merging two batches into one can have opposite effects. On the one hand, even when the uncommon references among the batches to be merged are only a few, merging them can result in a very costly route, and a considerable increase on the total time required to collect all the items. For instance, this happens when the uncommon references are distant from each other, which is quite frequent in the case of low demand items. On the other hand, it often occurs that merging two batches significantly reduces the total collection cost, even if they have no references in common. This is the case when the references in the batches, despite being different, are located along the same set of aisles.

# 4 A Variable Neighborhood Search Metaheuristic

The categories described in the previous section correspond to *pure* constructive methods: once two orders are assigned to the same batch, they cannot be separated by the procedure. That is, they do not incorporate any improvement step. We propose a new method based on a local search that incorporates reoptimization at intermediate steps so that increments the probability of exploring promising regions of the solution space in the current neighborhood. Variable Neighborhood Search (VNS) is a recent metaheuristic (see [18, 19]) for solving combinatorial and global optimization problems based on the idea of systematically changing the considered neighborhood to help escape from local optima. The reader is referred to [13] for a VNS survey where many applications to industry are briefly summarized. To be specific, we consider three neighborhoods, which are described below, and we have implemented a variable neighborhood descent method in which a change of neighborhoods is performed in a deterministic and sequential way. Our results, which evaluate the performance of the algorithm under three different routing strategies[2], show that the proposed method is competitive compared to heuristic methods which already exist. Moreover, those results show that even the pure local search within the first neighborhood is competitive. We next introduce our notation and definitions concerning the VNS metaheuristic:

$C$, picker load-capacity (cart-capacity).

$\mathcal{O}$, set of orders.

$p_o$, total load of items included in order $o \in \mathcal{O}$.

$p_B$, total load of orders included in batch $B \subset \mathcal{O}$.

Let $\mathcal{B}$ be a *feasible* partition of the order set $\mathcal{O}$ in batches, i.e., $p_B \leq C$, $\forall B \in \mathcal{B}$. Then, the set of neighborhood structures we use in the VNS are defined as follows, where $\mathcal{N}_k(\mathcal{B})$ denotes the set of solutions in the $k$-th neighborhood of $\mathcal{B}$, $k = 1, 2, 3$:

$\mathcal{N}_1(\mathcal{B})$ is the set of feasible partitions reachable from $\mathcal{B}$ by transferring an order from one batch to another one.

$\mathcal{N}_2(\mathcal{B})$ is the set of feasible partitions reachable from $\mathcal{B}$ by transferring at most two orders from one batch to others.

$\mathcal{N}_3(\mathcal{B})$ is the set of feasible partitions reachable from $\mathcal{B}$ by transferring at most two orders.

[2]See section 5.1 for a detailed description of those strategies

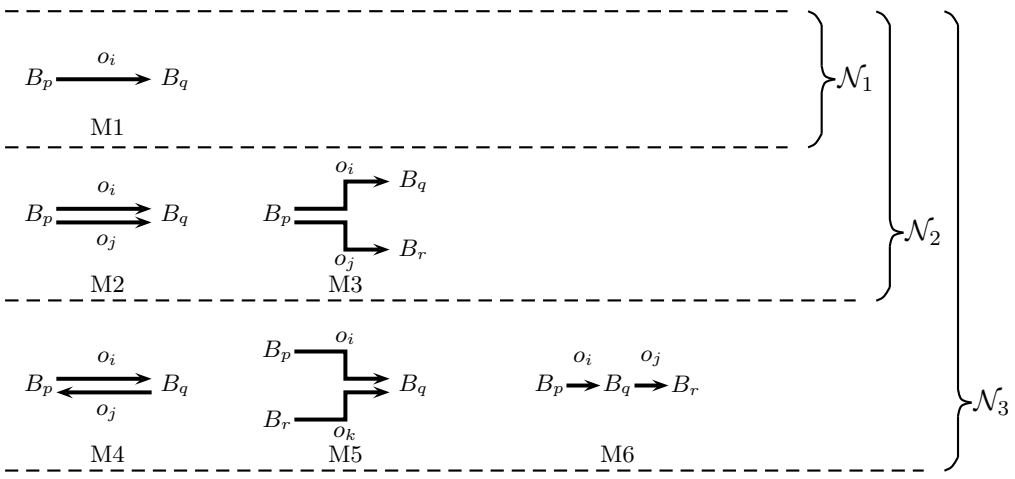


Figure 2: Movements considered in the Local Search

Note that the set of neighborhood structures we define form a nested set. Figure 2 shows the different movements that are considered. Movement M1 generates neighborhood $\mathcal{N}_1$, movements M1, M2 and M3 generate $\mathcal{N}_2$, whereas movements M1 to M6 generate $\mathcal{N}_3$.

A feasible partition $\mathcal{B}$ is a *local minimum with respect to* $\mathcal{N}_k$ if no member of neighborhood $\mathcal{N}_k(\mathcal{B})$ has a better objective value, i.e., a smaller travel time.

Additionally, an empty batch is kept along the algorithm to allow for the removal from a batch of an order that has been previously assigned to it, when this improves the objective function value.

**Variable neighborhood search metaheuristic**

**Step 0:** Set the initial solution $\mathcal{B}$ as the feasible partition which includes one batch per order plus an *empty* batch.

**Step 1:** *Exploration of neighborhood* $\mathcal{N}_1$: Find the best neighbor $\mathcal{B}' \in \mathcal{N}_1(\mathcal{B})$.
*Evaluate movement*: If the partition thus obtained $\mathcal{B}'$ is better than $\mathcal{B}$, set $\mathcal{B} \leftarrow \mathcal{B}'$ and repeat Step 1; otherwise, GO TO Step 2.

**Step 2:** *Merging batches:* For all $B_p, B_q \in \mathcal{B} : p_{B_p} + p_{B_q} \leq C$, compute $saving(p,q) = t(B_p) + t(B_q) - t(B_p \cup B_q)$, where $t(B)$ is the length (measured in travel time) of route on batch $B$ constructed according to the selected routing strategy.

If $saving(p,q) \leq 0$, $\forall B_p, B_q \in \mathcal{B} : p_{B_p} + p_{B_q} \leq C$, GO TO Step 3, no batch is merged.

Otherwise, create a new batch $B_* = B_{p^*} \cup B_{q^*}$, where $(p^*, q^*) = \arg \max_{\substack{B_p, B_q \in \mathcal{B} \\ p_{B_p} + p_{B_q} \leq C}} \{saving(p,q)\}$, update $\mathcal{B} \leftarrow \mathcal{B} \setminus \{B_{p^*}, B_{q^*}\} \cup \{B_*\}$, and REPEAT Step 2.

**Step 3:** Set $k \leftarrow 1$ and repeat the following steps as long as $k \leq 3$:
*Exploration of neighborhood* $\mathcal{N}_k$: Find the best neighbor $\mathcal{B}' \in \mathcal{N}_k(\mathcal{B})$.

*Evaluate movement*: If the new solution $\mathcal{B}'$ is better than $\mathcal{B}$, set $\mathcal{B} \leftarrow \mathcal{B}'$ and $k \leftarrow 2$; otherwise, set $k \leftarrow k+1$.

Step 2, which is a merging step, has been introduced in order to avoid having batches which are too small in the local optimal solution with respect to $\mathcal{N}_1$, which is used as the initial solution for the variable neighborhood search. It should be noted that it is not computationally costly since, in that solution, there are not many pairs of batches $B_p, B_q$ satisfying the capacity constraint $p_{B_p} + p_{B_q} \leq C$.

Each time a neighborhood of the current solution is explored (Steps 1 and 3), it is necessary to evaluate the reassignment of one or two orders to different batches, which requires the re-computation of the associated routes with the corresponding algorithm. Due to the special structure of the route time function associated with a given batch, this operation is computationally expensive. Since neighborhoods exploration in Steps 1 and 3 are made by explicit enumeration, it is crucial to devote additional effort to reduce this computational cost. A first step in this direction is taken by keeping, at each iteration, the value of the variables $\Delta^+_{o_i B_p}$ and $\Delta^-_{o_i B_p}$, $\forall o_i \in \mathcal{O}, B_p \in \mathcal{B}$. These variables are defined as follows: $\Delta^+_{o_i B_p}$ is the increase of the length of the route associated with batch $B_p$ if order $o_i$ is added to it ($\Delta^+_{o_i B_p} = 0$ if $o_i \in B_p$), and $\Delta^-_{o_i B_p}$ is the reduction of the length of the route of batch $B_p$ that is attained if order $o_i$ is unassigned from $B_p$ ($\Delta^-_{o_i B_p} = 0$ if $o_i \notin B_p$). These variables allow us to reduce the number of computations needed for evaluate the savings of each possible movement at each iteration. For instance, let us consider the movements represented in Figure 2. The variation in the value of the objective function of the current solution when movement M1 is applied, can be obtained directly as $\Delta^-_{o_i B_p} + \Delta^+_{o_i B_q}$. Moreover, the analysis of the combination of movements allows us to reduce the computations even more. For instance, in order to evaluate movement M2 it is necessary to obtain the length of the new routes $t^{\text{new}}_{B_p}$ and $t^{\text{new}}_{B_q}$ of batches $B_p$ and $B_q$ once M2 has been applied, since the variation on the value of the objective function is $t^{\text{new}}_{B_p} + t^{\text{new}}_{B_q} - t_{B_p} - t_{B_q}$. However, the variation on the value of the objective function when movement M3 is applied can be obtained as $t^{\text{new}}_{B_p} - t_{B_p} + \Delta^+_{o_i B_q} + \Delta^+_{o_j B_r}$; so, no additional routes need to be computed. Figure 3 shows the algorithm we have implemented to calculate the savings of each possible movement. Once these savings are calculated, the best movement is applied and $\Delta$-variables are updated.

# 5 Computational experiments

## 5.1 Test cases description

To show the effectiveness of the proposed order batching metaheuristic, we have performed a series of computational experiments over an extensive set of test cases where we have compared

M1:

$\forall B_p \in \mathcal{B}, \forall o_i \in B_p, \forall B_q \in \mathcal{B} : p \neq q, p_{B_q} + p_{o_i} \leq C$:

$\text{saving}_{M1}(o_i \to B_q) = \Delta^-_{o_i B_p} + \Delta^+_{o_i B_q}$.

M2 and M3:

$\forall B_p \in \mathcal{B}, \forall o_i, o_j \in B_p : i < j$:

Calculate $t^{\text{new}}_{B_p}$, length of route for batch $B_p \setminus \{o_i, o_j\}$.

$\forall B_q \in \mathcal{B} : p \neq q, p_{B_q} + p_{o_i} \leq C$:

If $p_{B_q} + p_{o_i} + p_{o_j} \leq C$:

Calculate $t^{\text{new}}_{B_q}$, length of route for batch $B_q \cup \{o_i, o_j\}$,

$\text{saving}_{M2}(o_i \to B_q, o_j \to B_q) = t^{\text{new}}_{B_p} + t^{\text{new}}_{B_q} - t_{B_p} - t_{B_q}$.

$\forall B_r \in \mathcal{B} : r \neq p, r \neq q, p_{B_r} + p_{o_j} \leq C$:

$\text{saving}_{M3}(o_i \to B_q, o_j \to B_r) = t^{\text{new}}_{B_p} - t_{B_p} + \Delta^+_{o_i B_q} + \Delta^+_{o_j B_r}$.

M4, M5, and M6:

$\forall B_p \in \mathcal{B}, \forall o_i \in B_p, \forall B_q \in \mathcal{B} : p \neq q$:

$\forall o_j \in B_q$: $p_{B_q} + p_{o_i} - p_{o_j} \leq C$:

Calculate $t^{\text{new}}_{B_q}$, length of route for batch $B_q \setminus \{o_j\} \cup \{o_i\}$,

If $q > p$ and $p_{B_p} - p_{o_i} + p_{o_j} \leq C$:

Calculate $t^{\text{new}}_{B_p}$, length of route for batch $B_p \setminus \{o_i\} \cup \{o_j\}$,

$\text{saving}_{M4}(o_i \to B_q, o_j \to B_p) = t^{\text{new}}_{B_p} + t^{\text{new}}_{B_q} - t_{B_p} - t_{B_q}$.

$\forall B_r \in \mathcal{B} : r \neq p, r \neq q, p_{B_r} + p_{o_j} \leq C$:

$\text{saving}_{M6}(o_i \to B_q, o_j \to B_r) = t^{\text{new}}_{B_q} - t_{B_q} + \Delta^-_{o_i B_p} + \Delta^+_{o_j B_r}$.

$\forall B_r \in \mathcal{B} : r > p, r \neq q$: $\forall o_k \in B_r : p_{B_q} + p_{o_i} + p_{o_k} \leq C$:

Calculate $t^{\text{new}}_{B_q}$, length of route for batch $B_q \cup \{o_i, o_k\}$,

$\text{saving}_{M5}(o_i \to B_q, o_k \to B_q) = t^{\text{new}}_{B_q} - t_{B_q} + \Delta^-_{o_i B_p} + \Delta^-_{o_k B_r}$.

Figure 3: Calculation of savings of each possible movement

our algorithm to previously existing methods. Furthermore, the comprehensive analysis carried out allows us to develop an extensive comparative study for order batching heuristics.

Four different warehouses have been considered that were taken from the literature. Table 1 shows their main characteristics. Warehouses 1, 2 and 3 are used in the computational results presented in [16], whereas warehouse 4 is the example used in [14]. The first warehouse is a narrow-aisle pallet warehouse where manned combi-trucks or cranes are used. The second warehouse is a shelf store with manual order picking using pick carts. As referred to in [16], the third warehouse is included for reference and it is based on a pallet warehouse as far as the dimensions are concerned. However, it is a non-existent, hypothetical warehouse, with regard to the equipment that can be used in such an environment, the picker's capacity, the aisle lengths combined with the number of aisles and the size of the orders to be picked. Warehouse 4 is used for very large-dimensioned orders. The following additional assumptions are made:

| | Warehouse | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| Order size | U(1,7) | U(2,10) | U(5,25) | U(1,36) |
| Item weight | 1 | 1 | 1 | U(1,3) |
| Capacity order picker (weight) | 12 | 24 | 150 | 80 |
| Number of aisles | 4 | 10 | 25 | 12 |
| Number of items per aisle | $2 \times 30$ | $2 \times 20$ | $2 \times 25$ | $2 \times 16$ |
| Total number of items | 240 | 400 | 1250 | 384 |
| Aisle length | 50m | 10m | 50m | 80m |
| Centre distance between two aisles | 4.3m | 2.4m | 5m | 15m |
| Travel speed within aisles | 1.5m/s | 0.6m/s | 2m/s | 1m/s |
| Travel speed outside aisles | 1m/s | 0.6m/s | 1m/s | 1m/s |
| Additional time to enter or leave an aisle | 15s | 0s | 20s | 0s |

Table 1: Warehouse characteristics

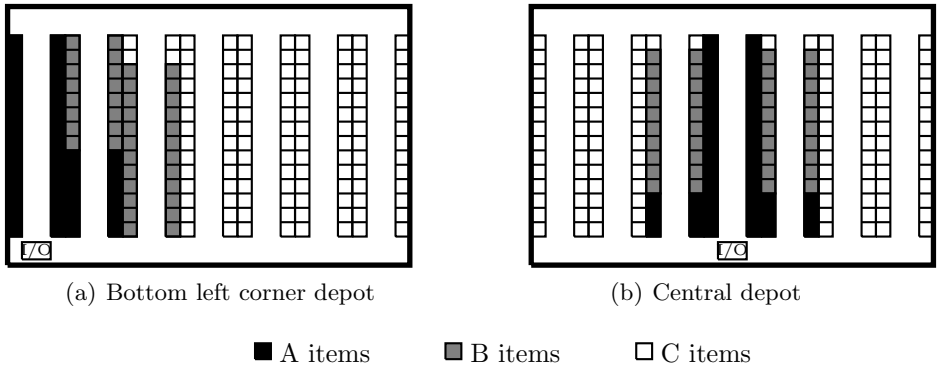
- All order information is known beforehand.
- Order splitting is not permitted; therefore, we assume the maximum order size does not exceed the cart capacity. Any order that does violate this assumption may always be split in advance.
- The time required for vertical movement of pickers can be disregarded. Note that since all items have to be retrieved and since the time required to retrieve an item is independent of what other items have been retrieved, the time required to retrieve items from the aisle can also be disregarded. We also assume that the time for a picker to rotate, without changing its location, so that it may retrieve items from an opposing aisle, is negligible.
- Order sizes follow a discrete uniform distribution (see Table 1).
- Pickers are able to traverse any aisle in both directions.
- Demands fit into an ABC scheme. Let us consider an ordered list of all items in the warehouse in which the most-demanded item is the first in the list and the least-demanded item is the last in the list. The first quintile of most demanded items (A items) concentrate 80% of demand, the second quintile of most demanded items (B items) concentrate 10% of demand while the last three quintiles of the items (C items) concentrate only 10% of the total demand. Note that, under such assumption, the tactical decisions concerning the distribution of goods in the warehouse have a very important effect on the final picking costs.

| | Warehouse | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| Orders per batch | 3 | 4 | 10 | 2.16 |
| Items per batch | 12 | 24 | 150 | 40 |
| Items per batch and aisle (random) | 3 | 2.4 | 6 | 3.33 |
| Travel time to traverse an aisle | 63.33 | 16.67 | 65 | 80 |
| Travel time to go from first to last aisle | 12.9 | 36 | 120 | 165 |

Table 2: Warehouse characteristics

Before reporting the computational results it is of interest to analyze the main differences between the four warehouses under study. Table 2 presents some operational features of these warehouses. From the cart capacities, the average order size and the average weight of the items, it is easy to estimate the expected number of orders per batch, if we assume that pickers will carry loads which approximate their capacities. Similarly, the number of items per cart can also be estimated. Bear in mind that each item represents a collection point on the route of the picker. In this sense, it is worth noting that, in Warehouse 3 batches will be formed with more orders than in the others. That is, for the same amount of orders, in Warehouse 3 we expect to obtain fewer routes, but with more collection points. Thus, it is reasonable to expect that solutions in this warehouse will be the most sensitive to the strategy used to devise the routes for each batch. On the other hand, if we consider that the items are uniformly distributed within the warehouse, the expected number of items from a given aisle that each batch contains ranges from 2.4 (Warehouse 2) to 6 (Warehouse 3). This value has a significant impact on the performance of the considered routing strategies. Scenarios with large values of this ratio are especially well suited for algorithms that devise routes which cross aisles from end to end as opposed to those that devise routes that enter and leave the aisles from the same end point. Finally, Table 2 also shows the time that each picker needs to go through an aisle (including the additional time to enter or leave it) and the time required to go from the first to the last aisle. These values allow us to estimate roughly the impact of the movements between corridors on the total travel time. In this sense, Warehouse 1 is particularly well laid out, since the time required to move between aisles is very small as compared with the time to pass along an aisle. For this reason, in this warehouse, the starting point of the routes will be less relevant than in the others.

For our experiment, we have generated instances consisting of order lists of 5 different sizes: 50, 100, 150, 200 and 250 orders. For each list size, we generated 120 instances for each warehouse. Thus, a total of 2400 instances have been generated. Different settings for the following three factors have been combined: depot location, storage policy and routing strategy. Depot location and storage policy refer to the warehouse organization, whereas the routing strategy




(a) Bottom left corner depot (b) Central depot

Figure 4: Demand-based storage policy

is a part of the order-picking process. We list the different options considered for each of these three factors below.

**Depot location.** Central and bottom left corner (aisle 0). See Figure 4 for an example.

**Storage policy.** Demand-based and random. In the first case, aisles close to the depot contain A items, whereas, aisles far from the depot contain C items. See Figure 4 for an example.

**Routing strategy.** Batch routing consists in deciding the sequence used by each picker to retrieve the items in his batch so as to minimize the total picking time. This problem classifies as a Steiner Traveling Salesman Problem (see [17]) embedded in a special kind metric space, which possesses properties that can be exploited to develop powerful heuristics. In fact, Ratliff and Rosenthal [23] propose an optimal procedure for routing in a rectangular warehouse based on dynamic programming, which requires a linear number of computations with respect to the number of aisles for warehouses with no cross aisles. However, the efficiency decreases when the warehouse has cross aisles. Furthermore, the proposed routes may seem illogical to the pickers who, then, as a result, deviate from the specified routes (Gademann and van de Velde [8]). As Petersen remarks in [22]: "heuristics are easy to understand and form routes which are fairly consistent in nature. Such consistency helps to minimize the risk of a missed pick, a much greater sin than having to walk a few extra steps". Thus, although it is possible to implement optimal routing algorithms efficiently, simple heuristically designed routes are often preferred in practice. Different heuristics have been proposed in the literature (see Hall [12], Petersen [22] and Roodbergen and Petersen [25]). From these, we have used three different routing heuristic strategies in our tests: *traversal*, *largest gap* and *combined*. Petersen in [22] carried out a number of numerical experiments to compare six routing methods. He concludes that the best heuristic

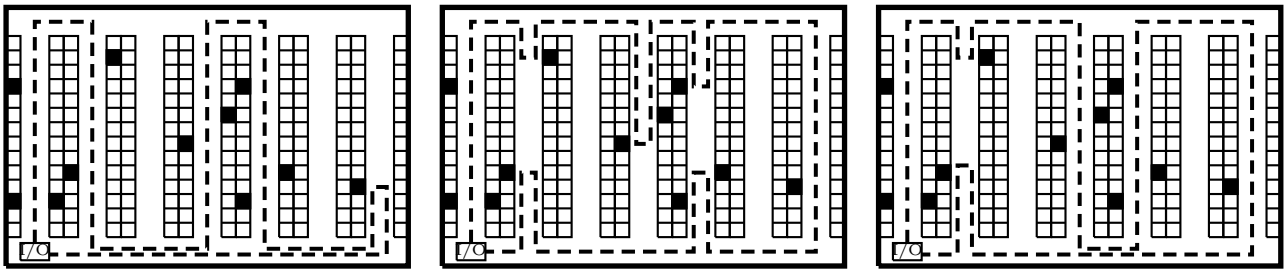
procedures, *composite* (Petersen, [21, 22]) and *largest gap*, were 9 to 10 percent above optimal. Overall, the best heuristic solution was 5 percent above the optimal solution.

The simplest of these heuristics is the *traversal* strategy (also called *S-shape* strategy). Following this strategy, picker enters every aisle that contains at least one required item, travels the length of the entire aisle while picking the required items, and exits the aisle at the end opposite to which the picker entered. An exception to this last point is made for the last aisle, in the case where the number of visited aisles is odd. In such a case, the picker visits the last aisle only to the farthest reference from the aisle entrance and exits through the same aisle entrance. From the last visited aisle, the picker returns to the depot. See Figure 5(a) for an example. This method is widely used in practice because, even though it is very simple, it provides very efficient routes, especially when the picker cannot easily change directions within an aisle or when considerable time is needed for changing aisles.

An alternative method is the *largest gap* strategy. In this case, each visited aisle, except the extreme ones, is exited through the same side from which it was entered. The picker goes from the depot to the first aisle with required items, traverses the aisle and continues along the back of the aisles to the last aisle with required items, traverses the aisle and returns to the depot. During this route, the picker visits all intermediate aisles with required items: if the picker needs to enter an aisle, gaps (measured in travel time) between each pair of adjacent points (picking points in the aisle or aisle end points) are measured. If the largest of such gaps is between two adjacent picking points, the picker performs routes from both ends of the aisle. Otherwise, the aisle is entered only once: a return route from the back of the aisle (if the largest gap is between the front end point and its nearest picking point), or from the front of the aisle (if the largest gap is between the back end point and its nearest picking point) is used. Note that the largest gap within an aisle is thus the portion of the aisle that is not traversed. See Figure 5(b) for an example. For problems with low density of items per aisle, *largest gap* can outperform the *traversal* strategy.

Finally, Roodbergen and Koster [24] proposed the *combined* strategy, that is fairly similar to the *composite* strategy. This method is a combination of the *largest gap* and the *traversal* strategy, which means that an aisle is either fully traversed or entered and left from the same side of the aisle. This method has a minor dynamic programming component, since at each aisle it looks one aisle ahead to select the strategy (traverse the aisle or enter and exit from the same side of the aisle), and the option to choose is selected taking into account which one gives the shortest route when combined with the best option for the next aisle. See Figure 5(c) for an example. A complete description of the *combined* routing strategy can be found in [24].

By combining different levels of the three above factors, $2 \times 2 \times 3 = 12$ different scenarios have been considered. For each scenario, 10 replicas have been generated. Therefore, at least




(a) Traversal (b) Largest Gap (c) Combined

Figure 5: Routing strategies

40 replicas per factor level have been considered.

### 5.2 Computational results

Four measures are reported to compare the travel times obtained with the different methods:

- the *average* of the travel time over the considered instances,
- the *average of the deviation from the best*, where the deviation from the best is defined as the percentage deviation of the solution found from the best solution obtained with all methods. For method $j_0$, its deviation from the best is calculated as

$$\frac{d_{ij_0} - \min_{j \in \mathcal{J}} d_{ij}}{\min_{j \in \mathcal{J}} d_{ij}} \times 100$$

  where $d_{ij}$ denotes the travel time for instance $i$ when method $j$ is applied, and $\mathcal{J}$ is the set of methods applied to solve the problem.
- the *maximum of the deviation from the best*.
- the *number of best solutions*, defined as the number of instances for which a given method provides the best solution.

These four measures have been recorded for 192 seed-based methods (12 seed-selection rules multiplied by 16 addition rules, implemented in a cumulative mode), 3 savings algorithms (*Clarke and Wright I*, *Clarke and Wright II* and *EQUAL*), and our proposal, *VNS*. The *First-Come-First-Served* (FCFS) strategy is added as a reference. *Average seed* is the average value of all obtained solutions by all seed methods whereas *best seed* represents the seed method that attains the best average over all problem instances.

Table 3 shows the main results for all instances. The first row (VNS) refers to the final solution obtained at the end of our algorithm, the second row (1-opt. sol.) refers to the locally optimal solution with respect to $\mathcal{N}_1$ to which we apply the variable neighborhood search[3], whereas the following rows correspond to the algorithms taken from the literature.

| | Average | %dev. from the best | | # of best |
|---|---|---|---|---|
| | objetive value | Average | Maximum | solutions |
| VNS | 28530.24 | 0.67 | 7.47 | 1338 |
| 1-opt. sol. | 28918.83 | 2.26 | 13.34 | 101 |
| C & W II | 29009.16 | 2.89 | 22.27 | 294 |
| C & W I | 29496.87 | 5.79 | 43.82 | 104 |
| Equal | 29141.48 | 3.74 | 23.74 | 109 |
| Best seed | 29048.67 | 3.97 | 18.42 | 99 |
| Average seed | 31849.19 | | | |
| FCFS | 36436.76 | 37.28 | 76.95 | 0 |

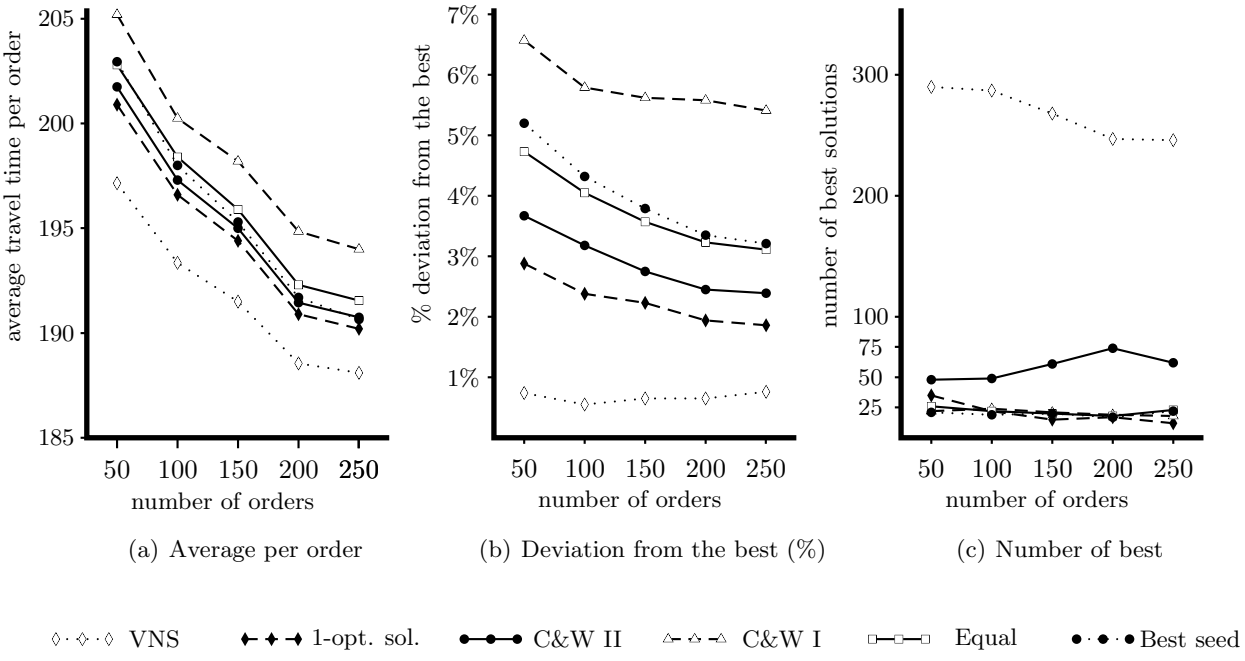
Table 3: Main Global Results

As can be seen in the table, our VNS algorithm outperforms all existing algorithms according to all four quality measures considered. Indeed, it gives the smallest average travel time and is the only method with a percentage deviation from the best known solution under 1%. Moreover, it provides the best solution in almost 55% of the instances[4] and for the rest it provides solutions that are less than 7.5% above the best, whereas these figures are under 12% and above 22%, respectively, for all the other methods. As will be seen later in the detailed analysis per warehouse, this behavior is quite uniform and results are similar to the average case in most warehouses. On the other hand, it is also clear from these results that seed methods are not competitive in general. However, as we will see in Table 5, they give competitive results for Warehouse 4.

Results in Table 3 aggregate data from different warehouses and different order list sizes. Next, we study how the number of orders and the warehouse affect the performance of the algorithms. To this end, Figure 6 shows the previous results disaggregated per number of orders; Figure 6(a) represents the average travel time per order (defined as the total average travel time divided by the number of orders), and Figures 6(b) and 6(c) represent the average of the deviation from the best and the number of best solutions. It can be seen that the VNS algorithm always provides the best average and its solutions are the closest to the best solution. In fact, as the number of orders increases, the travel time per order tends to decrease (a reduction

[3]That is, the solution obtained at the end of Step 2 of our algorithm, see page 8.

[4]The sum of the "# of best solutions" column is strictly less than the total number of instances because all those instances for which the best solution was found with a seed method different from the overall best seed (Best seed row) are not included in the table.




(a) Average per order (b) Deviation from the best (%) (c) Number of best

Figure 6: Main results for different instance sizes

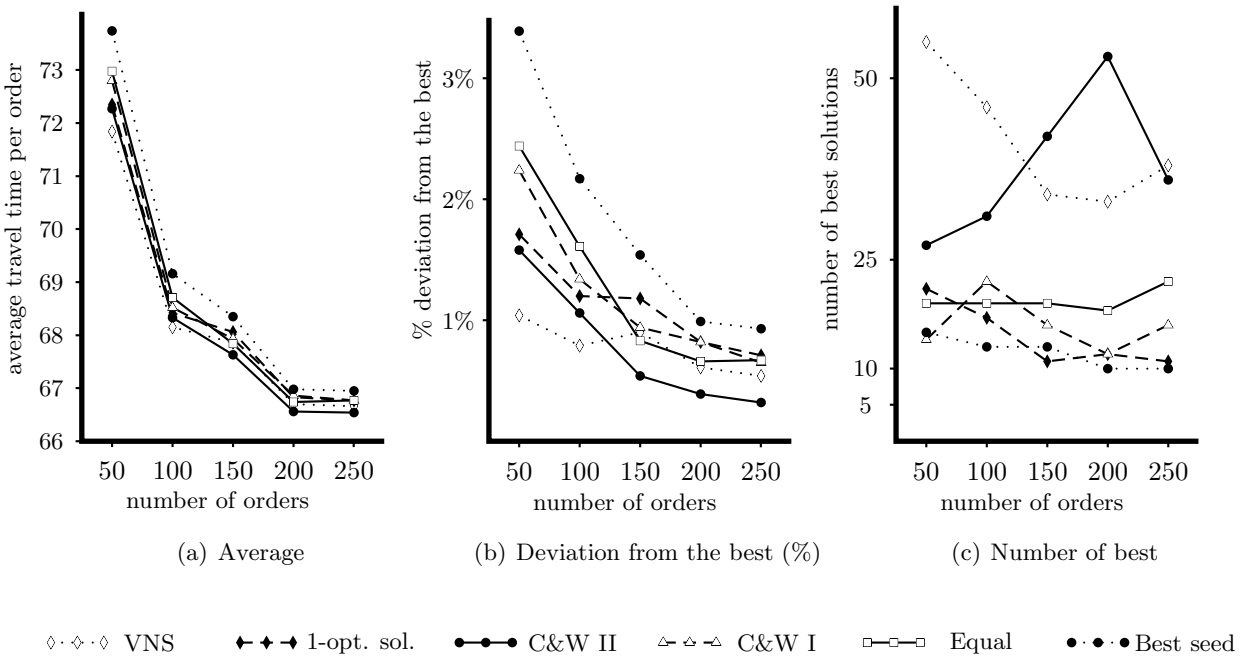
of 6% is obtained from 50-orders instances to 250-orders instances). If we count the number of times that a method proposes the best solutions, VNS is clearly the best. Note also that Steps 0 to 2 of the VNS algorithm (1-opt. sol.) is the second best approach in terms of average travel time per order and in terms of deviation from the best known solution. On the other hand, since four warehouses are used, Table 3 aggregates information from very different instances, and some differences are found when studying results by warehouse. Thus, in order to get more insight into the performance of the different algorithms depending on the warehouses, we report the partial results for each warehouse.

**Warehouse 1**

Table 4 shows the main statistics for Warehouse 1. From the results obtained in this warehouse, it follows that the best results are obtained by either VNS or Clarke and Wright II. The other algorithms behave similarly and the obtained results do not give reliable clues regarding whether to choose one from the others. Figure 7 shows the results for the different instance sizes that have been considered.

| | Average objetive value | %dev. from the best Average | %dev. from the best Maximum | # of best solutions |
|---|---|---|---|---|
| VNS | 10118.18 | 0.78 | 5.33 | 206 |
| 1-opt. sol. | 10145.50 | 1.12 | 6.02 | 72 |
| C & W II | 10107.69 | 0.78 | 7.54 | 189 |
| C & W I | 10149.06 | 1.20 | 9.71 | 80 |
| Equal | 10147.46 | 1.24 | 10.13 | 97 |
| Best seed | 10197.88 | 1.80 | 11.54 | 61 |
| Average seed | 12654.54 | | | |
| FCFS | 14927.83 | 47.72 | 71.45 | 0 |

Table 4: Warehouse 1. Main Results




(a) Average (b) Deviation from the best (%) (c) Number of best

Figure 7: Warehouse 1. Main results for different instance sizes

Here we observe a marked effect of the instance size on the performance of the algorithms. In all cases, deviations from the best known solution are smaller as the instance size grows getting near 1% for all methods. We attribute this effect to the fact that Warehouse 1 is the smallest, with only 4 aisles, and batches of up to 12 items. In this setting, each route will most likely have to enter all aisles, and only small savings can be obtained with the different methods. Note that, in this case, VNS is the most robust method, giving similar deviations from the best solution known for all instance sizes. For the small instances these deviations are significantly smaller than with other methods, but, as the instance size grows, Clark and Wright II tends to give better results than VNS.

**Warehouse 2**

When restricted to Warehouse 2 instances (see Table 5), VNS is clearly the best of the algorithms tested, since it provides the best solution known for more than 80% of the instances and for the rest it provides solutions that are less than 5.1% above the best (0.21%, on average). In this warehouse, the 1-optimal solution is also a better alternative than those algorithms which already exist.

When the results for Warehouse 2 are disaggregated per instance size (see Figure 8), a uniform pattern is obtained. Here, the figures do not exhibit a clear dependency on the instance size, and the results obtained with VNS are consistently much better than those of the other methods tested.

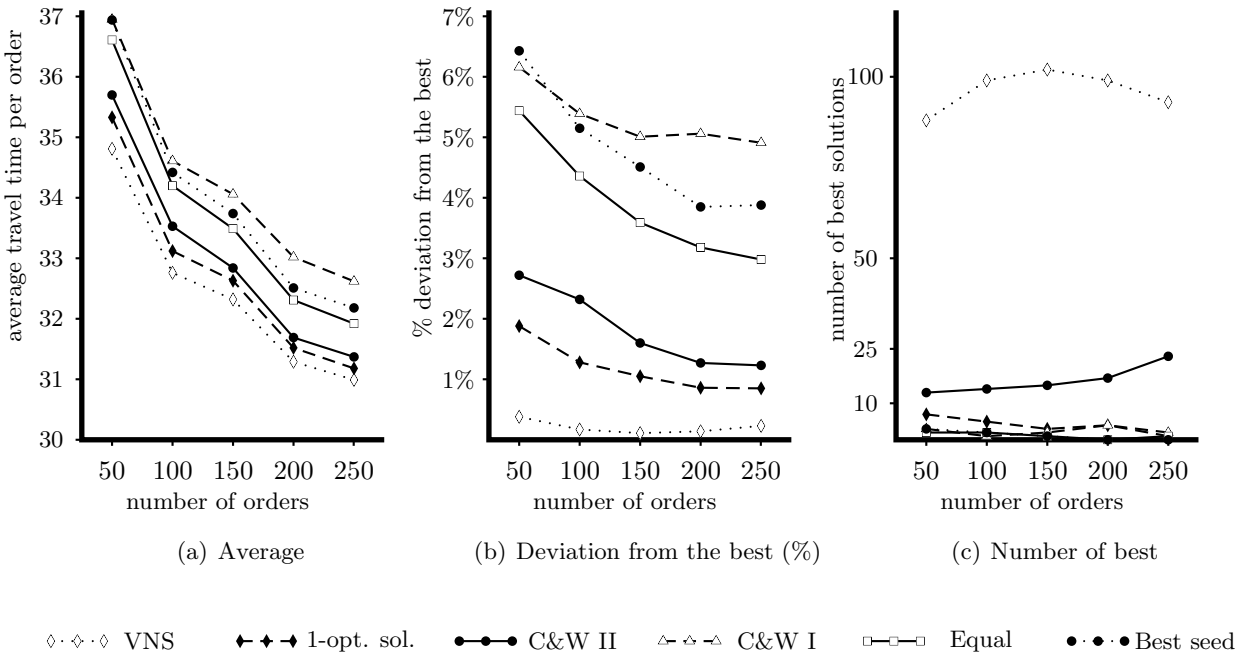
| | Average | %dev. from the best | | # of best |
|---|---|---|---|---|
| | objetive value | Average | Maximum | solutions |
| VNS | 4773.89 | 0.21 | 5.10 | 481 |
| 1-opt. sol. | 4814.21 | 1.18 | 9.30 | 20 |
| C & W II | 4848.70 | 1.83 | 10.84 | 82 |
| C & W I | 5035.33 | 5.30 | 19.25 | 12 |
| Equal | 4943.21 | 3.91 | 15.05 | 6 |
| Best seed | 4979.61 | 4.76 | 18.41 | 6 |
| Average seed | 6074.95 | | | |
| FCFS | 7123.35 | 49.85 | 76.95 | 0 |

Table 5: Warehouse 2. Main Results

**Warehouse 3**

As in the case of Warehouse 2, the results obtained in Warehouse 3 shown in Table 6, confirm the quality of the solutions proposed by the VNS algorithm. In this case, VNS provides the best known solution for 50% of the instances, and the deviation from the best known solution is, on average, under 1.3%. Moreover, in the worst case, the VNS deviation from the best is less than 7.5%, whereas applying all the other methods the worst case performance provides solutions that are over 18% above the best. Note that Warehouse 3 is a hypothetical warehouse. In this case, for instances with a large number of orders and when the *traversal* routing strategy is used, the best methods turned out to be seed heuristics. Even in this case, if each method is implemented together with the routing strategy that best suits it, VNS provides better results than methods already in existence.

In this warehouse it is important to note how all the existing methods, except in the case of the best seed, provide solutions that are significantly worse than the best. It is also noteworthy



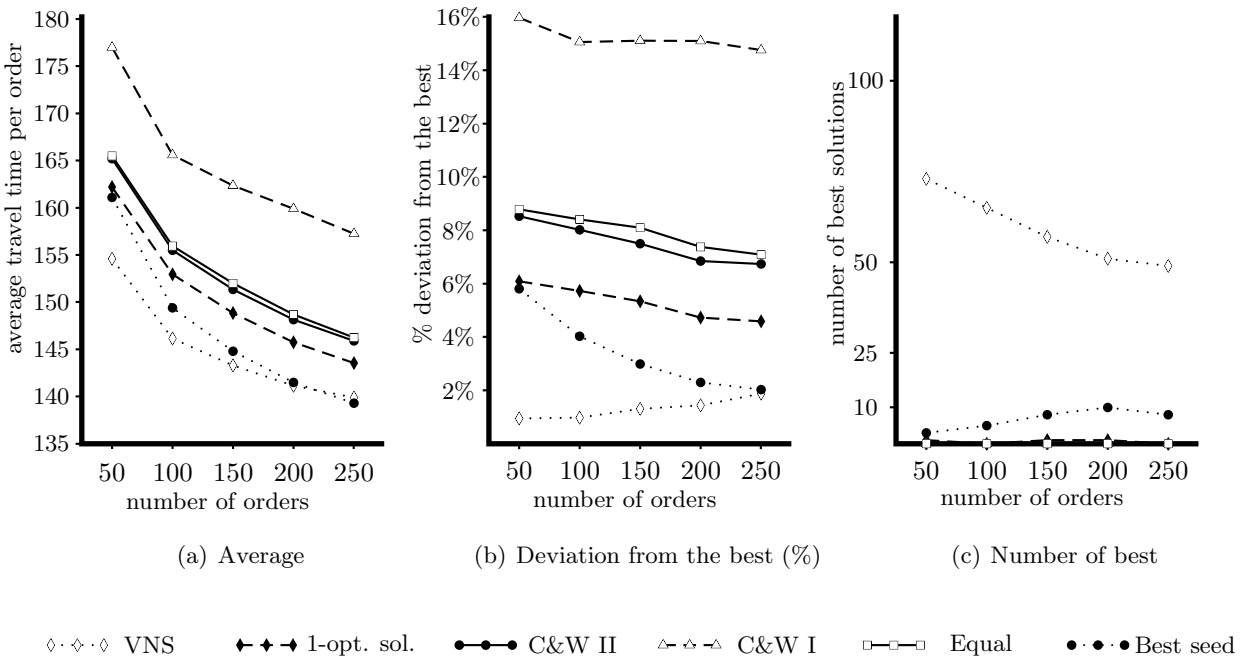

(a) Average (b) Deviation from the best (%) (c) Number of best

Figure 8: Warehouse 2. Main results for different instance sizes

that, as opposed to what occurs in the other warehouses, the seed method which performs best is not based on savings methods. However, there is no specific combination of seed selection and order addition rule that is uniformly better than the others. Similar to the case of the previous warehouse, the performance of the algorithms does not exhibit any dependency on the instance size, as can be seen in Figure 9.

| | Average objetive value | %dev. from the best Average | Maximum | # of best solutions |
|---|---|---|---|---|
| VNS | 21407.69 | 1.31 | 7.47 | 295 |
| 1-opt. sol. | 22154.95 | 5.30 | 13.34 | 3 |
| C & W II | 22524.25 | 7.53 | 22.27 | 0 |
| C & W I | 24211.67 | 15.20 | 43.82 | 0 |
| Equal | 22594.02 | 7.95 | 23.74 | 0 |
| Best seed | 21569.08 | 3.43 | 18.42 | 34 |
| Average seed | 23931.09 | | | |
| FCFS | 26494.56 | 26.74 | 55.67 | 0 |

Table 6: Warehouse 3. Main Results




(a) Average (b) Deviation from the best (%) (c) Number of best

Figure 9: Warehouse 3. Main results for different instance sizes

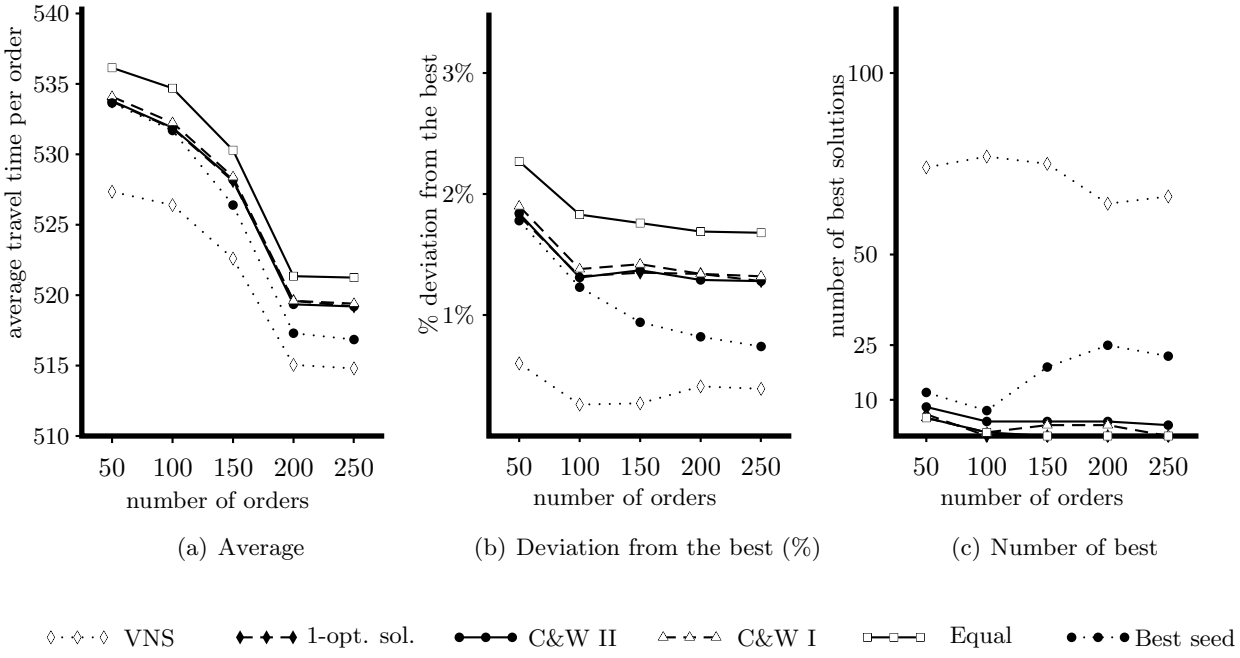
**Warehouse 4**

From the results obtained for Warehouse 4 (see Table 7 and Figure 10) we draw conclusions which are similar to those of the previous warehouses. VNS provides the best results, in terms of average total travel time, deviations from the best known solution (maximum and average), as well as in the number of instances when it provides the best solution. As in Warehouse 3, the best alternative to VNS is a seed method. When these results are disaggregated per instance size, it can be seen that VNS outperforms the other methods for all sizes, even though differences are not as significant as they were in the other warehouses.

After analyzing the performance of all methods per warehouse, we are able to draw some general conclusions. In this sense, from a general point of view, the VNS approach is the best. Moreover, the 1-optimal solution is competitive, and its solution is, on average, of the same quality as the alternatives. As expected (see [16]), savings methods obtain better solutions than seed methods. Furthermore, the seed method that exhibits the best performance from an overall perspective is where a savings rule is applied: The non-selected order with longest travel time is selected as the seed for a new batch (see [16]) and the order selected to be added to the current batch is the one that leads to the greatest time saving when added to the batch (derived from Clarke and Wright 1964, [3]). This combination is the best seed rule for Warehouses 1 and 2; whereas for Warehouse 4 the best rule uses other seed selection criteria, to select the order that

| | Average objetive value | %dev. from the best Average | %dev. from the best Maximum | # of best solutions |
|---|---|---|---|---|
| VNS | 77821.22 | 0.39 | 5.10 | 356 |
| 1-opt. sol. | 78560.66 | 1.42 | 8.73 | 6 |
| C & W II | 78556.02 | 1.42 | 8.19 | 23 |
| C & W I | 78591.43 | 1.47 | 8.19 | 12 |
| Equal | 78881.25 | 1.85 | 8.79 | 6 |
| Best seed | 78295.88 | 1.10 | 5.32 | 85 |
| Average seed | 84770.41 | | | |
| FCFS | 97201.31 | 24.79 | 35.63 | 0 |

Table 7: Warehouse 4. Main Results




(a) Average (b) Deviation from the best (%) (c) Number of best

Figure 10: Warehouse 4. Main results for different instance sizes

has the greatest number of picking locations to visit [14, 16, 20]. In contrast, the best seed rule for Warehouse 3 is not based on savings methods. As previously mentioned, there is no specific combination of seed selection and order addition rule that is uniformly better than the others for Warehouse 3. When we analyze the results per warehouse, we conclude that our proposed heuristic outperforms all other heuristics for all considered warehouses, except for Warehouse 1, where all algorithms are similar, and Clarke and Wright II seems to work better. There is an explanation for this fact: this warehouse is the smallest, in terms of number of aisles, and, therefore, there are no significant differences between the performed routes for each batch.

### Factor effects

Finally, since the performance of the algorithms depends heavily on factors such as storage policy, routing strategies, or the structure of the orders, we include an Analysis of Variance study in order to consider the effect of all the above factors on the different methods tested. Tables 8, 9 and 10 show the main results, separated according to each of the three factors that have been considered: routing strategy, depot location and storage policy, respectively. From Table 8, it can be stated that, in general, *combined* strategy works better than the other two routing strategies. Table 9 suggests that depot location is not a major factor in the proposed solution. As expected, storage policy is the most relevant factor (see Table 10), with a reduction of 30% in the total time travel spent when the demand-based policy is applied.

| | Average objective value | | | Av. %dev. from the best | | | # of best solutions | | |
|---|---|---|---|---|---|---|---|---|---|
| | Traversal | Lar. Gap | Combined | Trav. | L. Gap | Comb. | Trav. | L. Gap | Comb. |
| VNS | 29511.86 | 28937.70 | 27141.18 | 0.87 | 0.49 | 0.64 | 396 | 486 | 456 |
| 1-opt. sol. | 29904.73 | 29341.32 | 27510.44 | 2.33 | 2.13 | 2.30 | 51 | 23 | 27 |
| C & W II | 30046.10 | 29407.14 | 27574.25 | 3.24 | 2.61 | 2.82 | 98 | 97 | 99 |
| C & W I | 30582.45 | 29851.82 | 28056.35 | 6.56 | 5.16 | 5.66 | 29 | 39 | 36 |
| Equal | 30179.76 | 29535.51 | 27709.18 | 4.22 | 3.34 | 3.66 | 40 | 33 | 36 |
| Best Seed | 29959.15 | 29509.81 | 27677.04 | 4.26 | 3.69 | 3.97 | 40 | 27 | 32 |
| FCFS | 38205.37 | 36509.04 | 34595.88 | 39.50 | 35.98 | 36.35 | 0 | 0 | 0 |

Table 8: Routing strategy

| | Average objective value | | Av. %dev. from the best | | # of best solutions | |
|---|---|---|---|---|---|---|
| | Aisle 0 | Central | Aisle 0 | Central | Aisle 0 | Central |
| VNS | 28541.85 | 28518.64 | 0.74 | 0.60 | 610 | 728 |
| 1-opt. sol. | 28918.66 | 28919.00 | 2.28 | 2.23 | 52 | 49 |
| C & W II | 28997.02 | 29021.31 | 2.77 | 3.01 | 167 | 127 |
| C & W I | 29474.78 | 29518.96 | 5.72 | 5.87 | 63 | 41 |
| Equal | 29112.44 | 29170.53 | 3.54 | 3.94 | 58 | 51 |
| Best Seed | 29010.29 | 29087.05 | 3.77 | 4.17 | 52 | 47 |
| FCFS | 36402.98 | 36470.54 | 36.57 | 37.98 | 0 | 0 |

Table 9: Depot location

Taking into account that these tables aggregate data from different warehouses and, in order to confirm that the performance of the algorithms depends on these factors, an ANOVA analysis per warehouse has been performed. Three factors have been considered: Routing strategy (3

| | Average objective value | | Av. %dev. from the best | | # of best solutions | |
|---|---|---|---|---|---|---|
| | Random | ABC | Random | ABC | Random | ABC |
| VNS | 33499.93 | 23560.56 | 0.84 | 0.51 | 657 | 681 |
| 1-opt. sol. | 33900.37 | 23937.29 | 2.31 | 2.20 | 39 | 62 |
| C & W II | 33937.44 | 24080.89 | 2.71 | 3.07 | 116 | 178 |
| C & W I | 34617.06 | 24376.68 | 6.26 | 5.33 | 33 | 71 |
| Equal | 34083.84 | 24199.13 | 3.53 | 3.95 | 45 | 64 |
| Best Seed | 34043.09 | 24054.25 | 3.77 | 4.17 | 26 | 73 |
| FCFS | 42103.95 | 30769.58 | 31.81 | 42.75 | 0 | 0 |

Table 10: Storage policy

levels), depot location (2 levels), and storage policy (2 levels). 600 instances have been included in each analysis. In order to reduce the impact of the size, the dependent variable considered is the average travel time per order, defined as the ratio between the total travel time and the number of orders.

As a first step, the analysis has been applied to the VNS solution. In an iterative process, the 3-factor, depot location-routing strategy and depot location-storage policy interactions have been removed, since they are not significant. Table 11 shows the final ANOVA analysis, once the non-significant interactions were removed.

As can be seen in Table 11, the routing strategy, the storage policy and their interaction are the most relevant factors. Figure 11 represents the average travel time for both factors, storage policy and routing strategy: the impact of the routing strategy is less important when demand-based storage policy is applied, since 80% of demand is concentrated into a small number of aisles (only one and two aisles for Warehouses 1 and 2, respectively) and, therefore, all routing strategies build very similar routes (for Warehouse 1 and 2, there are no significant differences). Also from Figure 11, it can be stated that *combined* is the best routing strategy, as expected, since this rule chooses the best option between the other two rules. Results for *largest gap* are very similar to *combined* for Warehouses 1, 2 and 4 and better than *traversal*, while *traversal* is a valid alternative to *combined* for Warehouse 3, regardless of the storage policy. This conclusion is coherent with the rule stated by Hall [12], "*largest gap* is preferred to *traversal* when the number of picks per aisle is approximately less than 3.8": In our case, the average number of items retrieved per aisle and batch is 3, 2.4, 6 and 3.33 for warehouses 1 to 4, respectively. Finally, we should point out that these models explain more than 80% of the variability of the dependent variable (the determination coefficient, $R^2$, ranges from 0.79 to 0.94).

The same analysis has been performed for the solutions obtained with Clarke and Wright I, Clarke and Wright II and Equal, and the conclusions for those algorithms are very similar. As

Warehouse 1

| Source | Sum of Squares | df | Mean Square | $F$ | Sig. |
|---|---|---|---|---|---|
| Corrected Model | 141710.07 | 6 | 23618.34 | 823.77 | 0.000 |
| Intercept | 2794221.87 | 1 | 2794221.87 | 97457.70 | 0.000 |
| Routing strategy | 2747.03 | 2 | 1373.52 | 47.91 | 0.000 |
| Depot location | 6.73 | 1 | 6.73 | 0.23 | 0.628 |
| Storage policy | 136697.47 | 1 | 136697.47 | 4767.78 | 0.000 |
| Routing Str. * Storage Po. | 2258.83 | 2 | 1129.41 | 39.39 | 0.000 |
| Error | 17001.98 | 593 | 28.67 | | 0.000 |
| Total | 2952933.91 | 600 | | | 0.000 |
| Corrected Total | 158712.04 | 599 | | | 0.000 |

∗R squared = .893 (Adjusted R Squared = .892)

Warehouse 2

| Source | Sum of Squares | df | Mean Square | $F$ | Sig. |
|---|---|---|---|---|---|
| Corrected Model | 45018.06 | 6 | 7503.01 | 1176.58 | 0.000 |
| Intercept | 631134.36 | 1 | 631134.36 | 98970.89 | 0.000 |
| Routing strategy | 638.72 | 2 | 319.36 | 50.08 | 0.000 |
| Depot location | 399.23 | 1 | 399.23 | 62.61 | 0.000 |
| Storage policy | 43619.87 | 1 | 43619.87 | 6840.22 | 0.000 |
| Routing Str. * Storage Po. | 360.24 | 2 | 180.12 | 28.25 | 0.000 |
| Error | 3781.54 | 593 | 6.38 | | 0.000 |
| Total | 679933.96 | 600 | | | 0.000 |
| Corrected Total | 48799.60 | 599 | | | 0.000 |

∗R squared = .923 (Adjusted R Squared = .922)

Warehouse 3

| Source | Sum of Squares | df | Mean Square | $F$ | Sig. |
|---|---|---|---|---|---|
| Corrected Model | 713916.17 | 6 | 118986.03 | 1540.59 | 0.000 |
| Intercept | 12617298.02 | 1 | 12617298.02 | 163364.57 | 0.000 |
| Routing strategy | 35050.96 | 2 | 17525.48 | 226.91 | 0.000 |
| Depot location | 23.17 | 1 | 23.17 | 0.30 | 0.584 |
| Storage policy | 677544.25 | 1 | 677544.25 | 8772.62 | 0.000 |
| Routing Str. * Storage Po. | 1297.80 | 2 | 648.90 | 8.40 | 0.000 |
| Error | 45799.76 | 593 | 77.23 | | 0.000 |
| Total | 13377013.94 | 600 | | | 0.000 |
| Corrected Total | 759715.93 | 599 | | | 0.000 |

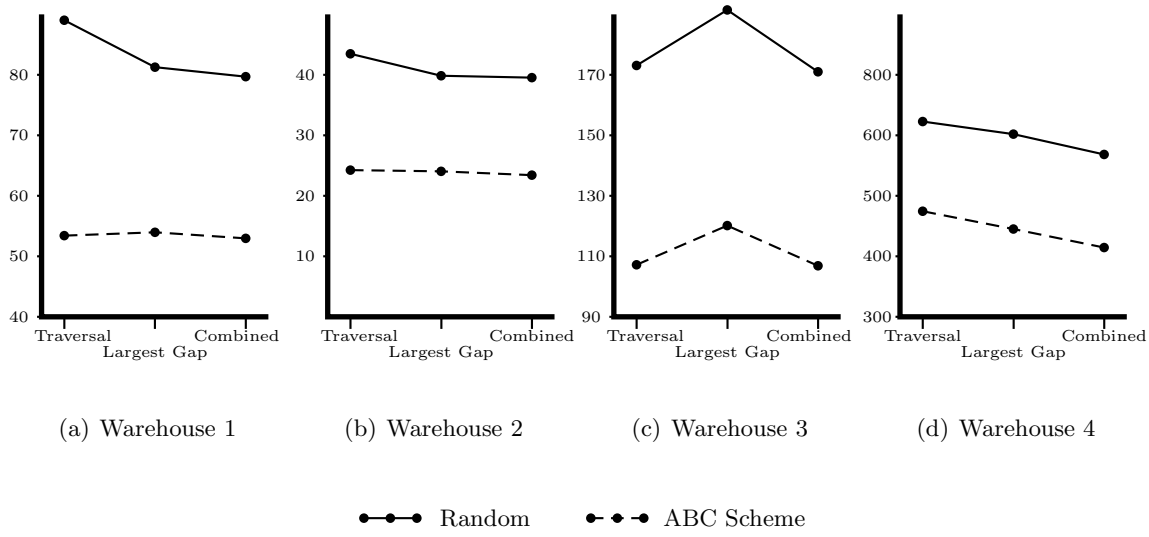
∗R squared = .940 (Adjusted R Squared = .939)

Warehouse 4

| Source | Sum of Squares | df | Mean Square | $F$ | Sig. |
|---|---|---|---|---|---|
| Corrected Model | 3843359.31 | 6 | 640559.89 | 376.17 | 0.000 |
| Intercept | 163014397.62 | 1 | 163014397.62 | 95731.39 | 0.000 |
| Routing strategy | 328610.74 | 2 | 164305.37 | 96.49 | 0.000 |
| Depot location | 40.13 | 1 | 40.13 | 0.02 | 0.878 |
| Storage policy | 3512794.47 | 1 | 3512794.47 | 2062.91 | 0.000 |
| Routing Str. * Storage Po. | 1913.98 | 2 | 956.99 | 0.56 | 0.570 |
| Error | 1009778.94 | 593 | 1702.83 | | 0.000 |
| Total | 167867535.87 | 600 | | | 0.000 |
| Corrected Total | 4853138.25 | 599 | | | 0.000 |

∗R squared = .792 (Adjusted R Squared = .790)

Table 11: ANOVA per warehouse




(a) Warehouse 1 (b) Warehouse 2 (c) Warehouse 3 (d) Warehouse 4

Figure 11: Post-analysis for Routing and Items location interaction

was found with the VNS results, storage policy is the most relevant factor as expected (with 30% savings when demand-based storage is applied). Routing strategy is also relevant, as *combined* is the best option, while *largest gap* is also competitive for Warehouses 1, 2 and 4 and *traversal* is a good alternative for Warehouse 3. Finally, the depot location is not relevant, except for Warehouse 2, where central location produces a savings of 5%.

In the previous pages, using a basic descriptive analysis, we have seen that VNS works better than the other alternatives. To decide whether the difference between our approach and the other methods is statistically significant, we have performed the following statistical hypothesis test: $H_0 : \mu_i \leq \mu_{VNS}$ versus $H_1 : \mu_i > \mu_{VNS}$, where $\mu_{VNS}$ is the mean value of the solution proposed by $VNS$ algorithm and $\mu_j$ represent the mean value of the solution proposed by algorithm $j$. For each warehouse and order size we have compared VNS with the 1-opt, Clarke and Wright I and II, Equal, Best Seed and FCFS algorithm. We have considered a significance level equal to 99%, so the $H_0$ hypothesis is rejected only if the $p$-value is lower than 0.01. According to the results obtained, $H_0$ is rejected, which means that the average VNS solution is significantly better than the average solutions of each of the other methods. Only for bigger instances (of orders 150, 200 and 250) for Warehouse 1 and saving methods (Clarke and Wright I and II and Equal), and for Warehouse 4 and the solutions provided by the best seed method are comparable to those provided by VNS. In those cases, Clarke and Wright II is significantly better than VNS (in the hypothesis test $H_0 : \mu_{VNS} \leq \mu_{C\&WII}$ versus $H_1 : \mu_{VNS} > \mu_{C\&WII}$, $H_0$ is rejected). In order to study if this conclusion can be extrapolated to all warehouses, we have performed a deeper analysis, comparing the difference between VNS and Clarke and Wright II. A 4-factor

ANOVA has been made (warehouse, routing strategy, storage policy and depot location), using the difference between the solution proposed by both methods. The first conclusion is that these factors cannot explain the difference ($R^2$ is really small and ranges 0.05 to 0.35). Therefore, it can be concluded that VNS is better than Clarke and Wright II, as a general algorithm, although in some specific cases Clarke and Wright II can outperform VNS.

### Computation time

To conclude the analysis, we include some comments on the computation time needed for obtaining the solutions applying the different methods. All of the algorithms tested in this paper have been implemented in C++. The computational experiments were conducted on a Dual Core with 2.00Ghz and 2GB of RAM. The Microsoft Visual C++ compiler v6.0 has been used.

VNS took more time, but even in the most difficult scenario (Warehouse 3), the average time needed to solve the 120 instances with 250 orders is around 35 seconds. On the other hand, Clarke and Wright needed 6 seconds. Taking into account that this is the warehouse where VNS performs the best (on average 6% better than Clarke and Wright II), this extra computational effort can be justified. Solutions for Warehouses 1, 2 and 4 were obtained in less than 30 seconds.

## 6 Conclusions

In this paper, we have dealt with the order batching problem in a parallel aisle warehouse with the objective of minimizing the total travel time. This is a highly relevant material handling problem, for which we have presented a new metaheuristic method based on variable neighborhood search. Simulations were conducted to study the performance of the method with four different warehouse configurations. From the test results it follows that the proposed method, VNS, is a competitive alternative to widely used methods in real-world warehouses. Total travel times can be significantly reduced with a moderate increase in computation times. It should be noted that the comparative study developed is even more exhaustive than that published by De Koster *et al.* in [16], which is described in [11] as one of the most exhaustive comparative analysis published up to 2007. In particular, we include an additional warehouse layout, more order batching rules and additional factors, as the central depot location and the combined routing strategy. With regard to future research, three main research lines are proposed: to include more constraints in the problem, such as, the *due dates* of the orders, and then change the objective to minimize the *makespan*; to formulate and compare different mixed integer programming models of the problem and, based on these formulations, try to find exact and/or heuristics methods along and lower bounds to assess the quality of the heuristic solutions.

## Acknowledgement

We thank one of the referees for his careful reading of the paper and useful suggestions that have improved the paper.

## References

[1] R.D. Armstrong, W.D. Cook, and A.L. Saipe. Optimal batching in a semi-automated order picking system. *Journal of Operational Research Society*, 30(711-720), 1979.

[2] M.-C. Chen and H.-P. Wu. An association-based clustering approach to order batching considering customer demand patterns. *Omega, International Journal of Management Science*, 33(333-343), 2005.

[3] G. Clarke and W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[4] J.J. Coyle, E.J. Bardi, and C.J. Langley. *The Management of Business Logistics*. South Western College Publishing, 1996.

[5] R. De Koster, K.J. Roodbergen, and R. Van Voorden. Reduction of walking time in the distribution center of de bijenkorf. Technical report, Rotterdam School of Management, Erasmus University of Rotterdam, 1998.

[6] E.A. Elsayed and I.O. Unal. Order batching algorithms and travel-time estimation for automated storage/retrieval systems. *International Journal of Production Research*, 27:1097–1114, 1989.

[7] N. Gademann, J. Berg, and H. Hoff. An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Transactions*, 33:385–398, 2001.

[8] N. Gademann and S. Velde. Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75, 2005.

[9] G. Ghiani, G. Laporte, and R. Musmanno. *Introduction to Logistic Systems Planning and Control*. Wiley, 2004.

[10] D.R. Gibson and G.P. Sharp. Order batching procedures. *European Journal of Operational Research*, 58:57–67, 1992.

[11] J. Gu, M. Goetschalckx, and L.F. McGinnis. Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177:1–21, 2007.

[12] R.W. Hall. Distance approximation for routing manual pickers in a warehouse. *IIE Transactions*, 25:77–87, 1993.

[13] P. Hansen and N. Mladenović. Variable neighborhood search. In F. Glover and G.A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publishers, 2003.

[14] Y.-C. Ho and Y.-Y. Tseng. A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. *International Journal of Production Research*, 44(17):3391–3417, 2006.

[15] C.-M. Hsu, K.-Y. Chen, and M.-C. Chen. Batching orders in warehouses by minimizing travel distances with genetic algorithms. *Computers in Industry*, 56:169–178, 2005.

[16] M.B.M. De Koster, E.S. van der Poort, and M. Wolters. Efficient orderbatching methods in warehouses. *International Journal of Production Research*, 37(7):1479–1504, 1999.

[17] R. De Koster, T. Le-Duc, and K.J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182:481–501, 2007.

[18] N. Mladenović. A variable neighborhood algorithm–a new metaheristic for combinatorial optimization. In *Abstracts of papers presented at Optimization Days*, page 112, Montreal, 1995.

[19] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers Operations Research*, 24:1097–1100, 1997.

[20] C.-H. Pan and S.-Y. Liu. A comparative study of order batching algorithms. *Omega, International Journal of Management Science*, 23:691–700, 1995.

[21] C.G. Petersen. Routeing and storage policy interaction in order picking operations. *Decision Science Institute Proceedings*, 31(3):1614–1616, 1995.

[22] C.G. Petersen. An evaluation of order picking routeing policies. *International Journal of Operations and Production Management*, 17(11):1098–1111, 1997.

[23] H.D. Ratliff and A.S. Rosenthal. Order picking in a rectangular warehouse: a solvable case of the travelling salesman problem. *Operations Research*, 31:507–521, 1983.

[24] K.J. Roodbergen and R. De Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001.

[25] K.J. Roodbergen and C.G. Petersen. How to improve order picking efficiency with routing and storage policies. In G.R. Forger, S.S. Heragu, M.A. Hernan, B.A. Peters, G.D. Taylor, and J.S. Usher, editors, *Progress in Material Handling Practice*, pages 107–124. Material Handling Institute, 1999.

[26] M.B. Rosenwein. A comparison of heuristics for the problem of batching orders for warehouse selection. *International Journal of Production Research*, 34:657–664, 1996.

[27] R. Shamlaty. This is not your father's warehouse. *IIE Solutions*, 1:31–36, 2000.

[28] J.P. Van den Berg. A literature survey on planning and control of warehousing systems. *IIE Transactions*, 31(8):751–762, 1999.

[29] M. Wolters. *Het batchen van orders in een magazijn; een vergelijking van heuristieken.* Master thesis, University of Groningen, 1996. (in Dutch).