Volume 35 (2), pp. 161–186
http://orion.journals.ac.za

ORiON
ISSN 0259–191X (print)
ISSN 2224–0004 (online)
©2019

Picking location metrics for order batching on a unidirectional cyclical picking line

FM Hofmann[*] SE Visagie[†]

Received: 18 March 2019; Revised: 5 July 2019; Accepted: 25 July 2019

Abstract

In this paper order batching is extended to a picking system with the layout of a unidirectional cyclical picking line. The objective is to minimise the walking distance of pickers in the picking line. The setup of the picking system under consideration is related to unidirectional carousel systems. Three order-to-route closeness metrics are introduced to approximate walking distance, since the orders will be batched before the pickers are routed. All metrics are based on the picking location describing when a picker has to stop at a location to collect the items for an order. These metrics comprise a number of stops, a number of non-identical stops and a stops ratio measurement. Besides exact solution approaches, four greedy heuristics as well as six metaheuristics are applied to combine similar orders in batches. All metrics are tested using real life data of 50 sample picking lines in a distribution centre of a prominent South African retailer. The capacity of the picking device is restricted, thus the maximum batch size of two orders per batch is allowed. The best combination of metric and solution approach is identified. A regression analysis supports the idea that the introduced metrics can be used to approximate walking distance. The combination of stops ratio metric and the greedy random heuristic generate the best results in terms of minimum number of total cycles traversed as well as computational time to find the solution.

Key words: Order batching, picking location metrics, unidirectional cyclical picking line, carousel

1 Introduction

The need for greater product variety and shorter response times emphasises a companies' ability to establish efficient logistical operations. This efficiency is, amongst others, determined by the operation of its warehouses or distribution centres (DC) that define the

[*]Department of Logistics, Stellenbosch University, South Africa, email: 20304269@sun.ac.za
[†]Corresponding author: **(Fellow of the Operations Research Society of South Africa)** Department of Logistics, Stellenbosch University, South Africa, email: svisagie@sun.ac.za

http://dx.doi.org/10.5784/35-2-646

nodes in the distribution network [48]. In a supply chain connecting production plants with end customers warehousing cannot be eliminated. The role of warehouse operations is constantly changing and thus has to remain flexible. A DC focuses on the consolidation and accumulation of numerous products from various suppliers to many customers. Products from different suppliers arrive in bulk at the DC. The stock is then turned into customer or store orders for delivery to the DC's customers [18].

Van den Berg & Zijm [54] subdivide DC activities into four categories namely receiving, storage, order picking and shipping. From the total logistics cost of a company, approximately 25% can be attributed to the cost of operating a warehouse [23]. Therein, the basic service of a DC is order picking that makes up for 50% to 65% of the operational cost accounting for labour, capital and supporting activities [54]. De Koster *et al.* [12] define the process of order picking as retrieving products from storage or buffer areas and turning them into specific orders in response to a customer request. The design of the order picking system is crucial to its performance. The number of orders together with the items per order that are picked in a day, the average size of an order and the layout of the storage racks are key parameters to set up an efficient order picking system [10].

The order picking system of a prominent South African retailer (referred to as the Retailer) with around 2 000 outlets or stores is considered here. A set of non-uniform orders is the result of a large number of stores with different sizes and various customer profiles that needs to be handled by the DC on a daily basis. A key characteristic of the Retailer's operations is the central planning of the inventory kept at a store level. Instead of stock requested by store managers, planners at a central planning department assign stock that is available at the DC to stores. Thereby, the central planning department allocates stock keeping units (SKUs) to stores. Planners in the central planning department decide on the number of SKUs destined for each store after the arrival of the SKUs at the DC. Planners issue instructions to the DC about the SKUs and the stores where they should go. The DC then selects a subset of SKUs to be picked in a single picking wave to satisfy all store requirements for that set of SKUs. In this study, the term order will thus refer to the set of store requirements for a single store for all SKUs selected to be processed in a wave. A wave is processed on a picking line in the DC. A single SKU is assigned to a single location on a specific picking line for each wave. The activities of populating the line with SKUs, the actual picking process and removing excess stock from the line comprise a picking wave [36].

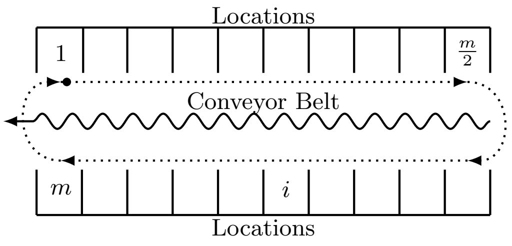


Figure 1: *A schematic representation of a picking line with m locations.*

A schematic representation of a picking line with m locations is depicted in Figure 1. At the Retailer's DC a picking line consists of 56 locations with a conveyor belt placed in the middle that has two access gates. From the storage area, a single SKU is assigned to a single location on the picking line. The number of items of each SKU to be picked for all stores are known prior to the start of a picking wave. Each location has the storage capacity of five pallets of an identical SKU. Additional stock is kept on the floor space between different picking lines for easy refill avoiding stock-outs. Therefore, stock does not have to be replenished from storage racks during a picking wave. Pickers move around the conveyor belt in a clockwise direction picking all orders. A voice recognition system (VRS) guides the pickers through the picking process by sending a picker to the closest required SKU. Therefore, the underlying picking strategy can be summarised as pick the closest SKU in a clockwise direction. The picking system is categorised as a picker-to-parts system. Before the picking of an order is started, an empty carton is placed on the trolley and registered through a barcode with the VRS. After picking, full cartons and finished orders are placed on the conveyor belt for further processing in the dispatch area of the DC [34, 35].

The setup of the picking system at the DC shows many similarities to unidirectional carousel systems, if the SKUs are viewed as moving relative to a static picker. A carousel is an automated warehouse system that can be used for picking small and medium products. It is designed as a rotatable circuit of shelving holding multiple SKUs. A picker, who remains at a fixed location during the picking process, operates the system. In literature this picking system is referred to as a parts-to-picker system, since the storage location travels to the picker instead of the picker travelling to the storage location. A bidirectional carousel can rotate in both directions to bring the required SKU in front of the picker, whereas a unidirectional carousel can only move in one direction [3]. The cyclical setup of the picking line and the assumptions that one picker processes one order as well as the automation of pick sequencing by the VRS resembles a unidirectional carousel. However, the presence of wave picking is the main difference in the Retailer's system to the carousel systems studied in literature. Not all SKUs may be picked and new orders may be added to the set during the picking operation in typical carousel systems. Therefore, optimisation approaches make use of expected SKU mixes in orders that are derived from historical data [33]. The deterministic nature of the orders (because all orders are known and fixed when a wave of picking starts) is unique to the Retailer's picking system in operations research literature [36].

Planning for order picking includes the order batching problem. Orders that are requested are subsequently released for fulfilment. This set of orders needs to be picked and accumulated for packing and shipping in a picking wave [22]. The combination of customer orders into picking orders describes the process of order batching [24]. According to Wäscher [55] order batching answers the question of how customer orders should be grouped into picking orders to minimise the total length of all picking tours necessary to collect all items while no customer order is split, given a set of customer orders consisting of a number of items, an assignment of items to storage locations in the order picking system of a warehouse and the capacity of a picking device. Tours that efficiently shorten the picking time can be generated, since several orders are picked simultaneously leading to a reduction in labour cost [11].

While there has been intensive research on the order batching problem in single-block warehouses with parallel aisles, according to Nicolas *et al.* [42] little research has been published on carousel systems. The literature on carousels can be categorised into either determining the pick sequence on a carousel or finding the best storage location for a product in a carousel. No study on order batching applied to a manual unidirectional carousel systems as implemented by the Retailer could be found in literature.

This study aims at answering the question if the order batching problem (OBP) can be applied to the specific layout of a unidirectional cyclical picking line. The orders will be batched before the pickers are routed. However, walking distance can not be determined *a priori* and thus a realistic approximation for distance has to be provided. Different location-based metrics are thus developed to estimate walking distance, since calculating walking distance is too time consuming. Different algorithms that correlate with these metrics to solve the order batching problem are tested. A suggested combination of metric and algorithm to minimise the total distance travelled by a picker to collect all orders during a picking wave concludes the study.

A brief overview on the literature of order-batching in single-block and automated storage and retrieval warehouse setups is given in Section 2. The model will be described in Section 3 including assumptions, measurements and an integer programming formulation of the problem. The location-based order-to-route closeness metrics will be proposed in Section 4. In Section 5 different heuristic approaches comprised of greedy heuristics and metaheuristics are introduced to solve the OBP in reasonable time. A case study with real life data from the Retailer is used to test the performance of the combinations of metrics and algorithms in Section 6. In Section 7 a summary as well as an outlook on future research opportunities is provided.

2 Literature review

The formulation of the standard order batching problem (OBP) was introduced and proved to be NP-hard in the strong sense by Gademann & Velde [19] for a single-block warehouse with parallel aisles. However, if no batch contains more than two orders the problem can be solved in polynomial time. The branch-and-prize algorithm developed by Gademann & Velde [19] solved test instances of up to 32 orders to optimality. Additionally, Henn & Wäscher [25] solved instances of up to 40 orders. Nevertheless, their solution approach required a time-consuming preprocessing of all feasible batches. Furthermore, Muter & Öncan [41] solved instances of up to 100 orders with their specially tailored column-generation based algorithm. In practice the number of customer orders is likely to exceed these test instances, but the exact algorithms proposed so far are not able to consistently solve larger instances. Nicolas *et al.* [42] introduced the OBP to a vertical carousel system that is closely related to the layout of the unidirectional cyclical picking line. The OBP is formulated as a mixed integer linear program (MILP). Nevertheless, the MILP has to be stopped after 30 minutes due to time constraints for test instances over 50 orders. In the Retailer's DC a picking line can service up to 1 500 orders. Larger problem instances must be solved using heuristics. Therefore, construction heuristics like seed algorithms and saving algorithms have been introduced to solve the OBP.

A straight-forward heuristic is the first-come-first-served method that groups together the first entries of the list of orders as close as possible to the predetermined maximum batch size. Then the next orders are grouped using a similar logic until all orders form part of a batch [20].

Seed algorithms start by initiating batches, then allocating orders to batches and terminate through a stopping rule when a batch has been completed. The objective is to minimise the total travel distance for collecting all orders. De Koster *et al.* [11] evaluated seed algorithms in a parallel aisle warehouse proposing the seed selection rules of the farthest storage location. Ho & Tseng [27] investigated several location- and aisles-based seed algorithms in the standard OBP environment of a single-block parallel aisle warehouse. Ho *et al.* [26] extended this study with further distance- and area-based selection rules in this environment. In their comparative study Pan & Liu [45] evaluated four initial seed selection and four order addition rules investigating the OBP in an automated storage and retrieval system that consists of a single storage rack with equally sized storage locations serviced by a single storage and retrieval machine.

Saving algorithms in the OBP are based on the Clarke and Wright-algorithm (CW) [9] and thus the time saving that can be obtained by comparing the collection of orders in one route to individual picking [11]. The minimum additional aisles heuristic proposed by Rosenwein [47] starts with calculating a score for each pair of orders in terms of additional aisles to pick from in the cases that orders are picked simultaneously or separately. Therefore, their algorithm is capable of generating fewer but shorter picking tours when compared to Gibson & Sharp [20]. The first OBP version of the algorithm (CW i) described by De Koster *et al.* [11] computes savings for each combination of orders. Each time a new combination of orders to batches has been determined the savings are recalculated in the second version of the algorithm (CW ii). Bozer & Kile [6] improved this version of the algorithm by introducing a normalised time saving value for each pair of orders. Each time an order has been assigned the initial savings matrix is modified resulting in a third version (CW iii) proposed by De Koster *et al.* [11]. Additionally, Elsayed & Unal [17] proposed a small and large algorithm. Therefore, orders are either classified as small or large according to a predefined value before they are assigned to batches. In general seed algorithms are more central processing unit (CPU) time saving than saving algorithms. The order-to-route closeness metric is central to all types of heuristics [22].

Metaheuristics are designed to solve a wide range of hard optimisation problems and can thus also be applied to the OBP. An iterated local search that explores a neighbourhood to identify a new solution with a smaller objective function value was applied to the OBP by Henn *et al.* [23]. Albareda-Sambola *et al.* [2] defined three neighbourhoods to introduce the OBP to a variable neighbourhood search. A tabu search that simulates the human memory processes including a tabu list was applied to the OBP by Henn & Wäscher [25]. Matusiak *et al.* [38] introduced a simulated annealing approach that simulates the cooling process of metal to the OBP. Moreover, Öncan [43] introduced a combination of an iterated local search with a tabu threshold to the OBP. A hybrid of a large adaptive neighbourhood search and a tabu search was developed by Žulj *et al.* [56] based on the findings of Henn & Wäscher [25] and Öncan [43].

3 Model formulation

The problem environment of this study is a DC of the Retailer. The DC is made up of several unidirectional cyclical picking lines functioning in parallel. Thus one such picking line will be in the centre of attention. This layout differs significantly from the parallel-aisle or the single-aisle automated storage and retrieval warehouses that are frequently studied in academic literature.

The following assumptions are made while modelling the Retailer's order picking system.

1. The orders that need to be picked during a wave as well as the SKUs and their location in the picking line will be fixed *a priori*.
2. For movability the picking devices are small. However, SKUs are generally bulky. Therefore, the capacity of the picking devices is currently restricted to accommodate two orders at a time restricting the batch size to two.
3. A picker must complete the entire order before starting the next, packing stock directly onto the trolley. When an order is completed or the carton is full, it is placed on the conveyor belt for further processing.
4. It is assumed that a picker walks at a constant speed and that the time taken to pick and pack a SKU or switch between orders is constant.

3.1 Measurements

The cycles traversed measurement, that was introduced by Matthews & Visagie [36], counts the number of cycles that have to be traversed to pick all items requested during a picking wave as a measure of distance travelled to pick all orders. This measurement counts the total number of cycles required to pick and link all orders.

In the system currently in use by the Retailer each order is processed by one picker completing one order at a time. Orders are sequentially assigned to the next available picker based on a fixed list of orders. In effect each order is assigned randomly, since the assignment does not take the previous order or the current position of the picker into consideration. Therefore, each picker picks a random sequence of orders. Matthews & Visagie [36] proposed a nearest end heuristic as an easily implementable option to determine a sequence of orders that minimises the distance that pickers have to travel during a wave. The nearest end heuristic sequentially selects the order with the nearest ending position from the current picking position. Therefore, it simultaneously considers the order sequence and the item sequence within the order. An example of a unidirectional cyclical picking line with 10 locations and 10 SKUs is depicted in Figure 2. Four different orders are indicated by the colours green, blue, red and yellow. The different shapes represent the SKUs and locations respectively. If the nearest end heuristic is applied to this example, the Order 4 (yellow) would be picked first, starting at the first location and ending at location seven. This would be followed by the picking of Order 1 (green), then Order 3 (red) and finally the Order 2 (blue). In total a picker has to traverse four cycles to collect all orders of the picking wave. The nearest end heuristic is easily reproducible. Therefore, it is used in determining an order sequence to test the effectiveness of the various batching logics in terms of number of cycles traversed.

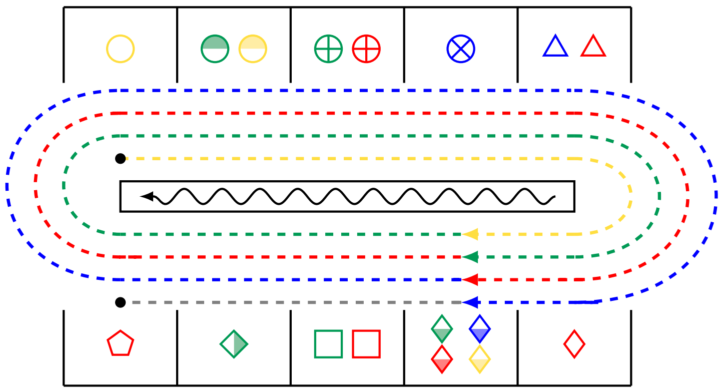


Figure 2: *Schemantic representation of a picking line with* 10 *SKUs and* 10 *locations.*

This study focuses on location-based order-to-route-closeness metrics. Each picking location defines where a picker has to stop at a location to pick items for an order. Different picking location or so-called stop metrics will be developed to identify compatible orders in terms of number of stops in common. The assumption is that a good overlap in stops may lead to a reduction in total walking distance.

3.2 Exact formulation

Minimising the completion time is equivalent to minimising the travel distance to collect all orders in a picking wave. Therefore, the objective is to minimise the incompatibility in terms of distance between orders that are described by a picking location metric to obtain the smallest number of cycles that have to be traversed. Orders must be combined in batches of size two. This can be formulated as an integer programming model. Let

n	be the number of orders,
c_{ij}	be the cost in terms of incompatible stops to batch order i and order j,

and define the set of variables as

$$x_{ij} = \begin{cases} 1, & \text{if order } i \text{ and order } j \text{ are in the same batch,} \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to

$$\text{minimise} \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij}\, x_{ij} \tag{1}$$

subject to

$$\sum_{k=1}^{i-1} x_{ki} + \sum_{j=i+1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \tag{2}$$

$$x_{ij} \in \{0,1\} \qquad \begin{cases} i & = 1, \ldots, n \\ j & = i+1, \ldots, n. \end{cases} \tag{3}$$

The incompatibility between orders in terms of stops expressed by a picking location metric is minimised by objective function (1). The set of triangular inequality constraints (2) and the binary condition (3) assign each order to only one other order in the case of a symmetric cost matrix.

The problem formulated in (1) – (3) is in essence an assignment problem. Another option to solve the assignment problem is the quick match algorithm that was introduced by Orlin & Lee [44]. It is based on the successive shortest path algorithm and combines a forward Dijkstra with a reverse Dijkstra algorithm. Moreover, heuristics are included to speed up its performance. Therefore, this algorithm will also be included for testing purposes.

4 Picking location metrics

The simplest way of introducing order batching to a unidirectional cyclical picking line would be to implement a first-in-first-out (FIFO) approach. According to the FIFO rule the first entries of a list of orders are grouped together until the predetermined maximum batch size is reached [20]. FIFO is often used in literature as a benchmark to compare different batching methods. It will also be used here to compare the metrics that measure the incompatibility of orders in terms of stops. An example of the picking locations of four orders is illustrated in Table 1. Employing the nearest end heuristic as a routing strategy (Section 3.1), a picker has to traverse four cycles to collect all four orders. If FIFO is applied as the random batching strategy and a batch is only allowed to contain two orders as illustrated in Figure 3, then the first two orders would form batch one (in orange) and batch two would be composed of Order 3 and Order 4 (in purple). The circles represent the movement of a picker showing that only two cycles have to be traversed. The number of circles traversed will indicate the walking distance to compare different metrics.

Picking line location example

Locations:	1	2	3	4	5	6	7	8	9	10
Order 1		◓	⊕				◈	□	◈	
Order 2				⊗	△		◈			
Order 3			⊕		△	◇	◈	□		⬠
Order 4	○	◓				◇	◈			

Table 1: *The picking locations of the four orders.*

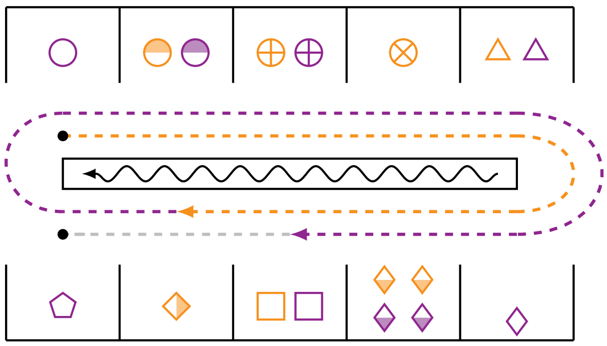


Figure 3: *Schematic representation of order batching applied to a picking line.*

In the remainder of this section, the stops metric, the non-identical stops metric and the stops ratio that approximate the compatibility between orders are introduced. Additionally, a small example on the basis of Table 1 will be provided to explain each metric.

The stops metric in matrix T with general element t_{ij} is a location-based batching rule. It results in the orders with the smallest total number of picking locations forming a batch. The logic of this metric combines the seed selection rule of choosing the smallest number of non-overlapping picking locations introduced by Elsayed & Stern [16] with the smallest number of additional picking locations rule proposed by Ho & Tseng [27]. The number of picking locations of each order when combined with every other order is calculated first. This results in the total number of picking locations that a picker has to visit if those two orders are batched. The orders with the smallest number of combined picking locations are then selected to form a batch. Therefore, each location at which a picker has to stop to pick for one of the orders is counted. On the contrary, counting all locations a picker passes not stopping to collect items for any order may result in the combination of orders which only have a small number of picking locations in common. This option is thus not further investigated.

The stops metric t_{ij}, with the sets $\mathcal{S}_i$ and $\mathcal{S}_j$, containing all stops for orders i and j may be calculated as

$$t_{ij} = |\mathcal{S}_i| + |\mathcal{S}_j| - |\mathcal{S}_i \cap \mathcal{S}_j|. \tag{4}$$

Batching Order 1 with $\mathcal{S}_1 = \{2, 3, 7, 8, 9\}$ and Order 2 with $\mathcal{S}_2 = \{4, 5, 7\}$ results in the stops entry $t_{12} = 5 + 3 - 1 = 7$. Calculating the stops metric for all orders in the example results in a symmetric matrix as illustrated in Table 2a. Combining the orders with the exact solution approach, as described in Section 3.2, batches the orders with the smallest number of picking locations. Therefore, the first batch could contain Order 1 and 3 and the second batch could be composed of Order 2 and 4 in this example.

An additional location-based rule is the non-identical stops metric in matrix N with general element n_{ij}. Batches are formed by combining orders with the smallest number of non-identical picking locations as a result of the application of this metric. Ho & Tseng [27] proposed the greatest number of identical picking locations rule. The logic of n_{ij} is contrary to their rule. The picking locations for combining each order with every other order are

$$T = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & - & 7 & 8 & 7 \\ 2 & 7 & - & 7 & 6 \\ 3 & 8 & 7 & - & 8 \\ 4 & 7 & 6 & 8 & - \end{array}$$

(a) The T matrix.

$$N = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & - & 6 & 5 & 5 \\ 2 & 6 & - & 5 & 5 \\ 3 & 5 & 5 & - & 6 \\ 4 & 5 & 5 & 6 & - \end{array}$$

(b) The N matrix.

$$R = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & - & 0.86 & 0.63 & 0.71 \\ 2 & 0.86 & - & 0.71 & 0.83 \\ 3 & 0.63 & 0.71 & - & 0.75 \\ 4 & 0.71 & 0.83 & 0.75 & - \end{array}$$

(c) The R matrix.

Table 2: *The T, N and R matrices for the picking line example.*

obtained in the first step. However, the order with the smallest number of non-identical picking locations between orders becomes the accompanying order in this case. Therefore, each location where a picker stops to collect for only one of the orders is counted to identify non-identical picking locations.

The non-identical stops matrix N, with element n_{ij}, can be calculated as

$$n_{ij} = |\mathcal{S}_i| + |\mathcal{S}_j| - 2\,|\mathcal{S}_i \cap \mathcal{S}_j|. \tag{5}$$

In the example Order 1 with stops $\mathcal{S}_1 = \{2,3,7,8,9\}$ and Order 3 with $\mathcal{S}_3 = \{3,5,6,7,8,10\}$ result in the non-identical stops entry $n_{13} = 5+6-6 = 5$. Table 2b results from calculating the non-identical stops metric for all orders in the example. Using the exact solution approach the combinations of orders with the smallest number of non-identical picking locations could result in the combination of Order 1 and 4 as well as Order 2 and 3.

The stops ratio metric in matrix R with general element r_{ij} combines the non-identical stops with the combined picking locations of the orders. This metric is also based on the location of items in an order. The non-identical stops metric is divided by the stops metric and thus results in the stops ratio metric. The closer the stops ratio is to zero, the lower the incompatibility of orders in terms of walking distance.

Formulating the stops ratio metric can be achieved by dividing n_{ij} by t_{ij}. Therefore, the stops ratio is calculated as

$$r_{ij} = \frac{n_{ij}}{t_{ij}}. \tag{6}$$

For batching Orders 1 and 4 with stops $\mathcal{S}_1 = \{2,3,7,8,9\}$ and $\mathcal{S}_4 = \{1,2,6,7\}$, the stops ratio metric $r_{14} = 5/7$ would result in 0.71. In Table 2c all stops ratios for all orders from the example are illustrated. Order 1 and 4 as well as Order 2 and 3 could be combined when applying the exact solution formulation.

Even in this small example as described in Table 1 the picking location metrics measure the incompatibility of orders in terms of distance differently and thus suggest different batch combinations ($1,2$ and $3,4$ in FIFO, $1,3$ and $2,4$ in T, $1,4$ and $2,3$ in N as well as R). With bigger data sets the differences in using different metrics become more evident. Furthermore, picking lines with a size of up to 2 000 orders have over a million different possible combinations in determining the best match between two. Applying an exact solution approach would make checking all possible combinations very time consuming. Therefore, heuristic and metaheuristic solution approaches that reduce computational time are described in the following section.

5 Heuristic solution approaches

After the orders have been measured using one of the proposed picking location metrics, different algorithms to combine the orders in batches of size two for a unidirectional cyclical picking line in the Retailer's DC are described in this section. Four greedy heuristics as well as six metaheuristics are introduced.

5.1 Greedy heuristics

The four different variations of greedy heuristics include a greedy top-down, a greedy bottom-up, a greedy random and a greedy smallest entry approach. All four variations search through the symmetric matrices generated by applying the stop metrics for minimum entries.

The greedy top-down (GTD) starts searching the matrix from the first row until it reaches the last row in a top-down fashion. Thereby, the tuple k_{ij} corresponding to the minimum entry m_{kj} of row k is recorded in a set $\mathcal{G}$ to indicate that orders k and j are batched. Then both row k and column j are removed. This process continues until all rows and columns of the matrix are eliminated. At this time set $\mathcal{G}$ has the cardinality n, where n is the size of the problem. The greedy bottom-up (GBU) and the greedy random (GR) progress in a similar way. The GBU starts at the last row and searches until it reaches the first row. The GR searches the rows in a random sequence. This is displayed in Algorithm 1. Furthermore, a greedy algorithm that globally searches for the smallest entry (GS) in the matrices was developed. This algorithm then eliminates rows and columns in the same way as the other greedy heuristics. There is no difference between searching rows or columns, since the picking location metrics generate symmetric matrices [13].

Algorithm 1: Greedy random heuristic (GR)

Input : A picking location metric M consisting of a $n \times n$ matrix with entries m_{ij}, an empty solution set $\mathcal{G}$

Output: The solution set $\mathcal{G}$ as a list of batched orders

```
1 $\mathcal{G} \leftarrow \emptyset$;
2 while $| \mathcal{G} | < n/2$ do
3     $k \leftarrow$ random row from $M$;
4     $m_{kq} = \min_j [m_{kj}]$;
5     $\mathcal{G} \leftarrow \mathcal{G} \cup (k, q)$;
6     Remove row and column $k$ from $M$;
7     Remove row and column $q$ from $M$;
8 end
9 Return $\mathcal{G}$;
```

5.2 Metaheuristics

Only single-solution based methods have been chosen for the six metaheuristics, since the objective is to generate batches of size two. These methods start with a single initial solution, then moving away from it to describe a trajectory in the search space [4]. The algorithms are described in this section, while the parameters for each metaheuristic controlling the trade-off between intensification and diversification have been calibrated specifically for the problem statement of the unidirectional cyclical picking line in the Retailer's DC. For each metaheuristic starting from parameter values found in literature the number of configurations used to calibrate differ between algorithms, due to their ability to find a solution within a limited number of iterations that allows for an overall optimisation of the picking process. Including a range in number of orders as well as SKUs 12 sample picking lines are used to determine the configuration for each metaheuristic that provides the lowest number of total cycles traversed.

The iterated local search (ILS) generates a new starting solution for the following iteration by perturbing the local optimum found at the current iteration. Therefore, the local search procedure is not repeatedly applied to a randomly generated starting solution, but to the best solution found in the previous iteration. The underlying assumption is that the perturbation mechanism, that is the key feature of the ILS, is able to provide a solution that lies in the basin of attraction of a better local optimum [5]. The ILS and its framework was first defined by Stützle [52]. In Algorithm 2 the pseudocode of the ILS for the OBP is displayed. The acceptance criterion incorporates a restart of the search after a predefined non-acceptance counter thus incorporating a simple form of history. By means of parameter calibration 12 configurations with different termination criterion and non-acceptance counters under limited time have been tested on the sample picking lines. The lowest numbers of total cycles traversed were found incorporating the configuration with a termination criterion t_I of 5 and a non-acceptance counter a_I of 3.

Algorithm 2: Iterated local search (ILS)

Input : An initial solution s_a, a cost function $c(\cdot)$ as well as a non-acceptance counter a_I and a termination criterion t_I

Output: The best solution s as a list of batched orders

```
1  $a_I, t_I \leftarrow 0$;
2  $s_a \leftarrow$ initiate a starting solution;
3  $s \leftarrow s_a$;
4  $s^* \leftarrow$ perform a local search on $s$;
5  while the termination criterion $t_I$ is not met do
6      $s^{'} \leftarrow$ perturb $s^*$;
7      $s^{*'} \leftarrow$ perform a local search $s^{'}$;
8      if $c(s^{*'}) \leq c(s^*)$ then
9          $s^* \leftarrow s^{*'}$;
10         $a_I = 0$;
11     else
12         $a_I = a_I + 1$;
13         if non-acceptance counter $a_I$ is not met then
14             $t_I = t_I + 1$;
15         else
16             $s^* \leftarrow$ restart search;
17             $t_I = 0$;
18         end
19     end
20     $s \leftarrow s^*$;
21 end
22 Return $s$;
```

The variable neighbourhood search (VNS) explores the solution space by dynamically changing neighbourhoods around a given solution. After an initial solution is introduced, the main cycle of the VNS consists of the steps shaking, local search and move. In the shaking step a solution $s^{'}$ is randomly selected in the h^{th} neighbourhood of the current solution. If the local search produces a better solution $s^{*'}$ than s, the solution is updated and the algorithm stays in the first neighbourhood. Otherwise, the algorithm moves to explore the next neighbourhood [5]. Mladenović & Hansen [40] introduced the structure of this algorithm. The pseudocode displayed in Algorithm 3, applies the VNS to the OBP. In this study three neighbourhoods are defined after testing different configurations in numerical experiments. In the first neighbourhood the order with the highest cost according to the picking location metric is swapped with the lowest, then the second highest and lowest are swapped as well as the third highest and lowest are interchanged respectively. The termination criterion t_V has been set to 5 through parameter calibration including 6 different configurations of termination criterion tested on the sample picking lines under time restriction.

Algorithm 3: Variable neighbourhood search (VNS)

```
Input  : An initial solution $s_a$, a set of neighbourhood structures $\mathcal{H}$ with $h = 1, 2, 3$, a cost
         function $c(\cdot)$ as well as a termination criterion $t_V$
Output: The best solution $s$ as a list of batched orders
$t_V \leftarrow 0$;
$\mathcal{H} \leftarrow$ generate a set of neighbourhood structures with $h = 1, 2, 3$;
$s_a \leftarrow$ initiate a starting solution;
$s \leftarrow s_a$;
while the termination criterion $t_V$ is not met do
    $h \leftarrow 1$;
    while in one of the defined neighbourhood structures do
        $s^{'} \leftarrow$ select a random solution in the $h^{th}$ neighbourhood $\mathcal{H}_h(s)$ of $s$;
        $s^{*'} \leftarrow$ perform a local search on $s^{'}$;
        if $c(s^{*'}) \leq c(s)$ then
            $s \leftarrow s^{*'}$;
            $h = 1$;
        else
            $h = h + 1$;
        end
    end
    $t_V = t_V + 1$;
end
Return $s$;
```

The variable neighbourhood descent (VND) is a variation of the variable neighbourhood search. The VND is a deterministic version of the VNS excluding the shaking step [53]. The search steps of the VND are illustrated in Algorithm 4. The neighbourhood structure stays the same as in the VNS apart from the termination criterion t_D that is set to 50 through parameter calibration by testing 10 configurations with varying termination criterion on the sample picking lines under limited time. The VND is able to find a feasible solution faster than the VNS in this study, thus more configurations can be tested.

Algorithm 4: Variable neighbourhood descent (VND)

Input : An initial solution s_a, a set of neighbourhood structures $\mathcal{H}$ with $h = 1, 2, 3$, a cost function $c(\cdot)$ as well as a termination criterion t_D

Output: The best solution s as a list of batched orders

```
$t_D \leftarrow 0$;
$\mathcal{H} \leftarrow$ generate a set of neighbourhood structures with $h = 1, 2, 3$;
$h \leftarrow 1$;
$s_a \leftarrow$ initiate a starting solution;
$s \leftarrow s_a$;
while in one of the neighbourhood structures and the termination criterion $t_D$ is not met do
    $s' \leftarrow$ perform a local search on $s$ in the $h^{th}$ neighbourhood $\mathcal{H}_h(s)$;
    if $c(s') \leq c(s)$ then
        $s \leftarrow s'$;
        $h = 1$;
    else
        $h = h + 1$;
    end
end
$t_D = t_D + 1$;
Return $s$;
```

In a tabu search (TS) the history of the search is used to escape from local optima as well as to implement an exploitative strategy. A defined number of previously encountered solutions is recorded in a tabu list and thus forbidden to be revisited. The list can be described as a short term memory that prevents the algorithm from endless cycling and forces the search to explore different solution spaces. The length of the tabu list thus controls the memory of the search process. Glover [21] first introduced the tabu search algorithm. The mechanisms of the human memory inspired the main characteristic of this algorithm. Its application to the OBP is illustrated in Algorithm 5. The configuration with its tabu list length l set to 0.8 relative to the size of the problem and its termination criterion t_T set to 3 provides the lowest number of total cycles traversed when testing 32 configurations with different length of tabu lists and termination criterion on the sample picking lines under time restriction.

Inspiration for the simulated annealing (SA) algorithm comes from the annealing technique used by metallurgists. Material is heated up to a high temperature and then lowered down slowly to obtain a well ordered state of minimum energy. Applying this technique, the objective function of the optimisation problem is minimised similar to the energy of the material [5]. The starting solution and the annealing scheme for the temperature decrease are initiated. At each iteration, a new solution is accepted with a certain probability determined by the Metropolis criterion [1]. The SA algorithm was introduced by Kirkpatrick *et al.* [29] and by Černý [7] independently. In Algorithm 6 the structure of the SA is illustrated. The annealing scheme is crucial to the performance of the algorithm [14]. Therefore, three different approaches in determining the annealing scheme have been analysed for this application. A constant lowering of the temperature, dynamically changing the temperature according to the acceptance of a number of perturbations as well as restarting to the initial solution after a number of non-acceptances of the new solution have been tested. Thereby, re-heating is incorporated in two variations in the second and

Algorithm 5: Tabu search (TS)

Input : An initial solution s_a, a neighbourhood $\mathcal{N}(s)$, a tabu list length l, a cost function $c(\cdot)$ as well as a termination criterion t_T

Output: The best solution s as a list of batched orders

```
$t_T \leftarrow 0$;
TabuList $\leftarrow \emptyset$;
$s_a \leftarrow$ initiate a starting solution;
$s \leftarrow s_a$;
while the termination criterion $t_T$ is not met do
    $s^{'} \leftarrow$ select the best solution in $\mathcal{N}(s) \setminus$ TabuList;
    if $c(s^{'}) \leq c(s)$ then
        $s \leftarrow s^{'}$;
        Update TabuList;
        $t_T = 0$;
    else
        $t_T = t_T + 1$;
    end
end
Return $s$;
```

third configuration of the algorithm. Numerical experiments showed that using the acceptance of a perturbation performs best in the application of this study. Additionally, the parameters initial temperature t_a was set to 1.0, the alpha value α to 0.9, the acceptance counter a_S to 12 as well as the termination criterion t_S to 3 through fine-tuning initial temperature, alpha value, acceptance counter and termination criterion in 120 different configurations under limited time.

The great deluge (GD) algorithm is a variation of the SA algorithm. It differs in the acceptance of solutions and is easier to apply, since it only has one parameter that needs to be determined. The metaphor this algorithm uses is that of a hiker that tries to keep her feet dry while visiting the peaks of an explored area under a slowly rising water level [14]. Dueck [15] proposed this algorithm and it is applied to the OBP as illustrated in the pseudocode of Algorithm 7. The parameters have been calibrated to a rain speed r of 0.5 as well as a termination criterion t_G that is set to 5 with 40 different configurations for rain speed and termination criterion tested on the sample picking lines under time restriction.

6 Experimental results

The proposed picking location metrics that approximate the distance needed to pick orders before pickers are routed and thus aims at minimising the picking incompatibility between orders as well as the algorithms that combine orders in batches on the basis of these metrics are tested on 50 sample picking lines. Thereby, the best combinations of metric and algorithm are determined and the best combination to introduce order batching to a unidirectional cyclical picking line is identified. Information about the data set, the experiment and the statistical analysis of the results are provided in the following sections.

Algorithm 6: Simulated annealing (SA)

Input : An initial solution s_a, a neighbourhood $\mathcal{N}(s)$, an initial temperature t_a, a alpha value α, a cost function $c(\cdot)$ as well as an acceptance counter a_S and a termination criterion t_S

Output: The best solution s as a list of batched orders

```
1  $a_S, t_S \leftarrow 0$;
2  $T \leftarrow t_a$;
3  $s_a \leftarrow$ initiate a starting solution;
4  $s \leftarrow s_a$;
5  while the termination criterion $t_S$ is not met do
6      $s^{'} \leftarrow$ randomly select a solution in $\mathcal{N}(s)$;
7      if $c(s^{'}) \leq c(s)$ or accept $s^{'}$ with probability $\exp(-\frac{c(s^{'})-c(s)}{T})$ then
8          $s \leftarrow s^{'}$;
9          $a_S = a_S + 1$;
10         $t_S = 0$;
11     else
12         $a_S = 0$;
13         $t_S = t_S + 1$;
14     end
15     if the thermodynamic equilibrium is reached through acceptance counter $a_S$ then
16         $T = T * \alpha$;
17         $a_S = 0$;
18     end
19 end
20 Return $s$;
```

Algorithm 7: Great deluge (GD)

Input : An initial solution s_a, a neighbourhood $\mathcal{N}(s)$, a rain speed r, a cost function $c(\cdot)$ as well as a termination criterion t_G

Output: The best solution s as a list of batched orders

```
1  $t_G \leftarrow 0$;
2  $s_a \leftarrow$ initiate a starting solution;
3  $s \leftarrow s_a$;
4  $w \leftarrow c(s)$;
5  while the termination criterion $t_G$ is not met do
6      $s^{'} \leftarrow$ randomly select a solution in $\mathcal{N}(s)$;
7      if $c(s) \leq c(s^{'})$ then
8          $t_G = t_G + 1$;
9      end
10     if $c(s^{'}) < w$ then
11         $s \leftarrow s^{'}$;
12         $w \leftarrow$ recalculate with $w - \frac{(w-c(s^{'}))}{r}$;
13         $t_G = 0$;
14     end
15 end
16 Return $s$;
```

All algorithms were implemented in Python 3.6 [46]. These implementations were run on an Intel (R) Core (TM) i7-7700 CPU at 3.60 GHz, 4 Cores, 8 Logical Processors running Microsoft Windows 10 Enterprise 2016 LTSB [39]. The integer programmes were solved by means of Lingo 11.0.1.3. [32]. Additionally, IBM SPSS Statistics 25 [28] was used for the statistical analysis of the generated results.

6.1 Data

A set of real life historical data were obtained from the Retailer. These data were made publicly available by Matthews and Visagie and can be accessed online [37]. For reporting purposes, 50 sample picking lines were randomly selected and divided into large data sets with more than 1 000 orders, medium data sets with 400 – 600 orders and small data sets with less than 100 orders. Within these data sets the picking lines are subdivided with respect to the number of SKUs into picking lines with a large, medium or small number of SKUs. The variety in size and location of the different retail stores together with the seasonality of the product portfolio lead to a set of non-uniform orders that is processed by the DC on a daily basis.

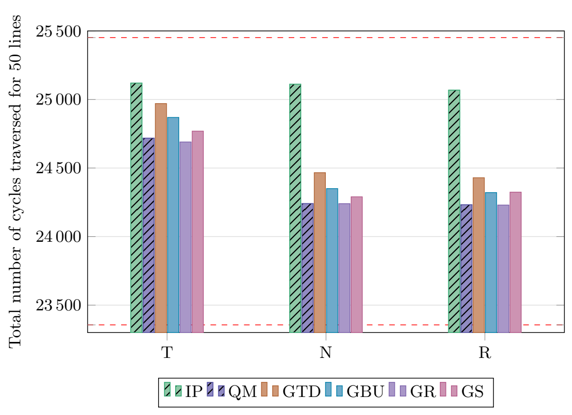
6.2 Computational results

The integer programming formulation (IP) in (1) – (3) was employed to combine orders based on the picking location metrics. The symmetric matrix resulting from applying one of the metrics was turned into an upper triangular matrix already containing all information needed to solve the problem more efficiently. Nevertheless, most test cases had to be stopped after the time limit proposed by Nicolas *et al.* [42] of 30 minutes. An average computational time of over 11 minutes per sample picking line without a guaranteed solution within 30 minutes is not feasible in a real life application as planning of the lines need to be completed in seconds. The picking lines that can be solved to optimality do not justify the reduction in walking distance, since only the picking location metric is solved optimally. Furthermore, solving to optimality is computationally expensive, thus implementing the IP on a stronger solver to get a guaranteed solution does not yield a return on time investment.

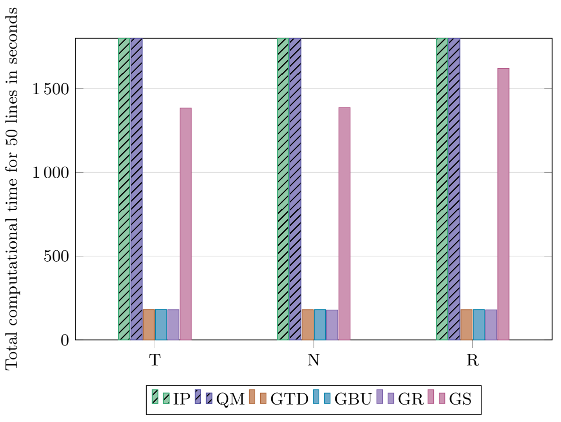
Applying the quick match algorithm (QM), seems to increase performance since the total number of cycles traversed is lower for QM than for IP as illustrated in Figure 4a. Nevertheless, QM is also not able to guarantee a solution within the set time frame and has to be terminated before exceeding the time limit as well.

If an exhaustive optimisation of the picking system is aimed at, two additional decision tiers have to be solved for each picking line determining where a SKU should be placed on the picking line as well as on which picking line a SKU should be picked. In consultation with the management of the Retailer a time frame of about 30 seconds per picking line for combining all orders to batches would be feasible. Thereby, line managers do not have to wait longer than a minute to solve all three decision tiers for a picking line.

Besides the time constraint, an upper and a lower bound help to evaluate the performance of metric and algorithm combination in terms of solution quality. The worst case would be to not introduce order batching. The upper bound would thus be at 46 711 total cycles traversed. The best case would be to reduce the walking distance by 50%, thus resulting in an absolute lower bound of 23 356 total cycles traversed. Therefore, if no batching was introduced the cycles traversed could be cut in two and put next to each other. However, this would not result in a feasible solution as orders that overlap do not have the same starting and end location. Linking up the starting and end locations would result in a longer walking distance than the application of one of the picking location metrics. Additionally, the benchmark incorporating a FIFO approach as discussed in Section 4 will give a guideline of 25 451 cycles to evaluate the performance of the combinations compared against a random batching approach. The lower bound as well as the FIFO benchmark will be indicated by a red dashed line in the following graphs.



(a) Cycles traversed per greedy.



(b) Computational time per greedy.

Figure 4: *Total number of cycles traversed and computational time per greedy heuristic per metric.*

Greedy heuristics choose the next best combination of orders in a greedy fashion and thus speed up performance. All four greedy heuristics, namely the GTD, GBU, GR and GS, are tested per metric on the 50 sample picking lines. For five test runs, the GR performs best in terms of minimum number of cycles traversed on all 50 sample picking lines. It also performs slightly better than the QM. This is depicted in Figure 4a. For 5 configurations, the standard deviation in total number of cycles traversed for the stops metric is 28.96, for the non-identical stops is 33.07 and for the stops ratio is 18.62. Calculating the coefficient of variation by division of the mean shows that all metrics have a low variation in relation to their means with stops at 0.12%, non-identical stops at 0.14% and stops ratio at 0.08%. As illustrated in Figure 4b, all greedy heuristics have a comparable total computational time of approximately 180 seconds, because of the similar mechanism of the algorithms.

Metaheuristics are higher level heuristics that do not have to be adapted deeply to a specific problem, but that are able to escape local optima. Therefore, metaheuristics might get even closer to the optimal solution of the minimum number of cycles traversed within the predetermined time frame. All six metaheuristics ILS, VNS, VND, TS, SA and GD are combined with the three picking location metrics. Test runs showed that using GR to generate an initial solution performs better than initialising a random solution. Therefore, a hybrid of GR with each metaheuristic is analysed. Each algorithm was run five times with the same greedy starting solution for each run. However, the computational time restrictions resulted in the metaheuristics only being able to improve on the number of cycles traversed in combination with the stops metric. This is depicted in Figure 5. For the stops metric, the GD followed by the SA shows slightly better results in terms of minimum number of total cycles traversed and computational time than the other four metaheuristics.

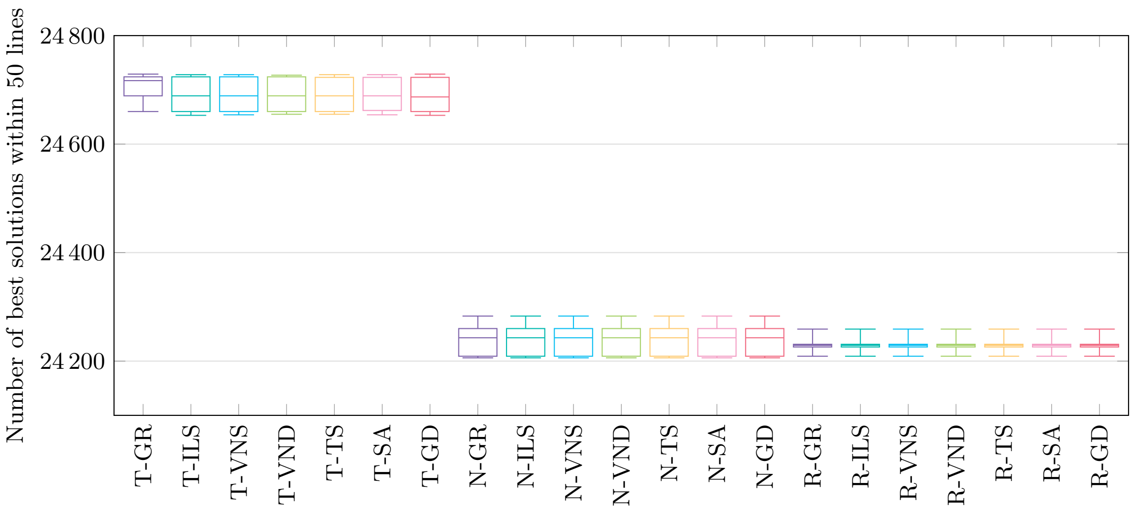


Figure 5: *Box and whisker plots of the total cycles traversed per metaheuristic per metric.*

All computational times per algorithm per metric for all 50 sample picking lines are illustrated in Figure 6. For all metrics the GR is the fastest, since all other metaheuristics are hybrids including the time of the GR algorithm. Nevertheless, the GR is followed by the GD- and SA-hybrid, while the VNS followed by the ILS and TS take the longest to get to a solution.

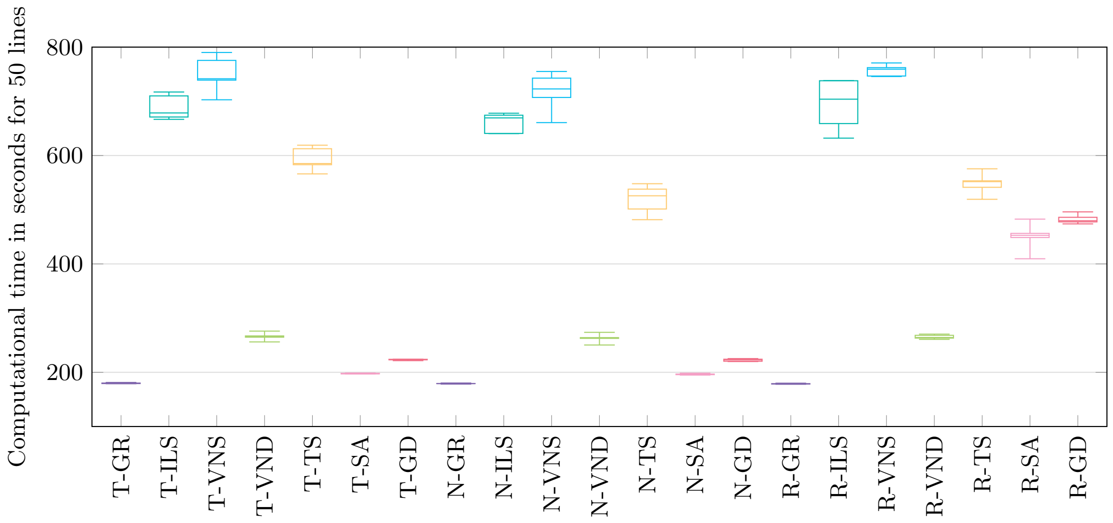


Figure 6: *Box and whisker plots of the computational time per metaheuristic per metric.*

The hybridisation between metaheuristics may also lead to a better solution quality and an improvement in computational time. Therefore, the combination of metaheuristics has been tested for example picking lines, but has shown no constant improvement under the time restriction.

Metaheuristics could not improve the solution quality under the time restriction for two out of three picking location metrics. This raises the question whether the solution found by GR is close to optimal (and thus few improvements exist) or the metaheuristics are incapable of finding improvements on a poor solution found by the GR. To evaluate this question a sample picking line in which the IP was able to find the optimal solution within the time frame is analysed. For the stops metric possible combinations on a sample picking line with 1 336 orders and 55 SKUs is investigated. The IP is able to generate the best solution of 620 cycles that have to be traversed to pick all orders in 137.73 seconds. While the GR algorithms is able to get to 631 cycles in approximately 5 seconds, the GD- as well as the SA-hybrid are able to reach 629 cycles in almost the same time. The solution are within 98.55% very close to the best solution. These results indicate that the more computational time a metaheuristics is allowed to use, the closer it can get to the best solution. However, if the walking distance could be reduced by 50% the lowest bound would be at 594 cycles. A sample picking line with 1 356 orders and 51 SKUs is analysed for the non-identical stops metric. It takes 169.72 seconds for the IP to generate the best solution of 599 cycles that have to be traversed to collect all orders. None of the metaheuristics is able to improve on the GR solution within the given time restrictions. Nevertheless, the GR solution is within 99.50% the best solution for this metric, while the lower bound for this example would be at 573 cycles. For the stops ratio metric a sample picking line with 1 354 orders and 56 SKUs is investigated. The IP generates the optimal solution of 583 cycles in 118.91 seconds that have to be traversed. No metaheuristic is able to improve on the GR solution within the predetermined time restriction. However, the solution is within 99.49% also very close to the best solution and the lower bound for this example is at 555 cycles.

Figure 5 suggests the combination of the stops metric with GR-GD-hybrid, the non-identical stops metric with GR as well as the stops ratio metric with GR, because of the minimum number of total cycles traversed generated by these combinations. This is supported by the lowest total computational times for these algorithms as illustrated in Figure 6.

6.3 Statistical analysis

A regression analysis per location metric supports the application of location-based metrics as approximations for walking distance before picker routing. The correlation between the objective value per metric and the number of cycles traversed is evaluated. This results in $R^2 = 0.783$ for stops, $R^2 = 0.784$ for non-identical stops and $R^2 = 0.776$ for stops ratio. This shows a strong correlation between all metrics and final walking distance. While the approximation purpose of each metric is validated, the regression analysis does not provide a basis for comparison of metrics, since the objective values of all metrics are expressed in different units.

Further inferential statistics are necessary to investigate the influence of metrics and algorithms on the solution. Chiarandini *et al.* [8] propose either a univariate model for analysing solution quality and computational time or a bivariate model combining both measurements. This analysis will focus on the solution quality in terms of total number of cycles traversed for all 50 picking lines, since the computational time has shown to be suitable for an application in the operations of the Retailer's DC. Therefore, A Welch-ANOVA with a Games Howell *post-hoc* test, that is robust to the equality of means, is used to analyse if there is a statistical difference by applying the three picking location metrics. Furthermore, a two-way ANOVA investigates the additional influence of the algorithm [49].

In Table 3, the Welch-ANOVA is statistically significant ($F(2, 102) = 3\ 419.293, p = 3.3682E\text{-}94$) emphasising the influence of each metric on the number of cycles traversed. The *post-hoc* Games Howell test shows that the difference in cycles traversed is significantly lower when using N ($24\ 240 \pm 29.993, p = 5.1011E\text{-}9$) and R ($24\ 230 \pm 16.90, p = 5.1012E\text{-}9$) compared to T ($24\ 693 \pm 30.937$) [30, 50]. Nevertheless, the two-way ANOVA reveals no significant influence on the number of cycles traversed through the interaction between metrics and algorithms ($F(12, 84) = 0.047, p = 1.00$) [31, 51].

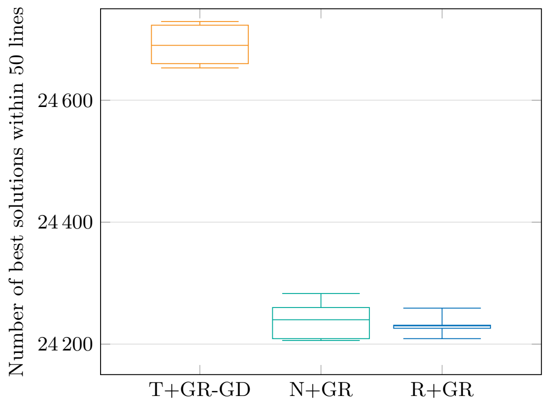
	Sum of squares	*df*	**Mean square**	*F*	*p*
Welch-ANOVA					
Between metrics	4 883 348.648	2	2 441 674.324	3 419.293	$3.3682E - 94$**
Within metrics	72 836.914	102	714.087		
Two-way ANOVA					
Metrics	4 883 348.648	2	2 441 674.324	2 844.117	$6.9709E - 78$**
Algorithms	241.962	6	40.327	0.047	1.00
Metrics × algorithms	480.952	12	40.079	0.047	1.00
Error	72 114.00	84	858.50		

Note: Two asterisks indicate significance at the 5% level or below.

Table 3: *Welch-ANOVA on metrics and two-way ANOVA on metrics and algorithms.*

Descriptive statistics are used to compare the best combinations of picking location metrics and their corresponding algorithms based on the homogeneous unit of total number of cycles traversed for all 50 sample picking lines. These combinations are depicted in Figure 4a. In general, N and GR as well as R and GR combined perform better in terms of minimum number of cycles traversed when compared to the T and GR-GD hybrid combination.

Moreover, stops ratio shows lower measures of central tendency than non-identical stops with a mean of 24 230 compared to 24 240 as illustrated in Table 4b. Comparing the measurements of variability between R and N, the range between maximum and minimum total number of cycles traversed is 32.5% smaller for R. Furthermore, the standard deviation of R is almost half the size of N resulting in smaller variance and thus providing a metric-algorithm combination that seems more stable than N. Therefore, this comparison suggests choosing the combination of stops ratio metric and greedy random heuristic to introduce order batching to a unidirectional cyclical picking line.



(a) Box-whisker plot for each combination.

	T+GR-GD	N+GR	R+GR
Mean	24 693	24 240	24 230
Median	24 687	24 243	24 229
Range	76	77	52
Std.Dev.	30.937	29.993	16.900
Variance	957.104	899.558	285.600

(b) Descriptive statistics.

Table 4: *Comparison between best combinations of metric and algorithm.*

7 Conclusion

In this paper order batching has been introduced to a unidirectional cyclical picking line as deployed in the layout of a South African retailer's DC. Three location-based order-to-route closeness metrics have been proposed to approximate the walking distance before picker routing and thus identify compatible orders. The picking location metrics include stops counting all locations a picker has to stop and pick for an order, non-identical stops counting the locations a picker has to stop to pick for only one order and stops ratio describing the ratio between non-identical and combined stops. The assignment problem of grouping orders in batches of size two has been solved with exact, greedy and metaheuristic solution approaches. The exact solution approaches take too much computational time to allow for an integrated optimisation approach. Therefore, four greedy heuristics including a top-down, bottom-up, random and smallest entry search approach as well as six metaheuristics namely an iterated local search, a variable neighbourhood search, a variable neighbourhood descent, a tabu search, a simulated annealing and a great deluge algorithm have been applied to 50 sample picking lines. The picking lines have been recorded from real life data and vary in size by number of orders and SKUs. The algorithms terminate after a reasonable computational time restriction to allow for a real life application. All metrics and algorithms have been tested in different combinations to identify the best combinations reducing the total number of cycles traversed. This results in the best combinations of stops and greedy random-great deluge, non-identical stops and greedy random as well as stops ratio and greedy random.

These best combinations are then compared in terms of minimum number of cycles traversed for all 50 picking lines. The stops ratio and greedy combination shows the lowest average of 24 230 cycles within the five test runs. Therefore, this combination is recommended to be applied if order batching is introduced to a unidirectional cyclical picking line. Compared to the FIFO benchmark of 25 451 cycles through applying random batching, this is 4.80% less walking distance. If batching is not introduced to the unidirectional cyclical picking line using this combination, pickers have to walk approximately 48.13% further. Additionally, this combination is only 3.74% higher than the absolute lower bound of reducing 50% of the walking distance. The reduction in walking distance through the combination of stops ratio and greedy random heuristic can be translated into time savings leading to a direct decrease of picking cost.

The analysis of the generated results indicate two findings. Firstly, the metric applied plays a more important role than the algorithm used to group the orders into batches. Secondly, the more information about the location of the items of an order is available, the better the results in terms of minimum number of cycles traversed. Therefore, future work should incorporate more information in the metric about the specific layout of a unidirectional cyclical picking. This could help to find an even better approximation of walking distance and thus get even closer to the lower bound of saving 50% of walking distance by picking two orders at a time.

References

[1] Aarts E & Korst J, 1988, *Simulated annealing and Boltzmann machines*, John Wiley and Sons Inc., New York.

[2] Albareda-Sambola M, Alonso-Ayuso A, Molina E & De Blas CS, 2009, *Variable neighborhood search for order batching in a warehouse*, Asia-Pacific Journal of Operational Research, **26(05)**, pp. 655–683.

[3] Bartholdi JJ & Hackman ST, 2016, *Warehouse & Distribution Science: Release 0.97*, Supply Chain and Logistics Institute, Available from https://www.warehouse-science.com/book/index.html.

[4] Blum C & Roli A, 2003, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM computing surveys (CSUR), **35(3)**, pp. 268–308.

[5] BoussaïD I, Lepagnot J & Siarry P, 2013, *A survey on optimization metaheuristics*, Information Sciences, **237**, pp. 82–117.

[6] Bozer YA & Kile JW, 2008, *Order batching in walk-and-pick order picking systems*, International Journal of Production Research, **46(7)**, pp. 1887–1909.

[7] Černỳ V, 1985, *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*, Journal of optimization theory and applications, **45(1)**, pp. 41–51.

[8] Chiarandini M, Paquete L, Preuss M & Ridge E, 2007, *Experiments on metaheuristics: Methodological overview and open issues*, Technical Report DMF-2007-03-003, Available from https://pdfs.semanticscholar.org/9fcd/f48d4ae1770a6c653e33caee25aeb109b44d.pdf.

[9] Clarke G & Wright JW, 1964, *Scheduling of vehicles from a central depot to a number of delivery points*, Operations research, **12(4)**, pp. 568–581.

[10] Dallari F, Marchet G & Melacini M, 2009, *Design of order picking system*, The International Journal of Advanced Manufacturing Technology, **42(1)**, pp. 1–12.

[11] De Koster M, Van der Poort ES & Wolters M, 1999, *Efficient orderbatching methods in warehouses*, International Journal of Production Research, **37(7)**, pp. 1479–1504.

[12] De Koster R, Le-Duc T & Roodbergen KJ, 2007, *Design and control of warehouse order picking: A literature review*, European Journal of Operational Research, **182(2)**, pp. 481–501.

[13] De Villiers AP, 2012, *Minimising the total travel distance to pick orders on a unidirectional picking line.*

[14] Dréo J, Pétrowski A, Siarry P & Taillard E, 2006, *Metaheuristics for hard optimization: methods and case studies*, Springer Science & Business Media, Berlin.

[15] Dueck G, 1993, *New optimization heuristics*, Journal of Computational physics, **104(1)**, pp. 86–92.

[16] Elsayed EA & Stern RG, 1983, *Computerized algorithms for order processing in automated warehousing systems*, The International Journal of Production Research, **21(4)**, pp. 579–586.

[17] Elsayed EA & Unal O, 1989, *Order batching algorithms and travel-time estimation for automated storage/retrieval systems*, The International Journal of Production Research, **27(7)**, pp. 1097–1114.

[18] Frazelle E, 2002, *World-class warehousing and material handling*, volume 1, McGraw-Hill New York.

[19] Gademann N & Velde S, 2005, *Order batching to minimize total travel time in a parallel-aisle warehouse*, Institute of Industrial Engineers Transactions, **37(1)**, pp. 63–75.

[20] Gibson DR & Sharp GP, 1992, *Order batching procedures*, European Journal of Operational Research, **58(1)**, pp. 57–67.

[21] Glover F, 1986, *Future paths for integer programming and links to artificial intelligence*, Computers & operations research, **13(5)**, pp. 533–549.

[22] Gu J, Goetschalckx M & McGinnis LF, 2007, *Research on warehouse operation: A comprehensive review*, European Journal of Operational Research, **177(1)**, pp. 1–21.

[23] Henn S, Koch S, Doerner KF, Strauss C & Wäscher G, 2010, *Metaheuristics for the order batching problem in manual order picking systems*, Business Research, **3(1)**, pp. 82–105.

[24] Henn S, Koch S & Wäscher G, 2012, *Order batching in order picking warehouses: a survey of solution approaches*, pp. 105–137 in *Warehousing in the Global Supply Chain*, pp. 105–137. Springer, London.

[25] Henn S & Wäscher G, 2012, *Tabu search heuristics for the order batching problem in manual order picking systems*, European Journal of Operational Research, **222(3)**, pp. 484–494.

[26] Ho YC, Su TS & Shi ZB, 2008, *Order-batching methods for an order-picking warehouse with two cross aisles*, Computers & Industrial Engineering, **55(2)**, pp. 321–347.

[27] Ho YC & Tseng YY, 2006, *A study on order-batching methods of order-picking in a distribution centre with two cross-aisles*, International Journal of Production Research, **44(17)**, pp. 3391–3417.

[28] IBM, 2018, *Spss version 25*, [Online], [Cited October 8th, 2018], Available from `https://www.ibm.com/analytics/spss-statistics-software`.

[29] Kirkpatrick S, Gelatt CD & Vecchi MP, 1983, *Optimization by simulated annealing*, science, **220(4598)**, pp. 671–680.

[30] Laerd Statistics, 2018, *One-way anova in spss statistics (cont)*, [Online], [Cited October 5th, 2018], Available from `https://statistics.laerd.com/spss-tutorials/one-way-anova-using-spss-statistics-2.php`.

[31] Laerd Statistics, 2018, *Two-way anova in spss statistics (cont)*, [Online], [Cited October 5th, 2018], Available from `https://statistics.laerd.com/spss-tutorials/two-way-anova-using-spss-statistics-2.php`.

[32] Lindo Systems, 2018, *Lingo 11*, [Online], [Cited October 8th, 2018], Available from `https://www.lindo.com/`.

[33] Litvak N & Vlasiou M, 2010, *A survey on performance analysis of warehouse carousel systems*, Statistica Neerlandica, **64(4)**, pp. 401–447.

[34] Matthews J, 2012, *Order sequencing and SKU arrangement on a unidirectional picking line.*

[35] Matthews J, 2015, *Sku assignment in a multiple picking line order picking system.*, Doctoral Dissertation, Stellenbosch: Stellenbosch University.

[36] Matthews J & Visagie SE, 2013, *Order sequencing on a unidirectional cyclical picking line*, European Journal of Operational Research, **231(1)**, pp. 79–87.

[37] Matthews J & Visagie SE, 2014, *Picking line data set a*, [Online], [Cited October 5th, 2018], Available from http://hdl.handle.net/10019.1/86143.

[38] Matusiak M, de Koster R, Kroon L & Saarinen J, 2014, *A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse*, European Journal of Operational Research, **236(3)**, pp. 968–977.

[39] Microsoft, 2018, [Cited October 8th, 2018], Available from https://www.microsoft.com/.

[40] Mladenović N & Hansen P, 1997, *Variable neighborhood search*, Computers & operations research, **24(11)**, pp. 1097–1100.

[41] Muter I & Öncan T, 2015, *An exact solution approach for the order batching problem*, IIE Transactions, **47(7)**, pp. 728–738.

[42] Nicolas L, Yannick F & Ramzi H, 2017, *Optimization of order batching in a picking system with carousels*, IFAC-PapersOnLine, **50(1)**, pp. 1106–1113.

[43] Öncan T, 2015, *Milp formulations and an iterated local search algorithm with tabu thresholding for the order batching problem*, European Journal of Operational Research, **243(1)**, pp. 142–155.

[44] Orlin JB & Lee Y, 1993, *Quickmatch–a very fast algorithm for the assignment problem*, Alfred P. Sloan School of Management.

[45] Pan CH & Liu S, 1995, *A comparative study of order batching algorithms*, Omega, **23(6)**, pp. 691–700.

[46] Python Software Foundation, 2018, *Python 3.6*, [Online], [Cited October 8th, 2018], Available from https://www.python.org/.

[47] Rosenwein M, 1996, *A comparison of heuristics for the problem of batching orders for warehouse selection*, International Journal of Production Research, **34(3)**, pp. 657–664.

[48] Rouwenhorst B, Reuter B, Stockrahm V, van Houtum GJ, Mantel R & Zijm W, 2000, *Warehouse design and control: Framework and literature review*, European Journal of Operational Research, **122(3)**, pp. 515–533.

[49] Sheskin DJ, 2000, *Handbook of parametric and nonparametric statistical procedures*, CRC Press, Florida, United States.

[50] SPSS Tutorials, 2018, *Spss one-way anova tutorial*, [Online], [Cited October 5th, 2018], Available from https://www.spss-tutorials.com/spss-one-way-anova/.

[51] SPSS Tutorials, 2018, *Spss two way anova – basics tutorial*, [Online], [Cited October 5th, 2018], Available from https://www.spss-tutorials.com/spss-two-way-anova-basics-tutorial/.

[52] Stützle T, 1998, *Local search algorithms for combinatorial problems*, Doctoral Dissertation, Darmstadt University of Technology.

[53] Talbi EG, 2009, *Metaheuristics: from design to implementation*, John Wiley & Sons, New Jersey, United States.

[54] Van den Berg JP & Zijm WH, 1999, *Models for warehouse management: Classification and examples*, International Journal of Production Economics, **59(1-3)**, pp. 519–528.

[55] Wäscher G, 2004, *Order picking: a survey of planning problems and methods*, pp. 323–347 in *Supply chain management and reverse logistics*, pp. 323–347. Springer, Heidelberg.

[56] Žulj I, Kramer S & Schneider M, 2018, *A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem*, European Journal of Operational Research, **264(2)**, pp. 653–664.