Order picker routing with product returns and interaction delays

Albert H. Schrotenboer, Susanne Wruck, Kees Jan Roodbergen, Marjolein Veenstra & Arjan S. Dijkstra

To cite this article: Albert H. Schrotenboer, Susanne Wruck, Kees Jan Roodbergen, Marjolein Veenstra & Arjan S. Dijkstra (2017) Order picker routing with product returns and interaction delays, International Journal of Production Research, 55:21, 6394-6406, DOI: 10.1080/00207543.2016.1206982

To link to this article: https://doi.org/10.1080/00207543.2016.1206982

International Journal of Production Research, 2017
Vol. 55, No. 21, 6394–6406, https://doi.org/10.1080/00207543.2016.1206982

Order picker routing with product returns and interaction delays

Albert H. Schrotenboer[a]*, Susanne Wruck[b], Kees Jan Roodbergen[a], Marjolein Veenstra[a] and Arjan S. Dijkstra[a]

[a] *Department of Operations, University of Groningen, Groningen, The Netherlands;* [b] *Bringmeister GmbH, Berlin, Germany*

(*Received 12 January 2016; accepted 13 June 2016*)

E-commerce companies often use manual order-picking systems in their warehouses since these systems can provide the required flexibility and scalability. Manual systems have been widely studied, but the operating policies may require significant changes for e-commerce settings. First, to maintain consumers' loyalty, it is important to maintain delivery reliability even on the busiest days. When the number of order pickers in an area increases, however, more delays due to interactions may occur. For example, travel speed may need to be lowered when order pickers pass each other in narrow aisles. Second, many products sold through e-commerce are returned by consumers. Before these returned products can be sold again, they must be reintegrated in the stock. This paper presents hybrid genetic algorithms to determine routes for simultaneous pickup of products in response to consumers' orders and delivery of returned products to storage locations. Furthermore, interactions between the order pickers are considered in the routing decisions. The developed algorithms use specific warehouse problem characteristics. We identify the mix of pickups and deliveries to realise the highest savings in practice. It is shown that order-picker interactions can be a significant cause for delay and should be accounted for in the routing.

Keywords: warehousing systems; order-picker interaction; pickup and delivery problem; genetic algorithms; routing

1. Introduction

Often flexible and complimentary return options are offered to consumers in e-commerce, which allow consumers to order more products than are actually needed. Consumers then make the actual purchase decision at home after the order is delivered. The remainder of the products is returned to the e-commerce company's warehouse. Return rates of up to 74% occur, as noticed by Mostard, De Koster, and Teunter (2005), and many of those products can be resold after inspection and repackaging. This return flow leads to an additional cost and labour effort in the warehouse, since the returned products have to be reintegrated in the stock before they are available for reselling.

Online retailers also face significant variations in demand. Especially in December, many e-commerce warehouses are challenged to keep up with demand for seasonal gifts. Although manual order picking systems are flexible and scalable, a doubling of the workforce does not necessarily lead to a doubling in throughput. With an increase in the number of order pickers in any area, more interactions between the order pickers arise, causing lower productivity. For example, aisles are typically so narrow that when two vehicles need to pass, careful manoeuvring is required. Furthermore, one order picker may be blocking access to a location from which another order picker needs to retrieve products.

E-commerce is thus redefining the requirements for the operation of warehouses, see De Koster, De Brito, and Van de Vendel (2002) and Stock and Mulki (2009). The warehouse process of retrieving products from storage in response to customers' orders is known as order picking. This process is generally thought to be the most costly and labor-intensive part in warehouse operations. It can contribute to 55% of the overall warehouse operation costs, see Tompkins et al. (2010). Furthermore, the largest portion of an order picker's time in manual picking systems is spent on travelling between locations. Product returns and order-picker interactions only add to these travel costs. The high costs involved in order picking and the challenges of product returns and interactions in busy e-commerce environments have motivated us to revisit the warehouse routing problem.

Our first goal is to incorporate the restocking of returned products in the order-picking routes. It is important to realise that the restocking of returned products is similar to the order-picking process, and quite different from regular stock replenishments. For regular stock replenishments, a vehicle typically replenishes only one or a few locations per trip with large quantities of the product. Restocking of returned items, on the other hand, requires visiting many locations while restocking only a single item per location, which results in a significant amount of travel. Thus for order picking, an order picker starts with an empty vehicle and gradually fills the vehicle to its capacity by retrieving products from storage locations

*Corresponding author. Email: a.h.schrotenboer@rug.nl

in response to customers' demand. While for restocking of returned products, an order picker starts with a full vehicle, which is gradually emptied by bringing returned products to their designated storage locations. As with the general Travelling Salesman Problem with Pickup and Delivery (TSPPD), which is NP-hard, see Mosheiov (1994), it seems advantageous for travel distances to integrate the two processes.

Our second goal is to give insights into the effects of order-picker interactions on efficiency and to incorporate interaction avoidance strategies as an integral part of the routing method. Order-picker interactions are especially of concern when designing routes that combine the restocking of returned products with the order picking of customers' orders, as is our first goal. Due to the capacity restriction of the vehicle, an order picker may not always be able to pick products when it is most logical from a routing point of view, since it may be necessary to first free capacity in the vehicle by dropping off returned products. This will cause the routes to be more complex and to include some backtracking. Furthermore, for any given capacity of the vehicle, more locations can be visited in a single route with restocking than in a route without restocking. Both the increased probability of backtracking and the increased number of stops per route will increase the probability of order-picker interactions; hence the need to study return handling and order-picker interactions simultaneously.

Next to routing, there are often also other control methods involved in operating a picking area, see De Koster, Le-Duc, and Roodbergen (2007). For example, batching methods aim at combining (parts of) several orders into a single picking route (e.g. Hong, Johnson, and Peters 2012). Our focus on routing can be explained from the fact that product returns and picker interactions must be included in the routing method, since the routing method serves to verify feasibility and route length. Furthermore, the use of additional methods is not precluded by our approach, since the products considered in the routing can be the result of a batching method.

Order-picker routing in warehouses, for picking activities only, is a well-studied topic in research. We refer to De Koster, Le-Duc, and Roodbergen (2007), Gong and De Koster (2011) and Gu, Goetschalckx, and McGinnis (2007) for general warehouse literature and order picking in specific. Recently, Theys et al. (2010) consider multi-aisle warehouse layouts and concluded that the inclusion of local search techniques in the warehouse routing problem is promising. To exploit the benefit of the inclusion of sophisticated search methods in the solution procedure, we propose a hybrid genetic algorithm (HGA), i.e. a genetic algorithm with local search aspects for solving the warehouse routing problem with pickups and deliveries (warehouse TSPPD). HGAs for general pickup and delivery problems already exist, see Zhao et al. (2009a, 2009b). We however construct a heuristic that is inspired by more sophisticated HGAs as presented by Vidal et al. (2012, 2013). Furthermore, we use specific characteristics of the warehouse routing problem in the algorithmic design.

Also an extended version of this HGA is presented that can reduce order-picker interactions by quantifying and penalising them. Interaction between order pickers was first investigated by Pan and Shih (2008) and Parikh and Meller (2009) from a queuing theory perspective. Recently, Chen et al. (2013, 2014) looked into a related problem for order-picking without product returns. Our method allows for a trade-off between delays caused by interactions and the time required for interaction avoidance strategies, while the approach of Chen et al. (2013, 2014) requires all interactions to be avoided.

Our two HGAs can identify routes for combined order picking and restocking in low computation times that are acceptable for real-time applications. Moreover, they are suitable for various warehouse layouts and can therefore be widely applied. For situations without order-picker interactions, we demonstrate that near-optimal, and often optimal, solutions are obtained by the HGA. Furthermore, we perform an analysis to identify the best way of mixing restocking and order-picking requests in the routes. Finally, we demonstrate that significant improvements are achieved by explicitly considering order-picker interactions in the algorithmic procedure.

The structure of the paper is as follows. In Section 2, we give a detailed problem description and introduce an ILP formulation of the problem under study. Section 3 is dedicated to the explanation of the HGA. A description of the extensions made to the HGA to account for order-picker interactions is given in Section 4. We discuss the results of our numerical experiments in Section 5, and conclude the paper in Section 6.

2. Problem description

This paper considers the warehouse pickup and delivery problem with order-picker interaction. We first present a description of the warehouse pickup and delivery problem without interactions (warehouse TSPPD) for which an Integer Linear Program (ILP) is constructed as well. After that, order-picker interaction is defined in the general setting of multiple order pickers. We consider a warehouse consisting of two cross aisles that are connected by n_{aisle} parallel aisles of length a_{length} and width a_{width}. A graphical representation of this system is given in Figure 7 in De Koster, Le-Duc, and Roodbergen (2007). All routes start and end at the central depot.

2.1 *Single order picker*

The warehouse TSPPD is defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = (\mathcal{P} \cup \mathcal{D} \cup \{0\})$ the set of vertices representing the products to be picked ($\mathcal{P}$), delivered ($\mathcal{D}$) and the central depot $\{0\}$, and $\mathcal{E}$ is the set of edges connecting all locations $v \in \mathcal{V}$. The orders used can be the result of batching methods. Let $n = |V| - 1$ be the order size and let $c : \mathcal{E} \to \mathbb{R}$ be the cost or distance function. For $i, j \in \mathcal{V}$, c_{ij} is defined as the shortest distance between i and j, see Theys et al. (2010). The order picker transport capacity $q > 0$ is defined as the maximum number of products an order picker can transport at any time.

We present an ILP formulation for the single order-picker case, based on the model in Mosheiov (1994). For $(i, j) \in \mathcal{V}$, let x_{ij} be a binary variable equalling 1 if the order picker travels along edge (i, j) and 0 otherwise. Furthermore, let $y_{ij} \geq 0$ be the total load already picked and transported along edge (i, j), and let $z_{ij} \geq 0$ be the total load to be delivered and transported along edge (i, j).

The warehouse TSPPD can now be formulated as:

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad \forall\, j \in \mathcal{V} \tag{2}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \forall\, i \in \mathcal{V} \tag{3}$$

$$\sum_{j=0}^{n} y_{ij} - \sum_{j=0}^{n} y_{ji} = \begin{cases} p_i & i \in \mathcal{P} \\ -\sum_{j=0}^{n} p_i & i = 0 \\ 0 & i \in \mathcal{D} \end{cases} \tag{4}$$

$$\sum_{j=0}^{n} z_{ij} - \sum_{j=0}^{n} z_{ji} = \begin{cases} -d_i & i \in \mathcal{D} \\ \sum_{j=0}^{n} d_i & i = 0 \\ 0 & i \in \mathcal{P} \end{cases} \tag{5}$$

$$y_{ij} + z_{ij} \leq q x_{ij} \qquad \forall\, i, j \in \mathcal{V} \tag{6}$$

$$x_{ij} \in \{0, 1\}, \qquad y_{ij},\ z_{ij} \geq 0 \qquad \forall\, i, j \in \mathcal{V} \tag{7}$$

The objective function (1) represents the total travel costs to be minimized when travelling a complete route. The Constraints (2) and (3) assure that each location is visited exactly once. With constraints (4) and (5), we control for the currently transported volume between any pair of locations i and j. Constraint (4) requires that the volume of any location i is picked up, if $i \in \mathcal{P}$, and that all products were picked up, when the order picker returns to the depot ($i = 0$). Constraint (5) states that the entire volume to be delivered to location i is delivered , if $i \in \mathcal{D}$, and all items were delivered at the end of the route. Constraint (6) restricts the volume of the total currently transported load between each pair of locations i and j to the transport capacity of the order picker.

2.2 *Multiple order pickers and interaction effects*

We consider two interaction events that cause delays for order pickers. First, if order pickers are in close proximity, both order pickers are assumed to incur a delay. Such delay may be caused by various reasons, including blocking of access to pick locations or simply slowing down for safety reasons. Second, two order pickers travelling in opposite directions slow down when passing each other, which is also registered as a delay. For brevity we will say that order pickers are *close* and that order pickers *cross*, for these two events, respectively. To be able to quantify the interaction effects, we first extend our description of the problem by introducing multiple order pickers and by adding a trace of the route each order picker travels. That is, at some well-defined moments in time, the location of the order pickers has to be known in order to detect order-picker interactions. To do so, it is assumed that order pickers travel at constant speed s_{cross} and s_{par} through cross aisles and parallel aisles, respectively. Product picking time and delivery time are constant at s_{pick}, and order pickers are assumed to be stationary during this time.

Let $\mathcal{M} = \{1, \ldots, m\}$ be the set representing all order pickers. Then for some order picker $a \in \mathcal{M}$, let $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$ be a complete graph with $\mathcal{V}_a = (\mathcal{P}_a \cup \mathcal{D}_a \cup \{0\})$ the set of vertices representing the products to be picked ($\mathcal{P}_a$), delivered ($\mathcal{D}_a$), and the central depot ($\{0\}$), and $\mathcal{E}_a$ is the set of edges connecting all locations $v \in \mathcal{V}_a$. The order size n, the cost function c and the transport capacity q are defined as before. Furthermore, let $\mathcal{S}_a$ be the set consisting of all solutions to the order picker routing problem on $\mathcal{G}_a$ and $\mathcal{S} = \cup_{a \in \mathcal{M}} \mathcal{S}_a$.

For some order picker $a \in \mathcal{M}$ and some solution $\sigma_a \in \mathcal{S}_a$, the interaction costs $\mathcal{I}(\sigma_a)$ are defined as the delay order picker a incurs due to interaction with the other $m-1$ order pickers. The time horizon is given by $\mathcal{T} = \{0, \delta_{\text{time}}, 2\delta_{\text{time}}, 3\delta_{\text{time}}, \dots, T\}$, where δ_{time} is a properly chosen step size transforming the continuous time horizon into a discrete one and T is the latest time an order picker is finished. The function $\tau : \mathcal{S}_a \times \mathcal{T} \to \mathbb{R}^2$ maps σ_a and time t to the location in the warehouse of order picker a at time t while travelling according to solution σ_a. Such a location is represented by a pair of coordinates, i.e. $\tau(\sigma_a, t) = (x_a, y_a) \in \mathbb{R}^2$. Then for some order pickers $a, b \in \mathcal{M}$ and corresponding solutions $\sigma_a \in \mathcal{S}_a, \sigma_b \in \mathcal{S}_b$ and time $t \in \mathcal{T}$, order pickers are said to be *close* if $|\tau(\sigma_a, t) - \tau(\sigma_b, t)| \le \delta_{\text{loc}}$, where $|\tau(\sigma_a, t) - \tau(\sigma_b, t)|$ is defined as the warehouse distance between $\tau(\sigma_a, t)$ and $\tau(\sigma_b, t)$. Then let $N_{\text{loc}}^{(\sigma_a, \sigma_b)}$ be the number of times order pickers a and b are close when they travel according to solution σ_a and σ_b, respectively. It is defined as

$$N_{\text{loc}}^{(\sigma_a,\sigma_b)} = \sum_{t \in \mathcal{T}, t \neq 0} I\{|\tau(\sigma_a, t) - \tau(\sigma_b, t)| \le \delta_{\text{loc}}\}, \tag{8}$$

where $I\{\cdot\}$ is an indicator function returning 1 if the condition between parentheses is true and 0 otherwise.

Let $h(\sigma_a, \sigma_b, t) = \big(\tau(\sigma_a, t) - \tau(\sigma_b, t)\big) \odot \big(\tau(\sigma_a, t - \delta_{\text{time}}) - \tau(\sigma_b, t - \delta_{\text{time}})\big) \in \mathbb{R}^2$, where $\odot$ is the Hadamard product, i.e. element-wise multiplication. Then, the number of times order pickers *cross* for solutions σ_a, σ_b, denoted by $N_{\text{cross}}^{(\sigma_a,\sigma_b)}$, is

$$N_{\text{cross}}^{(\sigma_a,\sigma_b)} = \sum_{t \in \mathcal{T}, t \neq 0} \sum_{i=1}^{2} I\Big\{h(\sigma_a, \sigma_b, t)_i < 0 \wedge h(\sigma_a, \sigma_b, t)_1 \cdot h(\sigma_a, \sigma_b, t)_2 = 0\Big\},$$

where $h(\sigma_a, \sigma_b, t)_i$ refers to the i-th element of the vector $h(\sigma_a, \sigma_b, t)$. The order picker crossings are penalised with a delay of d_c for both order pickers that cross, while the delay for being close is equal to d_l for both order pickers. Then, the interaction costs for some solution $\sigma_a \in \mathcal{S}_a$ are defined as

$$\mathcal{I}(\sigma_a) = \sum_{b \in \mathcal{M} \setminus \{a\}} \left[d_c \cdot N_{\text{cross}}^{(\sigma_a,\sigma_b)} + d_l \cdot N_{\text{loc}}^{(\sigma_a,\sigma_b)} \right]. \tag{9}$$

Finally, let $\mathcal{L} = \{\sigma_1, \dots \sigma_m\}$, $\sigma_i \in \mathcal{S}_i$, $\forall i \in \mathcal{M}$ be a set of solutions to m warehouse TSPPD problems with order-picker interaction. Then $\mathcal{I}(\mathcal{L})$ is the sum of all pairwise interaction costs between the solutions in $\mathcal{L}$,

$$\mathcal{I}(\mathcal{L}) = \sum_{a \in \mathcal{M}} \mathcal{I}(\sigma_a). \tag{10}$$

Notice that the interaction costs are a summation of the pairwise occurred delays between order pickers. For events when two order pickers interact, this is straightforward and exact. However, there may occasionally be events when three or more order pickers interact simultaneously. This may cause additional delays beyond those accounted for when summing the interactions between each pair of order pickers. Thus our formula for interaction delays may provide an underestimation in some situations. However, an event with simultaneous interactions between three or more order pickers already accounts for more delays than an interaction between two order pickers. Thus, our solution procedure, which aims to reduce interactions, will try to address these events with priority and therefore further minimise the occurrence of these already rare events.

3. Hybrid genetic algorithm

Hybrid genetic algorithms (HGAs) rely on the repeated generation of sets of solutions, starting from an initial population, which is iteratively altered by crossover, mutation and education functions. At each step, the quality of a solution is evaluated and influences the probability of its attributes to survive in the next generation. Considering recent successful applications of HGAs (e.g. Vidal et al. 2012, 2013) in related studies, hybrid genetic algorithms appear to be a well-suited tool for our problem. Mutation of individuals, i.e. single solutions in a population, allow us to incorporate warehouse-specific characteristics by developing mutation operators that make use of the warehouse layout information. In addition, HGAs allow for the inclusion of sophisticated search methods, as suggested in Theys et al. (2010). A schematic overview of the general procedure of the HGA we developed is given in Figure 1. The flow of the HGA can, after creating an initial population, be divided into four parts, which we will describe in detail in the following sections. Extensive preliminary experiments have shown that the actual construction of the initial population has no significant influence on the solution quality. We therefore employ a variant of the nearest neighbour heuristic with some degree of randomness in the sense that subsequent locations are selected

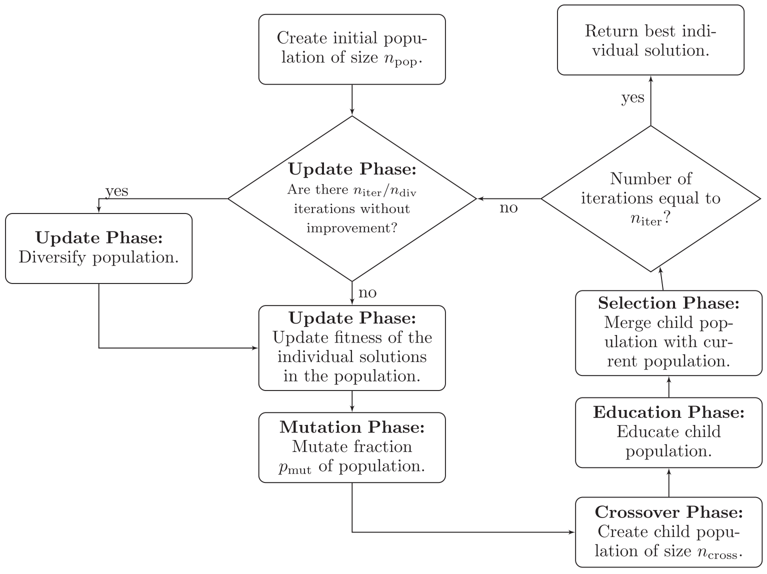
Create initial population of size n_{pop}.
Update Phase: Are there $n_{\text{iter}}/n_{\text{div}}$ iterations without improvement?
yes
no
Update Phase: Diversify population.
Update Phase: Update fitness of the individual solutions in the population.
Mutation Phase: Mutate fraction p_{mut} of population.
Crossover Phase: Create child population of size n_{cross}.
Education Phase: Educate child population.
Selection Phase: Merge child population with current population.
Number of iterations equal to n_{iter}?
no
yes
Return best individual solution.

Figure 1. Schematic overview of the general flow of the Hybrid Genetic Algorithm.

randomly, but based on probabilities according to their distance from the current location, i.e. parameterised regret-based random sampling.

3.1 *Update phase*

One of the major concerns in hybrid genetic algorithm design is convergence to local optima. It is therefore essential that promising attributes survive, whether they belong to feasible or to infeasible solutions. To achieve this, we allow a fraction p_{mut} of the population to be infeasible.

Besides allowing for infeasible solutions, also a strong diversification procedure is integrated in the HGA. Inspired by the approach of Vidal et al. (2012), if there are $n_{\text{iter}}/n_{\text{div}}$ iterations without improvement, the 90% worst solutions are removed from the population and replaced by newly generated initial solutions.

3.1.1 *Fitness function*

Although we allow for infeasible solutions, infeasibility is penalized. A similar approach is used in most modern genetic algorithms, see Vidal et al. (2012, 2013) and Zhao et al. (2009a, 2009b).

For a solution $\sigma \in \mathcal{S}$, we define $q^{\sigma}_{\max}$ as the maximum load the order picker at any point transports in solution σ . Hence the maximum transport capacity exceedance is given by $(q^{\sigma}_{\max} - q)^{+} = \max\{0, q_{\max} - q\}$. A linear increasing infeasibility penalty $p(i)$, where i denotes the iteration, is used to guide the search. The fitness of a solution $\sigma \in \mathcal{S}$ is the sum of the route length $\ell(\sigma)$ and the penalised transport capacity exceedance, and is given by

$$F(i, \sigma) = (q^{\sigma}_{\max} - q)^{+} \cdot p(i) + \ell(\sigma). \tag{11}$$

3.2 *Mutation phase*

The role of mutation operators for our solution approach is twofold. First, mutations add diversity to the search space of hybrid genetic algorithms. It prevents that the procedure stops at local minima, as crossover alone mostly facilitates the inheritance of attributes which are already available in the population. Second, in the mutation phase we also consider the warehouse-specific characteristics when defining mutation operators, such as sorting partial sequences in an intuitive order with respect to the warehouse layout. This facilitates the creation of attributes which might not be contained in the solutions of the population yet. Next to that, the mutation operators, as described below, exhibit a kind of local focus within single aisles to create potentially promising attributes.

We apply a mutation rate $p_m \in (0, 1)$, i.e. a percentage of the population size, to control the number of solutions which are altered by mutation operators in each iteration. A mutated individual solution thereby replaces the current worst not mutated

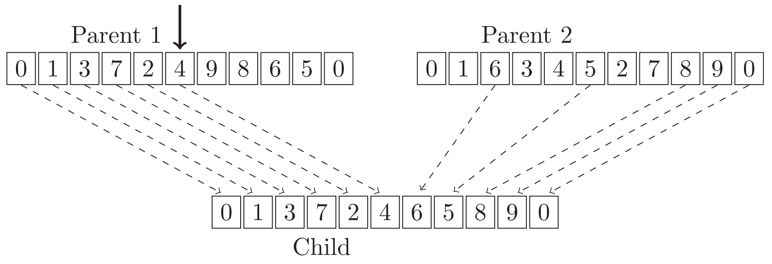

Figure 2. Illustration of the crossover operator. The solutions are represented by an array of product indices. The first 6 elements are selected from the first parent. Then from left to right, the products not already copied to the child are added from parent two.

individual solution from the population, if it is unique. A prioritisation of solutions to be selected for mutation is not made; each individual is equally likely to be selected. For the selected individuals, one out of three different mutation operators is randomly selected and applied a total of n_{mut} times to the individual. Clearly, a low values for n_{mut} cause less diversity in the population, which may tighten the search. In contrast, any too large value for n_{mut} might cause that mutated solutions are less likely to be selected for crossover, as in the crossover phase solutions are selected based on their fitness value.

With the first mutation operator, any aisle that contains locations to be visited is randomly selected. Next, all locations in this aisle are sequenced by their position in the aisle. The direction of sequencing is randomly selected. The resulting partial sequence containing all locations of one aisle is inserted in the complete routing sequence at any point of the original solution at which the corresponding aisle was visited before. By implication, all locations within this aisle are removed from the sequence at all other positions. The second mutation operator is designed in a similar manner. Here, again one aisle that contains locations to be visited is randomly selected. All locations in this aisle are sorted in such a way that the aisle is entered from one side by the picker, all deliveries are performed on the picker's way into the aisle, the picker turns at the farthest pickup or delivery location and all pickup requests are made on the picker's way back. Again, the side on which the picker enters and leaves the aisle is randomly selected with a uniform distribution. Finally, the third mutation operator randomly selects a visited location. It searches for all locations that are adjacent in the solution and located in the same aisle. This sequence of locations is removed from its current position in the solution and inserted at a new random position.

3.3 *Crossover phase*

The Crossover Phase aims to inherit well-performing attributes from the previous generation. Solutions of lower fitness value are more likely to contain well-performing attributes. To select such solutions with higher probability, parents are selected by means of a binary tournament selection. It is a frequently used parent selection method, see Vidal et al. (2012, 2013). It consists of choosing two pairs of two solutions at random, thereby selecting from each pair the individual solution with the lowest fitness value as a parent for the crossover.

The creation of a new individual solution, referred to as a child solution, from two parents can be described as follows. One location to be visited is randomly selected. Up to this location, all locations are inserted in the new individual in the same sequence as in the first parent. All remaining locations are added to the new individual in the sequence in which they appear in the second parent. The crossover procedure is illustrated in Figure 2. Doing so, two potentially well-performing individuals are re-combined in a way that maintains advantageous partial sequences. Obviously, the crossover as explained so far would allow only little variation in the beginning of sequences, as the first part (of random length) of an individual would always be adopted by the children from the first parent. To prevent this effect, we apply the crossover procedure either by starting at the beginning or at the end of the routing sequence. The selection of the direction is determined randomly beforehand for each newly created child.

This crossover procedure is used to construct a population of child solutions, called child population. Its size is controlled by the crossover rate p_{cross}, i.e. the number of new solutions entering the child population in each iteration.

3.4 *Education and selection phase*

The child solutions are at this stage typically of relatively low quality although they may contain promising attributes. To increase the probability that these attributes survive, all solutions of the child population are educated. We thereby increase the quality of the child solutions, which is relative beneficial for child solutions that contain promising attributes. Increasing child solutions' quality by means of local search seems to be standard today, see, among others, Vidal et al. (2012). We

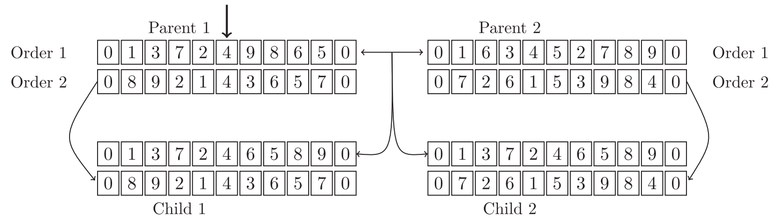

Figure 3. Illustration of a crossover between two sets of solutions of size 2. The first order is selected for actual crossover, thereby replacing the 'old' solutions of the first order, while the second order is simply copied to the child solutions.

therefore choose to let each child solution be subject to a fixed number n_{educ} of randomly chosen education operators, i.e. classical local search operators.

The first education operator is *Swap*. It considers in random order the exchange of two products, i.e. a possible move, in some child solution. As soon the operator finds a possible move that results in a beneficial change of the child solution's fitness, it is applied and the operator terminates. The second and third education operators are *Relocate* and *2-Opt*. They relocate a single location and reverse a part of the sequence of location visits, respectively. Again, if such a randomly chosen possible move is beneficial, it is applied and the operator terminates. Since the operators apply moves based on changing fitness values, a child solution may still be infeasible, or may become infeasible, at the end of the education phase. This is no issue, since infeasibility is penalised more for increasing iterations. Therefore, the education phase will produce relatively more feasible child solutions towards the end of the HGA.

Finally, the child solutions need to be merged with the parent solutions. There are two guidelines for merging these solutions. First, the fraction of infeasible solutions should not exceed its maximum p_{inf} and second, the fittest n_{pop} solutions should be selected to form the population for a new iteration.

4. Hybrid genetic algorithm with interaction effects

Order picker routing and order-picker interaction, as defined in this paper, are not simultaneously studied before. By adopting the general flow of the HGA, as presented in Section 3, and extending it by solving multiple warehouse TSPPDs simultaneously, thereby including order-picker interaction, a new HGA is developed, called Hybrid Genetic Algorithm with Interaction Effects (HGA-I).

Order-picker interaction is already quantified in Section 2.2. To allow for simultaneous solving of m warehouse TSPPDs, we store the information of a set of solutions $\mathcal{L} = \{\sigma_1, \dots, \sigma_m\},\ \sigma_i \in \mathcal{S}_i$. The population maintained by the HGA-I therefore consists of n_{pop} sets of solutions. To calculate the interaction costs $\mathcal{I}(\mathcal{L})$, the routes corresponding to the solutions need to be traced.

The array $\rho(\sigma_i)$ is filled with location–time pairs. Each pair consists of a location in the warehouse and the actual time it is visited by order picker $i \in \mathcal{M}$, if the order picker travels according to solution $\sigma_i \in S_i$. All relevant locations to detect order-picker interaction are contained by $\rho(\sigma_i)$. To be precise, all product visits and locations where the order picker turns, i.e. leaving an aisle or turning within the same aisle, are stored. In addition, every δ_{time} between two products visits or turning points the location and actual time are stored in $\rho(\sigma_i)$ as well. However, this implies that the location-time pairs for different order pickers can slightly differ in their time dimension. To overcome that problem, a bandwidth δ_{band} is introduced. If for some $a \in \rho(\sigma_a)$ and $b \in \rho(\sigma_b)$, the difference in time is less than δ_{band}, a and b are assumed to happen simultaneously.

The fitness function for a set of solutions $\mathcal{L}$ becomes

$$F(i, \mathcal{L}) = \sum_{\sigma \in \mathcal{L}} \left[(q^{\sigma}_{\max} - q)^{+} \cdot p(i) + \ell(\sigma) \right] + \mathcal{I}(\mathcal{L}), \tag{12}$$

where $I(\mathcal{L})$ is as given by Equation (10) and $p(i)$, $q^{\sigma}_{\max}$ and q are as given in Section 3.

The initial population generation is slightly changed to handle sets of solutions. Solutions are produced identically as in the HGA. After creation they are grouped to form a set of solutions and interaction costs are calculated afterwards. The Mutation Phase is unchanged. If a set of solutions is selected to be mutated all solutions are subject to mutation.

Parent selection is still performed according to binary tournament selection, although the flow of the Crossover Phase itself is adapted. Crossovers between two parent sets of solutions only alter one of the solutions in the set; all other solutions in the set remain the same. Two parents may generate two unique child solutions. See Figure 3 for an illustration.

Table 1. Performance of the HGA heuristic, compared with optimal solutions and the SLS heuristic.

Set	n_{aisle}	a_{length}	n	q	OPT	HGA	Difference HGA – OPT (%)	SLS	Difference SLS – HGA (%)
1	7	12	20	10	103.05	103.12	0.07	112.07	8.68
2	7	12	20	15	101.51	101.58	0.07	104.85	3.29
3	7	12	30	15	111.06	111.08	0.01	123.85	11.50
4	7	12	30	20	109.32	109.32	0.00	111.12	1.65
5	7	12	40	20	113.71	113.76	0.05	130.69	14.88
6	7	12	40	30	112.15	112.15	0.00	114.55	2.14

The Education Phase is slightly adapted compared to the case without interaction costs. Education is applied to every solution from a particular set of solutions. The education operators itself are not changed, implying that interaction costs are not updated during the exploration of the neighbourhood. Finally, the Selection Phase is left unchanged.

5. Numerical experiments

First, we provide insights in the performance and applicability of the HGA by comparing its computational results with optimal solutions and a simple local improvement heuristic. Lastly, we compare the HGA solutions (without taking interaction effects into account) to the HGA-I solutions (taking interaction effects into account). This will show that order-picker interaction is significant and should be taken into consideration when determining routes.

Our solution approach is applicable to a variety of warehouse layouts. Particularly, the HGA is independent of the length, alignment and number of storage aisles. We conduct our experiments in either a *small* warehouse, with $n_{\text{aisle}} = 7$, $a_{\text{length}} = 12$, $a_{\text{width}} = 2.5$, or a *large* warehouse, with $n_{\text{aisle}} = 15$, $a_{\text{length}} = 32$, $a_{\text{width}} = 2.5$. Pick and delivery locations are assumed to be solely in the parallel aisles and not in the cross aisles. Locations in the aisles are assigned with a 0.1 metres accuracy and random storage is assumed. Capacity and transported loads are measured for unit-size products. For each location, there is 1 unit of products to be picked or delivered. Extensive parameter calibration experiments showed that the following parameter values produce the highest quality solutions: $n_{\text{iter}} = 1500$, $n_{\text{pop}} = 100$, $n_{\text{cross}} = 100$, $p_{\text{surv}} = 0.3$, $p_{\text{mut}} = 0.05$, $n_{\text{mut}} = 1$, $p_{\text{inf}} = 0.05$, $n_{\text{educ}} = 6$ and $n_{\text{div}} = 5$.

In our preliminary experiments, we looked at the effect of allowing infeasible solutions at a penalty cost against not allowing them. Allowing infeasibility results in a bigger search space and consequently in slower convergence. Compared to not considering infeasible solutions the solution quality is on average not significantly different for the majority of the instances. We however found particular difficult instances of which the solution quality improved strongly by considering penalised infeasible solutions.

5.1 *Performance of the hybrid genetic algorithm*

The performance of the HGA is tested using 18 instance sets that consist of 100 randomly generated instances each. These can be interpreted as either the result from batching methods or as completely new customer orders. The instance sizes vary from 20 to 100 and have an equal number of pickups and deliveries, since this results in instances that are hardest to solve. For instances until size 40, optimal solutions could be obtained by means of CPLEX 12.5. A local search heuristic starting from a S-shaped solution, called SLS, serves as upper bound in the experiments. This heuristic works as follows. Using S-shaped routing, the picker traverses each aisle containing pickups or deliveries entirely. Pickup locations where in this route the capacity constraint would be violated, are skipped and the order picker returns to collect these pickups as soon as transport capacity suffices. After a solution is constructed, it is improved by applying the *2-Opt* operator, see Section 3.4, until convergence. It can be seen as the pickup-and-delivery counterpart of the 'S-shape + 2-Opt' heuristic as presented by Theys et al. (2010).

All methods are coded in C++11 and experiments are performed on an Intel Core i5-2400, 3.10 GHz processor. Solutions of the SLS are obtained within a second. The GA delivered results within 10 seconds for the smaller instances and within 2 minutes for the larger instances. Note that the final solutions are often reached within several seconds; a tuning of parameters to the specific warehouse layout at hand will therefore suffice to ensure the algorithm is fast enough for practical purposes. The calculation times for optimal solutions obtained through CPLEX varied from a few seconds for the instance sets 1 and 2 until 15 h for instance set 7. All results are presented in Tables 1 and 2.

Table 2. Performance of the HGA heuristic in comparison with the SLS heuristic.

Set	n_{aisle}	a_{length}	n	q	HGA	SLS	Difference SLS – GA (%)
7	7	12	50	25	117.08	135.84	16.02
8	7	12	50	35	114.51	117.36	2.48
9	15	32	60	30	502.60	555.55	10.53
10	15	32	60	40	498.91	510.53	2.33
11	15	32	70	35	514.73	580.23	12.73
12	15	32	70	45	511.06	523.11	2.36
13	15	32	80	45	528.76	590.65	11.70
14	15	32	80	50	524.96	537.15	2.32
15	15	32	90	45	537.80	609.97	13.41
16	15	32	90	55	533.92	545.53	2.17
17	15	32	100	50	543.20	623.68	14.82
18	15	32	100	60	539.54	551.69	2.25

The numerical results give insights in the performance of the HGA. As can be seen in Table 1, the HGA has an average gap to optimality of less than 0.1%, and can therefore be considered competitive by the current standard for meta-heuristics. The results are based on a single run of the HGA, and instances not solved to optimality with the HGA could typically be solved to optimality with just one extra run (not shown in the tables). The results for larger instances can be found in Table 2. The performance of the SLS heuristic is twofold. First, for instances, where the transport capacity is not very restrictive, it is at most 3.29% worse than the HGA. Second, the instances with more restrictive transport capacity show an average gap to the HGA between 8 and 16%. It is noticeable that the gap between the SLS and the HGA is comparable for small and large instances. We therefore are inclined to conclude that the HGA continues to produce high-quality solutions for larger instances.

5.1.1 *Practical implications*

In order to determine the best possible practical application of our solution approach, we performed a second set of experiments to analyse the most suitable composition of pickup and delivery requests in the routes. Clearly, the total number of delivery requests will typically be lower in e-commerce settings than the number of picking requests. Warehouses in online retailing are facing return rates ranging from 18 to 74%, see Mostard, De Koster, and Teunter (2005), depending on the product category and return opportunities. The data-set that was used for the following experiments consists of 7200 locations to be visited in total, of which one-third (i.e. 2400) are delivery requests. The goal of these experiments is to determine the best combinations of pickup and delivery requests in routes. We aim to find out whether the delivery requests should be distributed only over a few routes, or whether an even distribution of deliveries over all picking routes is more advantageous. For these experiments, the large warehouse layout was used. The locations of the 7200 pickup and delivery requests were assigned randomly with a uniform distribution. Here, the capacity of the picking device is set to 30. Seven cases for the composition of deliveries versus pickups were computed, which are characterized in Table 3. The cases vary between a full integration of pickup and delivery requests in one half of all routes, while the second half of the routes contains pickup requests only (case 1) and a completely separated processing of pickup and delivery requests in 240 routes, each containing 30 requests (case 7).

The results of these experiments are presented in Figure 4. Full integration of pickups and deliveries resulted in the shortest total travel distance (case 170.30km). However, we observe that a distribution of delivery requests over more routes does not affect the results significantly, as for case 2 (70.87 km), case 3 (71.13 km) and case 4 (71.86 km). For practical reasons, a distribution of deliveries over more routes might still be advantageous to allow for some flexibility for the order picker in sorting products in the picking cart. The use of the HGA contributes to these observations, since it is able to come up with high-quality routing solutions with product returns. In contrast, we find significant differences in the resulting travel distance for all cases in which (part of the) deliveries are processes separately. While in case 5 (76.72 km) and case 6 (84.52 km) still some pickup and delivery requests are performed in the same routes, in case 7 deliveries and pickup requests are entirely separated which leads to an overall travel distance of 91.87 km. This clearly shows that the integration of pickup and delivery requests can significantly reduce travel distance. Our experiments show savings of 23.48% between the cases 1 and 7.

Table 3. Cases for the route composition of deliveries versus pickups.

Case	Total number of batches	Pick batches	Delivery batches	# Mixed batches	Mix batch #pickup – # deliveries
1	160	80	0	80	30–30
2	160	64	0	96	30–25
3	160	40	0	120	30–20
4	160	0	0	160	30–15
5	176	0	16	160	30–12
6	208	0	48	160	30–6
7	240	160	80	0	–

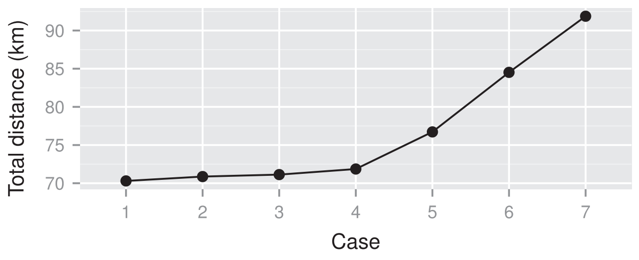

Figure 4. The resulting total traveled distance for the different batch composition cases.

Table 4. Warehouse properties for situations with interactions between order pickers.

Travel speed cross-aisle in m/s	$s_{\text{cross}} = 0.7$
Travel speed pick-aisle in m/s	$s_{\text{par}} = 1.3$
Item processing time in seconds	$s_{\text{pick}} = 20$
Penalty for order pickers' crossing in seconds	$d_{\text{c}} = 2$
Penalty for order picker's being close in seconds	$d_{\text{l}} = 1$
Bandwidth for space in meters	$\delta_{\text{loc}} = 1$
Bandwidth for time in seconds	$\delta_{\text{band}} = 1$
Step size for discrete approximation in seconds	$\delta_{\text{time}} = 1$

5.2 *Performance of the hybrid genetic algorithm with interaction effects*

The HGA provides us with high-quality solutions and showed its practical relevance. However, interaction effects were not yet taken into account. We now set out to test whether interaction effects can be mitigated by means of our HGA-I heuristic. To be able to quantify interaction effects, additional assumptions about warehouse properties must first be made. These are presented in Table 4. We use the same 18 instance sets as before, each consisting of 50 or 25 problems for, respectively, $m = 2$ or $m = 4$ order pickers. Parameters are again calibrated, but no significant differences with the results of the earlier calibration are obtained. We therefore use the same parameter settings.

We first determine order-picker interaction using the HGA for the 18 instance sets. This will provide a reference point that shows the amount of interactions when routing decisions are not yet adjusted to avoid interactions. Each problem is solved by the HGA, and afterwards the solutions are merged into groups of size m, the number of order pickers that simultaneously starts processing their orders. For each set of solutions, the interaction delays that arise from synchronously processing m orders are calculated afterwards. These are added to the original route durations of the m order pickers, resulting in a single objective. These results are presented as 'HGA' in Table 5. Secondly, we run our HGA-I to simultaneously determine routes for all m order pickers, while aiming to minimise total route time, including interaction delays. These results are presented as 'HGA-I' in Table 5. Note that, though the instances are the same, the objective values do not directly correspond to the

Table 5. Total route times including interaction delays when making routing decisions without taking interaction effects into account (HGA) and with taking interaction effect into account (HGA-I) for $m = 2$ and $m = 4$ order pickers. The total route times (obj.) and the total interaction delays (ID), which are part of total route times, are given for each instance set. The column 'difference' indicates the improvement in total route time when taking interaction effects into account for determining routes.

	$m = 2$					$m = 4$				
	HGA		HGA-I			HGA		HGA-I		
Inst.	Obj.	ID	Obj.	ID	Difference (%)	Obj.	ID	Obj.	ID	Difference (%)
1	204.30	5.80	200.05	0.68	2.08	458.20	61.20	412.58	9.76	9.96
2	200.94	5.24	195.74	0.00	2.58	451.88	60.48	398.71	6.24	11.76
3	216.61	5.80	213.19	0.68	1.58	492.88	71.20	440.86	9.52	10.55
4	212.62	4.84	207.84	0.00	2.25	481.17	65.60	422.99	5.68	12.09
5	221.73	7.08	217.52	1.20	1.90	504.40	74.88	448.56	9.60	11.07
6	218.84	6.72	212.15	0.00	3.06	497.69	73.44	431.45	5.28	13.31
7	225.16	5.84	221.53	0.76	1.61	526.23	87.68	456.99	9.04	13.16
8	223.38	7.56	215.83	0.00	3.38	512.84	81.20	438.76	4.96	14.44
9	873.91	4.80	872.21	0.44	0.19	1857.97	119.52	1772.20	15.12	4.62
10	868.43	6.80	861.63	0.00	0.78	1848.22	124.96	1730.84	6.32	6.35
11	895.66	7.80	892.07	1.20	0.40	1899.72	124.56	1815.68	16.24	4.42
12	888.16	8.52	879.67	0.00	0.96	1880.73	121.44	1767.80	7.36	6.00
13	914.24	5.80	912.51	1.28	0.19	1958.18	140.48	1862.68	18.56	4.88
14	906.14	5.00	901.14	0.00	0.55	1938.36	136.08	1808.84	6.32	6.68
15	929.85	6.40	929.39	1.04	0.05	1990.38	144.56	1893.81	17.76	4.58
16	921.16	6.32	916.84	0.00	0.47	1953.68	124.00	1837.85	6.48	5.93
17	937.99	6.20	936.69	1.24	0.14	2009.73	146.16	1910.31	17.12	4.95
18	930.41	7.24	923.23	0.00	0.77	2005.77	159.44	1854.54	6.48	7.54

values in Tables 1 and 2 since these are given in meters, whereas the results in Table 5 are for obvious reasons presented in seconds.

When determining routes in isolation, the percentage of total route time that is due to interaction delays varies between 0.5 and 3.4% in the $m = 2$ order pickers case, and between 6.3 and 15.8% in the $m = 4$ order pickers case. It is therefore evident that interaction effects increase strongly with an increase in the number of order pickers. The smaller warehouse (cases 1–8) gives higher interaction delays (13.4–15.8% for $m = 4$) than the larger warehouse (6.3–7.9% for $m = 4$). This is not surprising, since the probability that order pickers interact in a large warehouse is lower than in a small warehouse.

As can be seen from Table 5, the HGA-I is capable of decreasing the total route time significantly (refer to column 'difference'). For two simultaneously starting order pickers, including order-picker interaction considerations in the solution procedure results in up to 2.58% lower total route times. The relative decrease in objective value for the instances located in the larger warehouse is less than 1%, which again shows that in a relatively large warehouse with only two order pickers the interactions are not that significant. However, in the smaller warehouse with four order pickers, the HGA-I reduces total route time by up to 14.4%, which is due to a large decrease in interaction delays at the expense of only a slight increase in travel time. In depth analysis of individual routes showed that many of the HGA-I solutions are constructed without any conflicts between order pickers. This shows that in an environment with multiple pickers it may be very useful to use a routing method that can take order-picker interactions into account.

5.2.1 *Sensitivity analysis*

The interaction delays that the solutions of the HGA comprise are influenced by the specific characteristics of the order picking system in use. To give more insights in order-picker interaction and the effects of specific order picker characteristics on the order-picker interaction, a sensitivity analysis is conducted. At first, the effect of the item processing time is surprising. One would expect that a higher item processing time increases the probability that order pickers cross, since order pickers are stationary for a longer time giving more opportunity for other order pickers to cross. The results are, however, the opposite; higher item processing times leads to less order-picker interaction. For item processing times $s_{\text{pick}} = 5, 10, 15, \dots, 40$, the

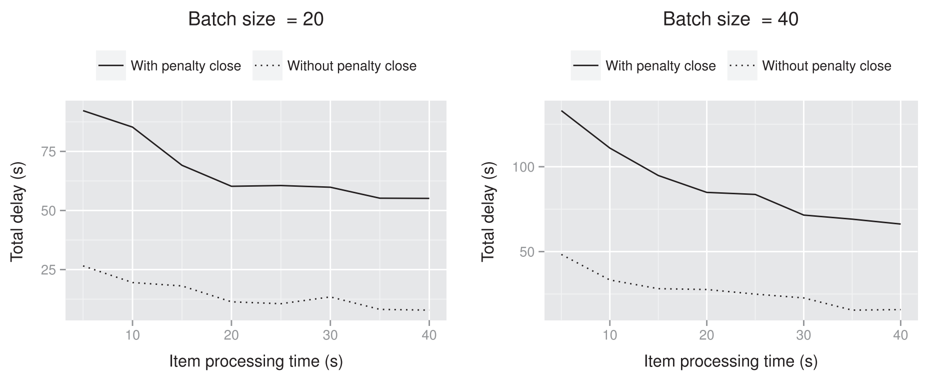

Figure 5. Sensitivity analysis of order-picker interaction for changing values of the item processing time. For batch sizes of 20 and 40, the results – with and without taking 'close' delays into account – are presented for various values of the item processing time.

resulting order-picker interaction delays between 4 order pickers are plotted in Figure 5. To obtain these results, we used the problems from instance set 7.

To investigate the composition of the order-picker interaction delays, we have performed these experiments with and without accounting for delays due to proximity (i.e. order pickers being 'close'). Thus, the results without 'close' only show delays for order picker crossing. Thirdly, the same experiments are performed for order picker problems that consist of smaller order sizes. We used the problems from instance set 1. The results are also presented in Figure 5. They show the same effect of decreasing order-picker interaction for an increasing item processing time.

Concluding, it is shown that order-picker interaction needs to be considered in devising routes and that the developed heuristic is capable of constructing routes that are nearly conflict free.

6. Conclusions

In this paper, we consider a routing problem in e-commerce warehouses for two types of jobs, order picking and restocking of returned products. The inclusion of products that are returned to the warehouse by customers required reconsideration of the classical warehouse order-picker routing problem. We propose a hybrid genetic algorithm to identify routes by which product returns can be returned to their storage locations, while simultaneously customer orders are picked. In numerical experiments, we demonstrate the performance of the solution approach and evaluate the potential gains in travel distance in comparison with a local search heuristic and with optimal solutions. It is shown that the hybrid genetic algorithm yields near-optimal, and mostly optimal, solutions, and that travel distances are decreased by up to 23.48% when product returns are included in the picking routes, instead of processing product returns separately. In addition, we explored the most suitable manner to integrate product returns in the picking routes and found that an incorporation of many returns in fewer picking routes is slightly preferable over an even distribution of returns over all picking routes.

Since e-commerce warehouses tend to be faced with periods of extreme work loads, which result in a high number of order pickers being employed in the same area, we also set out to investigate the effects of order-picker interaction. Furthermore, the integration of restocking activities of returned products in the order picking routes will increase order-picker interaction as well, which brought us to investigate both aspects in conjunction. To this end, the hybrid genetic algorithm is extended to include order-picker interaction effects. It is shown that order-picker interaction significantly contributes to the total route time and should be accounted for in solution approaches, especially when product returns are integrated in the regular order picking process.

There are some opportunities for further research. We have discovered some instances in which allowing for multiple depot visits in a single route yielded shorter route lengths, since the picker can drop already picked products at the depot, thus increasing capacity. Regarding order-picker interaction, it may be of interest to investigate how storage location assignments influences the significance of order-picker interaction. Finally, the possibilities for combining batching methods with our routing methods seem promising. Most of the batching literature assumes simple constructive heuristics for order picker routing, reasoning that this prevents order pickers from interacting. We, however, showed that near-optimal routes can be constructed that are almost interaction free.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This study was supported by a grant of the Dutch Institute for Advanced Logistics (Dinalog).

ORCID

Albert H. Schrotenboer http://orcid.org/0000-0001-8365-2204

References

Chen, Fangyu, Hongwei Wang, Chao Qi, and Yong Xie. 2013. “An Ant Colony Optimization Routing Algorithm for Two Order Pickers with Congestion Consideration.” *Computers & Industrial Engineering* 66 (1): 77–85.

Chen, Fangyu, Hongwei Wang, Yong Xie, and Chao Qi. 2014. “An ACO-Based Online Routing Method for Multiple Order Pickers with Congestion Consideration in Warehouse.” *Journal of Intelligent Manufacturing* 27 (2): 389–408.

De Koster, René, Marisa P. De Brito, and Masja A. Van de Vendel. 2002. “Return Handling: An Exploratory Study with Nine Retailer Warehouses.” *International Journal of Retail & Distribution Management* 30 (8): 407–421.

De Koster, René, Tho Le-Duc, and Kees Jan Roodbergen. 2007. “Design and Control of Warehouse Order Picking: A Literature Review.” *European Journal of Operational Research* 182 (2): 481–501.

Gong, Yeming, and René B. M. De Koster. 2011. “A Review on Stochastic Models and Analysis of Warehouse Operations.” *Logistics Research* 3 (4): 191–205.

Gu, Jinxiang, Marc Goetschalckx, and Leon F. McGinnis. 2007. “Research on Warehouse Operation: A Comprehensive Review.” *European Journal of Operational Research* 177 (1): 1–21.

Hong, Soondo, Andrew L. Johnson, and Brett A. Peters. 2012. “Large-Scale Order Batching in Parallel-Aisle Picking Systems.” *IIE Transactions* 44 (2):88–106.

Mosheiov, Gur. 1994. “The Travelling Salesman Problem with Pick-Up and Delivery.” *European Journal of Operational Research* 79 (2): 299–310.

Mostard, Julien, Rene De Koster, and Ruud Teunter. 2005. “The Distribution-Free Newsboy Problem with Resalable Returns.” *International Journal of Production Economics* 97 (3): 329–342.

Pan, Jason Chao-Hsien, and Po-Hsun Shih. 2008. “Evaluation of the Throughput of a Multiple-Picker Order Picking System with Congestion Consideration.” *Computers & Industrial Engineering* 55 (2): 379–389.

Parikh, Pratik J., and Russell D. Meller. 2009. “Estimating Picker Blocking in Wide-Aisle Order Picking Systems.” *IIE Transactions* 41 (3): 232–246.

Stock, James R., and Jay P. Mulki. 2009. “Product Returns Processing: An Examination of Practices of Manufacturers, Wholesalers/Distributors, and Retailers.” *Journal of Business Logistics* 30 (1): 33–62.

Theys, Christophe, Olli Bräysy, Wout Dullaert, and Birger Raa. 2010. “Using a TSP Heuristic for Routing Order Pickers in Warehouses.” *European Journal of Operational Research* 200 (3): 755–763.

Tompkins, James A., John A. White, Yavuz A. Bozer, and Jose Mario Azaña Tanchoco. 2010. *Facilities Planning*. New York: John Wiley & Sons.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. 2012. “A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems.” *Operations Research* 60 (3): 611–624.

Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 2013. “A Hybrid Genetic Algorithm with Adaptive Diversity Management for a Large Class of Vehicle Routing Problems with Time-Windows.” *Computers & Operations Research* 40 (1): 475–489.

Zhao, Fanggeng, Sujian Li, Jiangsheng Sun, and Dong Mei. 2009a. “Genetic Algorithm for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem.” *Computers & Industrial Engineering* 56 (4): 1642–1648.

Zhao, Fang-Geng, Jiang-Sheng Sun, Su-Jian Li, and Wei-Min Liu. 2009b. “A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Pickup and Delivery.” *International Journal of Automation and Computing* 6 (1): 97–102.