Mathematical Biosciences and Engineering

MBE, 19(6): 5546–5563.
DOI: 10.3934/mbe.2022259
Received: 29 December 2021
Revised: 02 March 2022
Accepted: 22 March 2022
Published: 28 March 2022

http://www.aimspress.com/journal/MBE

Research article

Order batching and order picking with 3D positioning of the articles: solution through a hybrid evolutionary algorithm

Fabio M. Miguel[1,*], Mariano Frutos[2,*], Máximo Méndez[3] and Fernando Tohmé[4]

[1] Universidad Nacional de Río Negro, Sede Alto Valle y Valle Medio, Argentina
[2] Engineering Department, Universidad Nacional del Sur, IIESS UNS CONICET, Argentina
[3] Universidad de Las Palmas de Gran Canaria (ULPGC), Instituto Universitario SIANI, Spain
[4] Economics Department, Universidad Nacional del Sur, INMABB UNS CONICET, Argentina

* **Correspondence:** Email: fmiguel@unrn.edu.ar, mfrutos@uns.edu.ar.

Abstract: A critical factor in the logistic management of firms is the degree of efficiency of the operations in distribution centers. Of particular interest is the pick-up process, since it is the costliest operation, amounting to 50 and up to 75% of the total cost of the activities in storage facilities. In this paper we jointly address the order batching problem (OBP) and the order picking problem (OPP). The former problem amounts to find optimal batches of goods to be picked up, by restructuring incoming orders by either splitting up large orders or combining small orders into larger ones that can then be picked in a single picking tour. The OPP, in turn, involves identifying optimal sequences of visits to the storage positions in which the goods to be included in each batch are stored. We seek to design a plan that minimizes the total operational cost of the pick-up process, proportional to the displacement times around the storage area as well as to all the time spent in pick-ups and finishing up orders to be punctually delivered. Earliness or tardiness will induce inefficiency costs, be it because of the excessive use of space or breaches of contracts with customers. Tsai, Liou and Huang in 2008 have generated 2D and 3D instances. In previous works we have addressed the 2D ones, achieving very good results. Here we focus on 3D instances (the articles are placed at different levels in the storage center), which involve a higher complexity. This contributes to improve the performance of the hybrid evolutionary algorithm (HEA) applied in our previous works.

Keywords: order batching; order picking; evolutionary algorithm; local search; optimization

1. Introduction

Interest on the operation of warehouses and distribution centers has grown considerably in recent years, particularly due to the growing importance of e-commerce and the increasing number of deliveries of small size packages [1]. Improving the efficiency of the operational process in storing facilities is critical for the overall performance of a firm [2–5]. These processes include all the activities carried out in distribution centers, like receiving, transferring, putting away, sorting, picking-up, classifying, grouping, accumulating and dispatching orders [6–8]. This requires coordinating different technical teams and designing the lay-out of the storage center [9,10].

The receiving activities consist in unloading orders from vehicles, carrying them inside the depot, updating the inventory databases and inspecting the items to detect eventual inconsistencies with the quality, quantity and packaging declared. The putting away tasks amount to moving the items from the unloading dock to their assigned place in the storage area, registering their corresponding placement. This involves handling the goods, verifying the suitability of the placement sites and proceeding to store them until they are requested in an order [11]. The order picking process covers the most expensive activities in most warehouses, consisting in picking up the right amount of requested goods from the corresponding storage sites and taking them to the preparation/dispatch area [3,12,13]. Classifying and grouping the requested and picked up articles consists in consolidating the orders of the clients in the modular packages (boxes, pallets, containers, etc.) to be dispatched [14,15]. In most cases this involves marking, labeling and grouping several goods into a single unit load. Dispatch is the process of verifying the load units to be transported, checking that the orders received are fulfilled, elaborating the documents to accompany them, to finally load them on transports.

Given the high costs in resources and time of the logistics in the storage facilities, leads us to focus on the order picking activities. We are particularly interested in the solution of the integral batching and pick-up problems, by taking the displacement costs and the earliness and tardiness penalties into consideration. We treat this combined problem in a multi-level storage system (with a 3D positioning of items) solved by applying a hybrid evolutionary algorithms.

The rest of this paper is organized as follows: Section 2 describes the problem and reviews the relevant literature. Section 3 presents the formal model, while Section 4 describes the algorithm devoted to solve this question. Section 5 presents datasets with different order characteristics and 3D warehouse environments and section 6 discusses the results of computer experiments Finally, section 7 discusses the results and the prospects for further work.

2. Problem description and literature review

To pick up orders fastly and at a minimal cost, three planning problems, involving operations in warehouses or distribution centers, have to be addressed. One is the allocation of storage positions for the received articles. Another one is batching-up the orders in lots to be collected. Finally, we have the problem of scheduling the sequence of pick-ups and delivery displacements to the dispatch area [16]. In this paper we focus on the integrated treatment of the two last problems, which are critical for the efficiency of the activities in the storage floor. Furthermore, they generate most of the costs associated to the operation of the distribution center since they involve activities that are intensive in labor and equipment [17–19].

A more precise description of the *order picking process* is the following. The process starts with incoming multiple customer orders. Each order involves requests of different articles made by a customer, detailing the amount of each article and the due time at which the order has to be available at the dispatch area. The articles must be picked-up from the storage positions at the right time by a pick-up team that has to visit them in a sequence [20,21]. Masae et al. [22] review systematically the literature on order picker routing policies.

Small orders may reduce the displacement time, by finishing them in single picking tours. This can be taken advantage by order batching procedures that split up larger orders into smaller ones. Alternatively, small orders can be combined in a single large order that can then be picked in a single picking tour [23]. Figure 1 depicts this process, indicating that it starts and ends at the dispatch area.

So, the integrated process consists in designing a plan to reduce the cost of selecting, picking-up batches of different orders simultaneously, given precise deadlines for the finishing of each one. This cost is proportional to the displacement times across the storage floor and the punctuality in finishing the requested tasks. Since the final goal is to dispatch the batches, the schedule of pick-ups depends on the schedule of deliveries to customers. If the order were finished with delays it would lead to violations of the contracts with customers. Penalties and related costs ensue. On the other hand, if the orders are finished before the stipulated time the goods may clog the regular flow in the dispatch area, increasing the processing time of urgent requests, increasing the total cost of operations.

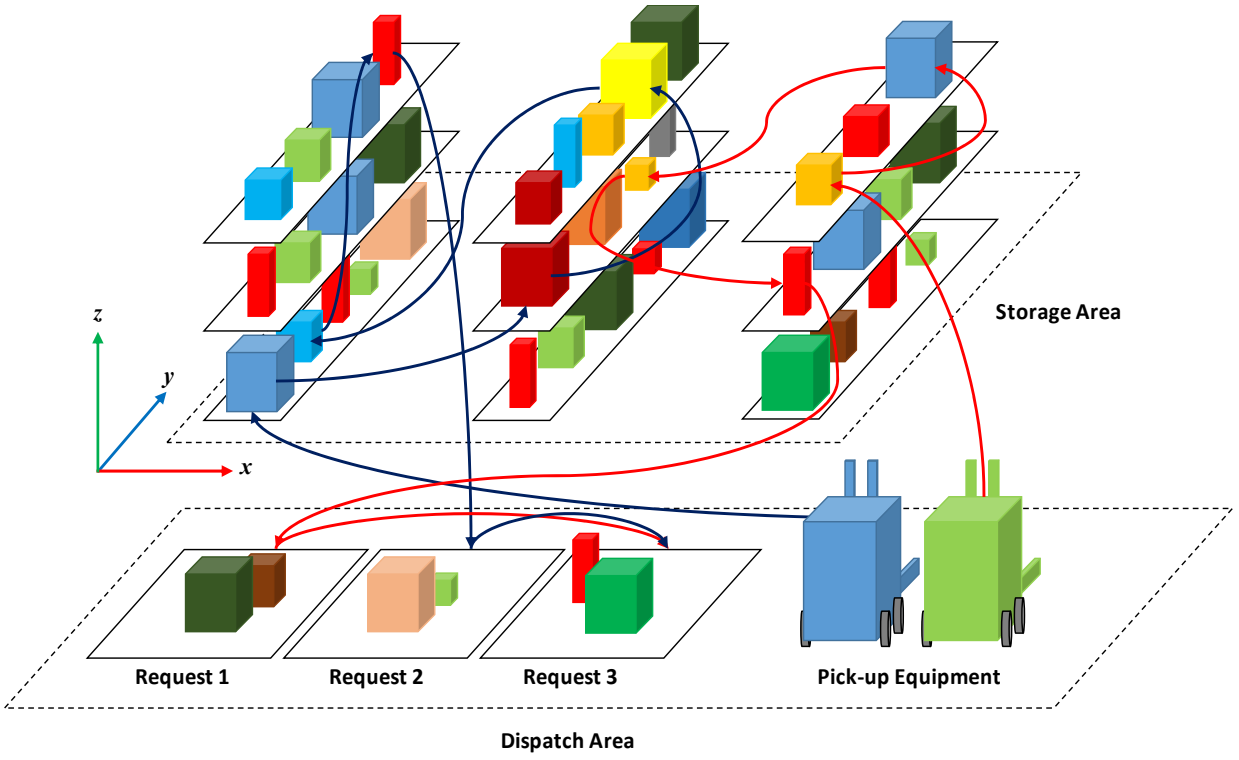

Figure 1. Pick-up process.

In more formal terms, this problem integrates two already known NP-Hard problems, the order batching problem (OBP) and the order picking problem (OPP). OBP consists in determining how to conform the batches that will be collected by the pick-up teams, taking into account the capacities of the

teams and the time at which each article should be at the dispatch area. OPP in turn consists in identifying optimal visit sequences to the storage positions in which the goods of a given batch can be found, minimizing the distance covered and the processing time of the batch, visiting each storage position only ones. We denote the integrated problem as OBP/OPP.

De Koster et al. [7] present a thorough literature revision on OBP and OPP. Ho and Tseng [24] reviewed the heuristics used to solve OBP. Chen and Wu [14] use a clustering-based approach to solve OBP taking into account demand patterns instead of the distance covered in each sequence of visits. Henn et al. [15] proposed using heuristics like taboo search and attribute-based hill climbing for OBP. Henn and Schmid [25] added the iterated local search heuristic to address OBP. Later on, Lam et al. [26] proposed and integer programming model for OBP in which the distance covered by each sequence is estimated and the problem is solved using a fuzzy logic-based heuristic.

Van Gils et al. [27] reviewed and classified the literature on OPP. Petersen [28], De Koster et al. [6] as well as Theys et al. [29] presents revisions of the heuristics for OPP. Henn et al. [30] used ant colony optimization and iterated local search to solve OPP, while Chen and Lin [31] used a very efficient two-stage method. Lu et al. [32] present a routing algorithm for dynamical OPP.

Tsai et al. [2] used a multiple-GA to solve the integrated OBP/OPP on 2D and 3D item positions. The novelty of this work resided in the use of flexible time windows for the delivery of orders, allowing a certain degree of earliness and tardiness. Miguel et al. [3], presented an evolutionary algorithm hybridized with a local search method to solve the 2D instances in reference [2]. Later on, Miguel et al. [4] changed the representation of individuals, improving the results of reference [3].

In this paper we seek to further test and improve the performance, using experimentation, of the algorithm presented in reference [3] applying it on the 3D instances of reference [2].

The main contributions of this paper are the following:

i) We addressed the order batching and order picking with a more realistic model, by considering a multi-level storage system (with a 3D positioning of items), explicitly incorporating the due times of the requests.

ii) We take into account that the actual procedures in warehouses involve several pick-up teams. This has been barely approached in the literature on the processing and routing of pick-ups in distribution centers.

iii) The improvements are obtained by implementing an algorithm that solves the problem in a single stage, unlike the more complex algorithm presented in [2], which requires 2 stages.

iv) The algorithm is tested on sets of orders generated probabilistically using the methodology presented in reference [2], showing its robustness.

The literature has not covered sufficiently the resolution of the integrated problem by dividing larger problems into smaller ones to be fulfilled by multiple pick-up teams. While reference [33] comes close to this, it does not allow the splitting of orders. This aspect of our work indicates its scientific relevance.

3. Specification of the problem

We present here a specification of OBP/OPP based on a mathematical programming formulation [2]. In this setting we define the decision variables, the objective function as well as the constraints that involve delivery deadlines, number of pick-up teams and the operational restrictions on the use of the storage facility.

3.1. Parameters

$\mathcal{P}$ denotes the set of different items in storage[1]. We assume that there are $nArt$ items. Each unit of an item has a weight, being the set of those weights $\mathcal{W} = \{w_1, \dots, w_p, \dots, w_{nArt}\}$. $\mathcal{P}_i$ is the subset of items requested by customer i. We assume that each customer places a single order or request of different articles, with different amounts of them. Thus, the number of customers nC, equals the number of requests $nReq$, being the set of customers/orders $\mathcal{I} = \{1, \dots, i, \dots, nC\}$. Each request has a specified deadline, with the class of those deadlines being $\mathcal{T} = \{t_1, \dots, t_i, \dots, t_{nPed}\}$. $\mathcal{P}_r$ is the set of items in a batch r. The items in $\mathcal{P}_r$ can be requested by a single or several customers. $\mathcal{L} = \{\ell_0, \ell_1, \dots, \ell_p, \dots, \ell_{nArt}\}$ represents the set of storage positions of the types of articles plus ℓ_0 which indicates the dispatch area. For an item $p \in \mathcal{P}$, its storage position ℓ_p is given by its coordinates in the store, $\ell_p = (x_p, y_p, z_p)$. $\mathcal{R} = \{1, .., r, \dots, |\mathcal{R}|\}$ indicates that set of batches to be picked-up. $\mathcal{S}_r = \langle s_1, \dots, s_u, \dots, s_{|\mathcal{S}_r|} \rangle$ is the sequence of positions to be visited to conform batch r, where s_u is the $u-th$ position to be visited. $q_{i,p} \in Q$ denotes the number of units of item p requested by customer i, with $Q_i = \sum_{p \in \mathcal{P}_i} q_{i,p}$ being the total number of goods demanded by i while $Q_p = \sum_{i \in \mathcal{I}} q_{i,p}$ denotes the total number of requested units of p. Finally, $\mathcal{K} = \{1, \dots, |\mathcal{K}|\}$ is the set of pick-up teams, each one with total weight capacity Cap.

We define an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ are the nodes, each one corresponding to a storage position of an item $p \in \mathcal{P}$, plus two copies (0 and $n+1$) of the node corresponding to the dispatch area. $\mathcal{A}$, represents the class of edges connecting pairs of nodes of $\mathcal{V}$. Each $edge(h,l) \in \mathcal{A}$ has an associated travel time t_{hl} given by the distance between positions h and l over the speed of a picking-up team, v (i.e., $t_{hl} = D_{h,l}/v$) and an operational cost for each unit of time, ς. t_{pick} is the average time to pick a unit of any item, once the team has reached the corresponding storage site.

3.2. Binary flow variables

$x_{hlkr} = 1$ if h is picked-up right before item in storage position l by the pick-up team k in the sequence corresponding to batch r, where $h, l \in \mathcal{V}$, $k \in \mathcal{K}$ and $r \in \mathcal{R}$. This means that $x_{hlkr} = 1$ if team k has to travel through $edge(h,l)$ to form batch r.

3.3. Binary index variables

$y_{hkr} = 1$ if k picks up item in storage position h for batch r, where $h \in \mathcal{V}$, $k \in \mathcal{K}$ and $r \in \mathcal{R}$.

3.4. OBP/OPP model

$$\min TOC: \left[\frac{\sum_{h \in \mathcal{V}} \sum_{l \in \mathcal{V}} D_{h,l} \cdot \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} x_{hlkr}}{v} + \sum_{\substack{p \in \mathcal{P} \\ q \in Q}} q_p \cdot t_{pick}\right] \cdot \varsigma + \sum_{i \in \mathcal{I}} (\alpha \cdot E_i + \beta \cdot T_i) \quad (1)$$

s.t.

$$\sum_{h \in \mathcal{V}_r} (q_p \cdot w_p) \cdot y_{hkr} \leq Cap, \ \forall k \in \mathcal{K}, r \in \mathcal{R} \quad (2)$$

[1] Each item has a different reference code (SKU). So for instance, 200 units of the smartphone EjS22 share the same SKU, which is not the same of the smartphone EjS7, another item.

$$\sum_{r\in\mathcal{R}} y_{hkr} = 1,\ \forall h \in \mathcal{P}, \forall k \in \mathcal{K} \tag{3}$$

$$\sum_{k\in\mathcal{K}} y_{hkr} = |\mathcal{K}|,\ \forall h \in \{0, n+1\}, r \in \mathcal{R} \tag{4}$$

$$\sum_{h\in\mathcal{V}} x_{hlkr} = y_{lkr},\ \forall l \in \mathcal{V} \setminus \{0\}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5}$$

$$\sum_{l\in\mathcal{V}} x_{hlkr} = y_{hkr},\ \forall h \in \mathcal{V} \setminus \{n+1\}, k \in \mathcal{K}, r \in \mathcal{R} \tag{6}$$

$$\sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} q_{i,p} \cdot y_{pkr} = \mathcal{Q}_p,\ \forall p \in \mathcal{P} \tag{7}$$

$$\sum_{p\in\mathcal{P}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} q_{i,p} \cdot y_{pkr} = \mathcal{Q}_i,\ \forall i \in \mathcal{I} \tag{8}$$

$$x_{hlkr} \in \{0,1\},\ \forall h, l \in \mathcal{V}, k \in \mathcal{K}, r \in \mathcal{R} \tag{9}$$

$$y_{hkr} \in \{0,1\},\ \forall h \in \mathcal{V}, k \in \mathcal{K}, r \in \mathcal{R} \tag{10}$$

The objective function Eq (1) represents the total operational cost of the pick-up process, expressed in monetary units, corresponding to the collection of batches, plus penalties for earliness and tardiness. The travel times in the first term obtain from the analysis of the lay-out of the storage facility.

With respect to the second term, α is the penalty per unit of time for earliness, while β is the current penalty per unit of time for tardiness. E_r is the earliness in making up order i, while T_i is tardiness in making up order i. Formally, $E_i = max\{0, t_i - c_i\}$ and $T_i = max\{0, c_i - t_i\}$, where c_i is the finishing time of order i, defined as the time at which all the units of all items corresponding to order i are picked up and collected to form the order.

Constraints Eq (2) forbid that the total weight of a batch exceed the capacity of the teams and their equipment, $\mathcal{V}_r$ is the set of storage positions of the items that belong to lot r ($p \in \mathcal{P}_r$). Constraints Eq (3) indicate that each storage position cannot be visited more than once for each batch r. Constraints Eq (4) ensure that all pick-up teams start and end at the dispatch area. In turn Eqs (5) and (6) preserve the flow of pick-up operations. If pick-up team k gets item l for batch r, it has to have picked up item h or vice versa. Constraints Eq (7) indicate that all the requests of item p are satisfied, while constraints Eq (8) state that all the requests of customer i have to be satisfied. Constraints Eqs (9) and (10) restrict the range of values of variables.

3.5. Lay-out of the distribution center

Figure 2 shows the lay-out of the distribution center used in our previous analysis of the OBP/OPP problem. The bottom left corner corresponds to the access to the dispatch area, where all the sequences of pick-up operations start and end. That is, a team leaves the dispatch area following a pre-specified sequence, going from a storage position to another, picking-up the items before moving on to the next storage position, until all the articles in a batch have been collected. After that, the team returns with the articles to the dispatch area. Graphically, we have surrounded with a dashed red curve the storage and dispatch areas, where all the actual OBP/OPP operations are carried out.

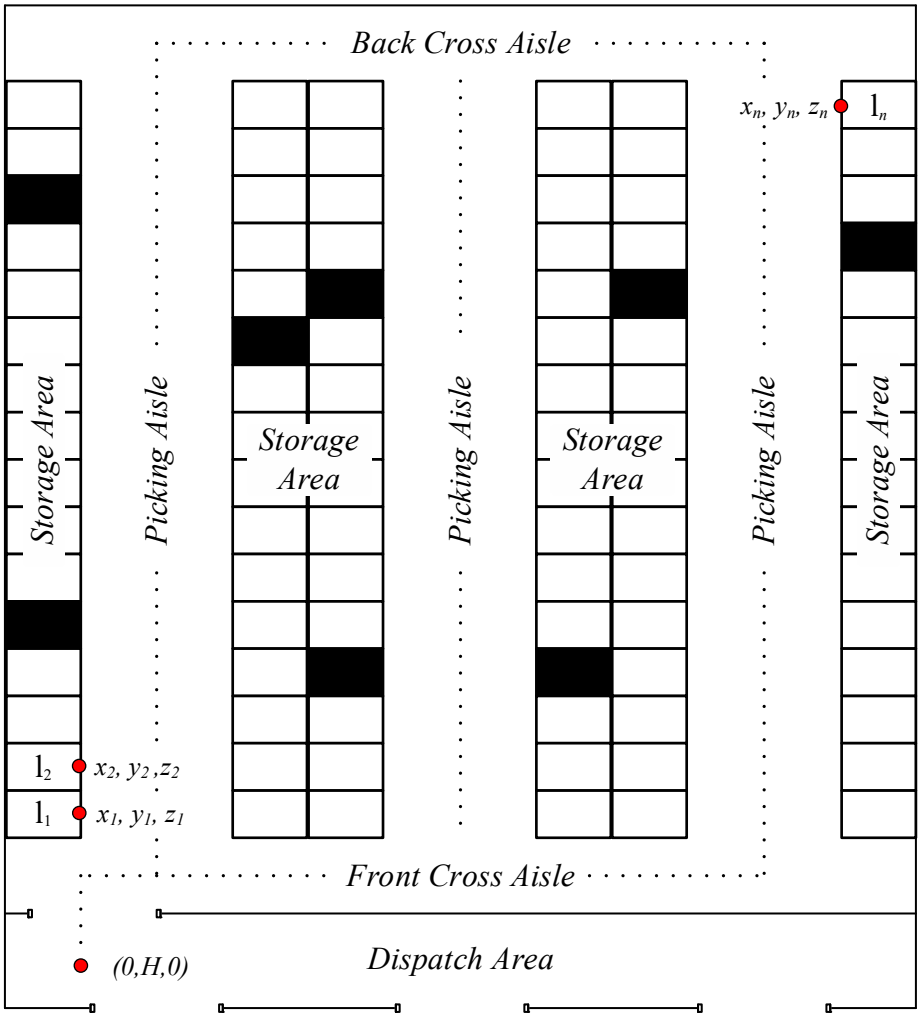

Figure 2. Layout of the storage center.

This configuration agrees with the one presented originally by Tsai et al. [2]. We keep the parameters used by them to test and validate our proposed solution method for the OBP/OPP problem. More precisely, we assume that there are two lateral racks and two double racks in the middle of the storage area. Pick-ups can be carried out along two transversal and three longitudinal aisles. The distance traveled from the storage position of item l to that of h, i.e., from ℓ_l to ℓ_h, where $\ell_l = (x_l, y_l, z_l)$ and $\ell_h = (x_h, y_h, z_h)$, can be expressed as follows:

$$D_{l,h} = \begin{cases} |x_l - x_h| + |y_l - y_h| + |z_l - z_h|, & if \ \ \mathcal{A}_l = \mathcal{A}_h \\ \\ |x_l - x_h| + min\{|2\mathcal{Y} - (y_l - \mathcal{H}) - (y_h - \mathcal{H})|; |(y_l - \mathcal{H}) - (y_h - \mathcal{H})|\} + |z_l - z_h|, if \ \ \mathcal{A}_l \neq \mathcal{A}_h \end{cases}$$

In this expression $\mathcal{A}_l$ and $\mathcal{A}_h$ denote the aisles along which ℓ_l and ℓ_h can be reached, respectively. $\mathcal{H}$ is the second coordinate of ℓ_0, the start and end of a pick-up sequence. $\mathcal{Y}$ is the length of the picking aisles.

4. Hybrid evolutionary algorithm (HEA)

The algorithm applied was introduced in reference [3] to solve the 2D instances presented by Tsai et al. [2]. It consists of an evolutionary algorithm [34] that evolves the solutions to the OBP/OPP problem hybridized with a constructive method based on the k closest neighbors heuristic, to improve the sequences using a local search with λ-exchanges. It uses the representation of as permutation of integers, usual in the treatment of combinatorial problems. Each chromosome consists of two genomes. A simple example shows how this representation works. Consider three requests and four items. Table 1 indicates that, for instance, request 1 consists of 2 items (one unit of A and two units of C), totaling 3 articles.

Table 1. Example of an OBP/OPP problem.

Request	Item				Total
	A	B	C	D	
1	1	0	2	0	3
2	1	1	2	1	5
3	0	2	0	2	4
Total	2	3	4	3	12

Table 2 shows a representation consisting of two rows, one for items and the other for requests. The columns indicate the total number of requested articles (in this case, 12).

Table 2. Items and corresponding requests.

Item	A	A	B	B	B	C	C	C	C	D	D	D
Request to which it belongs	1	2	2	3	3	1	1	2	2	2	3	3

Table 3 indicates how the information in Table 2 is used to define two genomes. The first genome contains the information of the items assigned to batches (we consider three batches in this case). So, for instance, one unit of item A in request 1, is assigned to batch 2. On the other hand, genome 2 corresponds to the sequence of visits to the storage positions of the items. For instance, in batch 2, item A is collected from position 2. Table 4 presents the final chromosome to be evolved.

Table 3. Genome 1 (Assigned batch) and Genome 2 (Position in the sequence of a batch).

Item	A	A	B	B	B	C	C	C	C	D	D	D
Request to which it belongs	1	2	2	3	3	1	1	2	2	2	3	3
Genome 1 / Assigned batch	2	3	2	3	3	1	1	3	1	1	1	3
Genome 2 / Position in sequence	2	1	1	3	5	1	3	4	5	4	2	2

Table 4. Chromosome (Genome 1 + Genome 2).

Genome 1												Genome 2											
2	3	2	3	3	1	1	3	1	1	1	3	2	1	1	3	5	1	3	4	5	4	2	2

Table 5 shows how the chromosome of Table 4 is decoded, indicating the batches and their corresponding pick up sequences.

Table 5. Decoding a chromosome.

Batch	Item (Request)				
1	C (1)	D (3)	C (1)	D (2)	C (2)
2	B (2)	A (1)			
3	A (2)	D (3)	B (3)	C (2)	B (3)

We can see that this chromosome is not yet a solution for the problem since, for instance for batch 1, this sequence prescribes going from the dispatch area to the position where item C is, pick up one unit, then go the position of item D, pick up one unit and *return* to the position of C to pick up another unit. After that the sequence prescribes going back to the position of D and then again back to C. This is clearly inefficient, since it could pick up three units of C and then two of D (or the other way around) with a lower cost. The evolutionary process ends up discarding chromosomes like that of Table 4.

The main advantage of this type of representation that incorporates specific knowledge about the problem is that it allows reaching higher levels of efficiency than Holland's original binary representation [35]. The downside is that it requires using operators adapted to this representation instead of general ones [36].

The penalty term in the objective function facilitates discarding non-feasible chromosomes. On the other hand, the satisfaction of the family of constraints Eqs (2)–(6) is ensured by the hybridization of the genetic operators with the closest neighbor heuristic. The constraints given by Eqs (7) and (8) are trivially satisfied by the representation.

The initial population is generated randomly, to ensure a larger variety of alternatives. The process terminates according a cost-oriented criterion, which limits the maximal number of iterations.

We use here the tournament selection approach [37], according to which the individuals that get selected are those with the higher scores among k individuals chosen at random from the current population. This is repeated until a new population is generated.

Table 6 presents the pseudo-code of the algorithm, as presented in reference [3].

5. Datasets and parameter settings

Tsai et al. [2] generated instances for OBP/OPP positioning items in two and three dimensions, as shown in Table 7. We have previously applied the HEA to find very good solutions to some 2D instances (DS0, DS1, DS2 and DS3) [3,4]. Here we address 3D instances (DS4, DS5 and DS6). In this, more complex setting, we can test and improve the performance of the algorithm.

Table 6. Pseudo-code of the HEA.

```
1: Load Input % information of requests, the lay-out of the storage center and parameters of the algorithm.
2: nBatch ← nBatchMin
3: while nBatch < nBatchMax
4:    t ← 0;
5:    P(t) ← InitPop(input);
6:    FitP(t) ← EvalPop(P(t));
7:    For t ← 1 a MaxNumGen
8:          Q(t) ← SelecBreeders (P(t), FitP(t));
9:          Q(t) ← HybridCrossover(Q(t));
10:         Q(t) ← HybridMutation(Q(t));
11:         FitQ(t) ← EvalPop(Q(t));
12:         P(t) ← SelSurviv(P(t), Q(t), FitP(t), FitQ(t));
13:         FitP(t) ← EvalPop(P(t));
14:         if TermCond(P(t), FitP(t))
15:               break;
16:         end
17:   end
18: nBatch ← nBatch + 1;
18: end
```

Table 7. Instances of the OBP/OPP problem.

	DS0	DS1	DS2	DS3	DS4	DS5	DS6
Problem size	Small	Small	Medium	Large	Small	Medium	Large
Number of requests	25	40	80	200	40	100	250
Number of different items	30	80	160	300	80	200	400
Lay-out type	2D	2D	2D	2D	3D	3D	3D
Average total weight (kg.)	18584	13704	37152	158784	12424	54048	295784
Capacity of pick-up teams (kg.)	7000	10000	10000	20000	10000	10000	50000

We assume that the number of units of item p requested by a customer i is uniformly distributed between 1 and 10, i.e., $q_{i,p} \sim U(1, \dots, 10)$. In turn, the number of different ítems requested by a customer follows a normal distribution, with mean 10 and standard deviation 5, i.e., $|\mathcal{P}_i| \sim N(10,5)$. The deadline for finishing the request of i follows a uniform distribution over the range of seconds between 10:00 am and 18:00 pm, $t_i \sim U(36000, \dots, 64800)$. The unit weight of each item p is uniformly distributed between 8 and 24 kilograms, i.e., $w_p \sim U(8, \dots, 24)$. With respect to the parameters of the pick-up teams, we assume an average travel speed of $v = 2\ m/s$, an average pick-up time for each unit of $t_{pick} = 15$ seconds, a travel cost per unit of time $\varsigma = \$\ 0.05$ and a capacity (Cap) that depends on the instance under consideration.

For the objective function we consider an earliness penalty per unit of time of $\alpha = 0.5$ while

the corresponding penalty for tardiness is $\beta = 1$.

With respect to the strategy to limit the search space and facilitating the comparison with the results of Tsai, Liou and Huang [2], we postulate lower and upper bounds on the number of possible batches, $|\mathcal{R}|_{min}$ and $|\mathcal{R}|_{max}$, respectively. These bounds are defined as follows: $|\mathcal{R}|_{min} = (\varphi_1 \cdot \sum_{p\in\mathcal{P}} w_p)/Cap$ and $|\mathcal{R}|_{max} = (\varphi_2 \cdot \sum_{p\in\mathcal{P}} w_p)/Cap$, where φ_1 and φ_2 are constants such that $\varphi_2 \geq \varphi_1$. If φ_1 is too large or φ_2 is too small, unfeasible sequences may be generated. The latter case generates longer pick-up routes and the batches may exceed the capacity of the pick-up teams. Similarly, a large φ_1 would induce a very large number of batches that may generate large travel costs.

Tables 8–10 summarize the information about instances DS4, DS5 and DS6, respectively. Each row indicates the items and amounts of them requested by customer i. The last column shows the deadline for finishing the requests.

Table 8. Inputs for instance DS4.

Request	Items:quantities	Deadline
$\mathcal{P}_1$	(4:2) (5:9) (18:5) (40:4) (55:5) (62:6) (67:5)	17:38:41
$\mathcal{P}_2$	(1:6) (11:7) (13:1) (16:3) (18:4) (20:2) (58:6) (59:9) (60:5) (64:8) (68:6) (76:3) (77:2)	17:11:44
$\mathcal{P}_3$	(7:1) (21:10) (27:5) (31:4) (38:7) (41:8) (42:5) (44:4) (46:8) (51:7) (60:7) (78:5)	15:56:12
$\mathcal{P}_4$	(23:4) (35:4) (53:9) (54:8) (62:4) (64:8) (70:7) (80:8)	17:57:13
$\mathcal{P}_5$	(10:2) (33:7) (38:8) (39:1) (41:7) (46:7) (52:7) (53:5) (63:3) (71:10)	15:15:02
$\mathcal{P}_6$	(11:5) (19:3) (44:9) (59:2) (72:7) (80:8)	15:52:38
$\mathcal{P}_7$	(11:10) (18:4) (26:5) (34:4) (37:4) (53:4) (54:4)	14:22:30
...	...	...
$\mathcal{P}_{38}$	(17:10) (34:3) (47:4) (49:3)	14:35:54
$\mathcal{P}_{39}$	(15:2) (23:10) (31:9) (45:8) (59:9) (66:2) (69:6) (76:6)	15:24:01
$\mathcal{P}_{40}$	(9:2) (29:1) (37:6) (49:6) (58:3) (74:2) (76:1)	16:36:54

Table 9. Inputs for instance DS5.

Request	Items:quantities	Deadline
$\mathcal{P}_1$	(15:9) (34:6) (107:10) (120:4) (134:7) (178:2) (184:1) (192:6)	15:05:09
$\mathcal{P}_2$	(1:6) (7:8) (40:7) (62:2) (85:1) (98:10) (121:8) (123:6)	16:10:57
$\mathcal{P}_3$	(6:1) (19:10) (44:4) (113:9) (115:2) (124:10) (132:5) (160:6) (167:10) (195:5) (198:2)	16:24:33
$\mathcal{P}_4$	(57:6) (78:1) (85:9) (89:1) (94:2) (107:7) (133:3) (143:4) (166:8) (180:9) (199:2)	15:27:07
$\mathcal{P}_5$	(7:10) (21:8) (33:5) (37:1) (50:6) (74:2) (119:4) (171:3) (176:5)	16:42:02

Continued next page

Request	Items:quantities	Deadline
$\mathcal{P}_6$	(11:3) (15:4) (34:5) (37:2) (48:2) (69:8) (155:6) (174:9) (176:9) (177:8) (182:9) (188:9)	15:38:01
$\mathcal{P}_7$	(73:1) (145:9) (191:7) (193:6)	15:25:51
...	...	...
$\mathcal{P}_{98}$	(44:1) (57:8) (65:7) (79:5) (121:3) (157:3) (189:7)	14:47:52
$\mathcal{P}_{99}$	(34:10) (52:5) (120:8) (133:1) (160:2)	15:29:12
$\mathcal{P}_{100}$	(7:8) (13:6) (28:10) (38:2) (51:8) (86:2) (121:5) (127:3) (183:10)	14:58:35

Table 10. Inputs for instance DS6.

Request	Items:quantities	Deadline
$\mathcal{P}_1$	(10:2) (87:5) (126:7) (172:1) (207:4) (276:4) (314:1) (351:8) (396:7)	14:46:43
$\mathcal{P}_2$	(54:1) (107:4) (157:3) (255:7) (328:2)	15:31:42
$\mathcal{P}_3$	(141:5) (152:10) (171:7) (243:7) (306:5)	15:16:38
$\mathcal{P}_4$	(35:1) (68:7) (194:3) (228:6) (313:8) (321:2) (335:4) (359:8) (375:7) (391:6)	14:13:20
$\mathcal{P}_5$	(54:9) (70:9) (85:7) (115:3) (154:6) (280:8) (370:2) (399:4)	15:30:15
$\mathcal{P}_6$	(45:5) (64:4) (111:7) (150:1) (161:6) (166:5) (346:10)	15:26:27
$\mathcal{P}_7$	(84:8) (151:10) (313:9) (353:2) (400:4)	14:06:20
...	...	...
$\mathcal{P}_{248}$	(43:4) (157:4) (313:10) (362:2)	15:01:47
$\mathcal{P}_{249}$	(66:8) (99:10) (114:4) (140:4) (174:9) (178:10) (195:5) (196:3) (292:7)	16:17:18
$\mathcal{P}_{250}$	(26:6) (56:8) (97:4) (106:6) (107:1) (126:4) (186:6) (228:4) (254:3)	17:27:49

A first stage of calibration allows assigning values to other parameters. So, the maximal number of iterations is $MaxGen = 500$, the size of populations is $PopSize = 150$, the size of tournaments in the selection process is $SizeTournament = 2$, the parameters in the bounds on the number of batches are $\varphi_1 = 2$ and $\varphi_2 = 4$, the probability of crossover is $Pr_{cross} = 0.9$, the probability of mutations is $Pr_{mut} = 0.15$ while the number of individuals in the elite is (5% of the population) $nElite = 0.05 \cdot PopSize$, using direct sampling as elite selection rule.

6. Computational experiment

In this section we present the results of running our HEA on 3D instances (DS4, DS5 and DS6) generated by Tsai et al. [2]. We used a PC with an Intel Core i7 3.00 GHz processor with a 8 GB RAM. 100 runs for each instance yielded the results presented in this section. Figures 3–5 depict the TOC when the number of batches $\mathcal{R}$, varies between $|\mathcal{R}|_{min}$ and $|\mathcal{R}|_{max}$ for each instance.

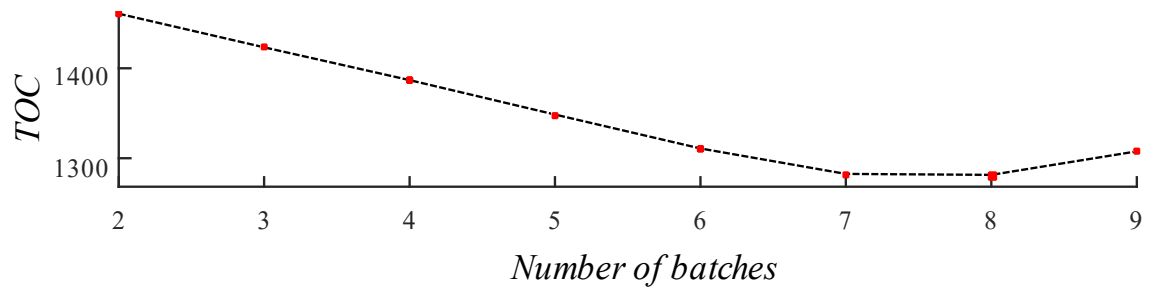

Figure 3. DS4: TOC values under different numbers of batches.

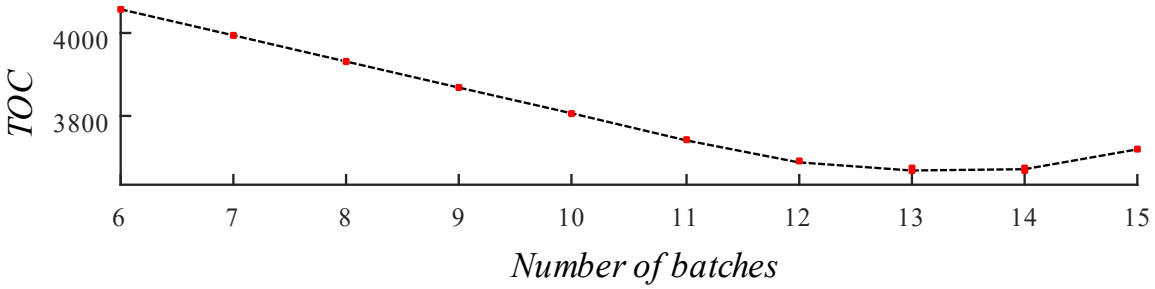

Figure 4. DS5: TOC values under different numbers of batches.

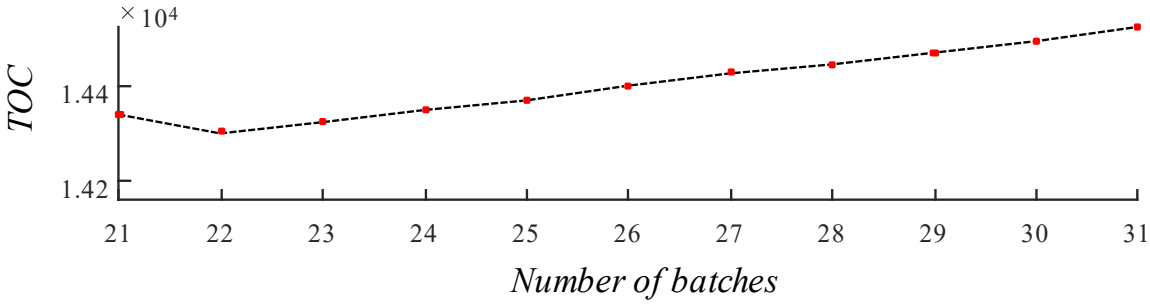

Figure 5. DS6: TOC values under different numbers of batches.

We can see that the number of batches is optimal when TOC reaches its lowest value. In order to find more precise optimal values, we penalized the solutions with excess weight, adding an additional cost to them.

Table 11 presents the best solutions obtained by the HEA for instances DS4, DS5 and DS6. We compare them to results in reference [2]. The rows indicate some natural measures of performance, like the total travel distance in the pick-up plan (D_{total}), the optimal number of batches, the mean of the distribution of distances travelled on batches (D_{med}), the standard deviation of that distribution (D_{std}), the sum of earliness and tardiness (ET_{time}) and the total operational cost (TOC). Here distances are measured in meters, times in seconds and monetary costs in US dollars.

Table 12 indicates that HEA yields a better TOC on the different instances. This is, in general not the case for the total distance traveled (except for instance DS5).

Table 11. Performance of HEA and comparison with a benchmark.

	Multiple-GA [2]			HEA [3]		
	DS4	DS5	DS6	DS4	DS5	DS6
D_{total}	948.00	4905.00	17799.00	1036.74	4470.20	19448.89
$nLot$	8.00	15.00	23.00	8.00	14.00	22.00
D_{med}	118.50	327.00	773.87	129.59	319.30	884.04
D_{std}	8.26	18.96	24.35	11.26	24.55	59.94
ET_{time}	1790.00	6826.00	25500.00	1700.50	6552.96	22440.00
TOC	1322.20	4431.03	14546.63	1281.58	3672.14	14317.82

Table 12. Performance ratios.

	DS4	DS5	DS6	Average
D_{total}(Multiple-GA)/D_{total}(HEA)	0.914	1.097	0.915	0.976
TOC(Multiple-GA)/TOC(HEA)	1.032	1.207	1.016	1.085

Figure 6 presents boxplots indicating the variability, degree of asymmetry and the extreme values of the distributions of D_{total}, ET_{time} and TOC obtained with our HEA on instances DS4, DS5 and DS6, using the optimal number of batches in each case.

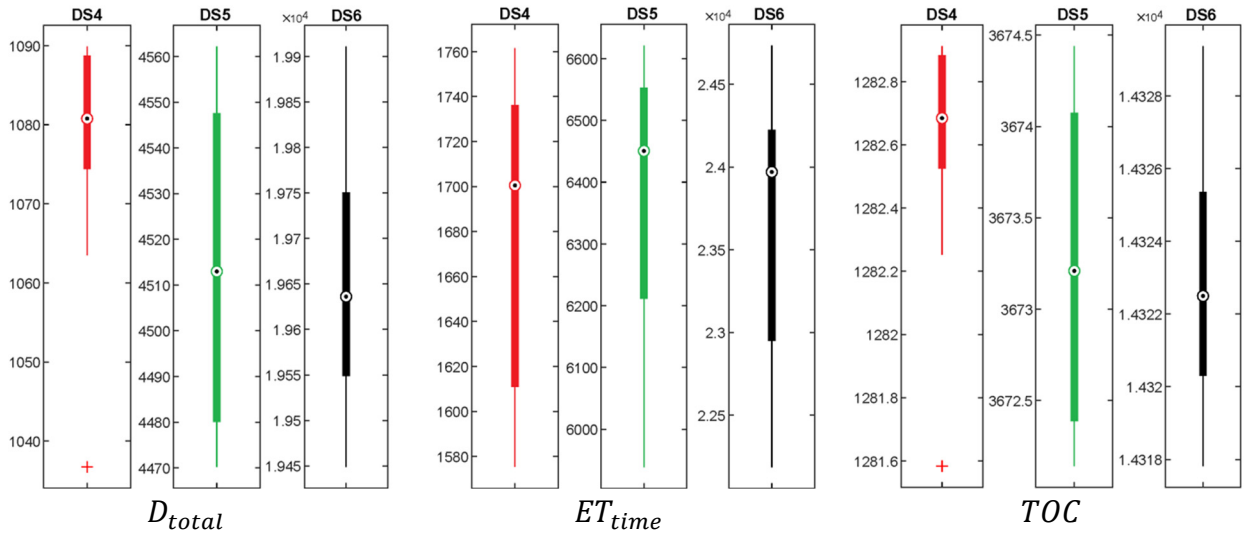

Figure 6. Boxplots: DS4, DS5 y DS6.

We can see that in the boxplot of instance DS4 for D_{total}, we have an atypical value close to the minimal one for 8 batches, but 50% of the values of the distribution of D_{total} is way above that value. In the case of ET_{time} the central values are below their corresponding benchmark values.

This is also the case for TOC. In summary, even if HEA does not yield better results in terms of D_{total}, the quality of solutions measured by TOC and ET_{time} improves over the known results.

7. Conclusions

In this paper we addressed an integrated problem that combines the Order Batching Problem (OBP) and the Order Picking Problem (OPP) using a hybrid evolutionary algorithm (HEA). This algorithm uses a novel representation of the problem using specific knowledge that allows restricting the search space. The crossover and mutation operators are hybridized with a heuristic that allows finding better routes for the pick-up teams. Tsai et al. [2] generated 2D instances of this problem (in which items are stored at the same level) and 3D ones (with items placed at different levels). We have previously applied the HEA to 2D instances, with very good results [3,4]. In this paper we have addressed the more complex 3D instances. The results, in terms of the objective function, improve over those in reference [2], which were obtained using a Multiple-GA. This completes the testing of the HEA.

Future work involves considering this problem but in a multi-objective and cross-dock setting. The reach of the problem can be extended, studying the relation between the placement of items in the different layouts and the distribution of the last mile. Other aspects should also be considered, in particular those impacting on the operational planning of the activities in storage sites, like the volume and size of the packages, the relative demands of the different items, their rotation and similar considerations.

Acknowledgments

The authors are grateful for partial support from the following sources: National Scientific and Technical Research Council CONICET Grant PIP No. 11220150100777, Universidad Nacional del Sur Grant PGI 24/J086 and Universidad Nacional de Río Negro PI-JI 40-A-917.

Conflict of interest

The authors declare they have no financial interests. In addition, the authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. E. Sancaklı, İ. Dumlupınar, A. O. Akçın, E. Çınar, İ. Geylani, Z. Düzgit, Design of a routing algorithm for efficient order picking in a non-traditional rectangular warehouse layout, in *Digitizing Production Systems* (eds. N. M. Durakbasa, M. G. Gençyılmaz), LNME, (2022), 401–412. https://doi.org/10.1007/978-3-030-90421-0_33
2. C. Y. Tsai, J. J. Liou, T. M. Huang, Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. *Int. J. Prod. Res.*, **46** (2008), 6533–6555. https://doi.org/10.1080/00207540701441947

3. F. Miguel, M. Frutos, F. Tohmé, D. A. Rossit, A memetic algorithm for the integral OBP/OPP problem in a logistics distribution center, *Uncertain Supply Chain Manage.*, **7** (2019), 203–214. https://doi.org/10.5267/j.uscm.2018.10.005
4. F. Miguel, M. Frutos, M. Méndez, F. Tohmé, Solving order batching / picking problems with an evolutionary algorithm, in *Communications in Computer and Information Science* (eds. D. A. Rossit, F. Tohmé, G. Mejía Delgadillo), (2021), 177–186. https://doi.org/10.1007/978-3-030-76307-7_14
5. T. Van Gils, K. Ramaekers, K. Braekers, B. Depaire, A. Carisa, Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions, *Int. J. Prod. Econo.*, **197** (2018), 243–261. https://doi.org/10.1016/j.ijpe.2017.11.021
6. R. De Koster, E. S. Van der Poort, M. Wolters, Efficient order batching methods in warehouses, *Int. J. Prod. Res.*, **37** (1999), 1479–1504. https://doi.org/10.1080/002075499191094
7. R. De Koster, T. Le-Duc, K. J. Roodbergen, Design and control of warehouse order picking: A literature review, *Eur. J. Oper. Res.*, **182** (2007), 481–501. https://doi.org/10.1016/j.ejor.2006.07.009
8. F. M. Hofmann, S. E. Visagie, The effect of order batching on a cyclical order picking system, in *Computational Logistics ICCL 2021* (eds. M. Mes, E. Lalla Ruiz, S. Voß), LNCS, **13004** (2021), 252–268. https://doi.org/10.1007/978-3-030-87672-2_17
9. G. Villarreal-Zapata, T. E. Salais-Fierro, J. A. Saucedo-Martínez, Intelligent system for selection of order picking technologies, *Wireless Networks*, **26** (2020), 1–8. https://doi.org/10.1007/s11276-020-02262-x
10. E. Tappia, D. Roy, M. Melacini, R. De Koster, Integrated storage-order picking systems: technology, performance models, and design insights, *Eur. J. Oper.*, **274** (2018), 947–965. https://doi.org/10.1016/j.ejor.2018.10.048
11. Ö. Öztürkoğlu, D. Hoser, A discrete cross aisle design model for order-picking warehouses, *Eur. J. Oper. Res.*, **275** (2018), 411–430. https://doi.org/10.1016/j.ejor.2018.11.037
12. E. H. Grosse, C. H. Glock, The effect of worker learning on manual order picking processes, *Int. J. Prod. Econo.*, **170** (2015), 882–890. https://doi.org/10.1016/j.ijpe.2014.12.018
13. I. Žulj, C. H. Glock, E. H. Grosse, M. Schneider, Picker routing and storage-assignment strategies for precedence-constrained order picking, *Comput. Ind. Eng.*, **123** (2018), 338–347. https://doi.org/10.1016/j.cie.2018.06.015
14. M. C. Chen, H. P. Wu, An association-based clustering approach to order batching considering customer demand patterns, *Omega*, **33** (2005), 333–343. https://doi.org/10.1016/J.OMEGA.2004.05.003
15. S. Henn, S. Y. Koch, G. Wäscher, Order batching in order picking warehouses: a survey of solution approaches, in *Warehousing in the Global Supply Chain* (eds. R. Manzini), Springer, (2012), 105–137. https://doi.org/10.1007/978-1-4471-2274-6_6
16. S. Henn, G. Wäscher, Tabu search heuristics for the order batching problem in manual order picking systems, *Eur. J. Oper. Res.*, **222** (2012), 484–494. https://doi.org/10.1016/j.ejor.2012.05.049
17. K. Rana, Order-picking in narrow-aisle warehouse, *Int. J. Phys. Distrib. Logistics*, **20** (1991), 9–15. https://doi.org/10.1108/09600039010005133

18. H. Hwang, D. Kim, Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system, *Eur. J. Oper. Res.*, **43** (2005), 3657–3670. https://doi.org/10.1080/00207540500151325
19. J. Olmos, R. Florencia, V. García, M. V. González, G. Rivera, P. Sánchez Solís, Metaheuristics for order picking optimization: A comparison among three swarm-intelligence algorithms, in *Technological and Industrial Applications Associated with Industry 4.0* (eds. A. Ochoa Zezzatti, D. Oliva, A. E. Hassanien), (2022), 1–23. https://doi.org/10.1007/978-3-030-68663-5_13
20. A. Scholz, D. Schubert, G. Wäscher, Order picking with multiple pickers and due dates-simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems, *Eur. J. Oper. Res.*, **263** (2017), 461–478. https://doi.org/10.1016/j.ejor.2017.04.038
21. E. Ardjmand, H. Shakeri, M. Singh, O. S. Bajgiran, Minimizing order picking makespan with multiple pickers in a wave picking warehouse, *Int. J. Prod. Econo.*, **206** (2018), 169–183. https://doi.org/10.1016/j.ijpe.2018.10.001
22. M. Masae, C. H. Glock, E. H. Grosse, Order picker routing in warehouses: a systematic literature review, *Int. J. Prod. Econo.*, **224** (2020), 107564. https://doi.org/10.1016/j.ijpe.2019.107564
23. Ç. Cergibozan, A. S. Tasan, Order batching operations: an overview of classification, solution techniques, and future research, *J. Intell. Manuf.*, **30** (2016), 335–349. https://doi.org/0.1007/s10845-016-1248-4
24. Y. C. Ho, Y. Y. Tseng, A study on order-batching methods of order-picking in a distribution centre with two cross-aisles, *Int. J. Prod. Res.*, **44** (2006), 3391–3471. https://doi.org/10.1080/00207540600558015
25. S. Henn, V. Schmid, Metaheuristics for order batching and sequencing in manual order picking systems, *Comput. Ind. Eng.*, **66** (2013), 338–351. https://doi.org/10.1016/j.cie.2013.07.003
26. C. H. Lam, K. L. Choy, G. T. Ho, C. K. Lee, An order-picking operations system for managing the batching activities in a warehouse, *Int. J. Syst. Sci.*, **45** (2014), 1283–1295. https://doi.org/10.1080/00207721.2012.761461
27. T. Van Gils, K. Ramaekers, A. Caris, R. B. M. De Koster, Designing efficient order picking systems by combining planning problems: state-of-the-art classification and review, *Eur. J. Oper. Res.*, **267** (2018), 1–15. https://doi.org/10.1016/j.ejor.2017.09.002
28. C. G. Petersen, An evaluation of order picking routeing policies, *Int. J. Oper. Prod. Manage.*, **17** (1997), 1098–1111. https://doi.org/10.1108/01443579710177860
29. C. Theys, O. Bräysy, W. Dullaert, B. Raa, Using a TSP heuristic for routing order pickers in warehouses, *Eur. J. Oper. Res.*, **200** (2010), 755–763. https://doi.org/10.1016/j.ejor.2009.01.036
30. S. Henn, S. Koch, K. Doerner, C. Strauss, G. Wäscher, Metaheuristics for the order batching problem in manual order picking systems, *Business Res.*, **3** (2010), 82–105. https://doi.org/10.1007/BF03342717
31. R. C. Chen, C. Y. Lin, An efficient two-stage method for solving the order-picking problem, *J. Supercomput.*, **76** (2020), 1–22. https://doi.org/10.1007/s11227-019-02775-z
32. W. Lu, D. McFarlane, V. Giannikas, Q. Zhang, An algorithm for dynamic order-picking in warehouse operations, *Eur. J. Oper. Res.*, **248** (2016), 107–122. https://doi.org/10.1016/j.ejor.2015.06.074

33. P. Kübler, C. H. Glock, T. Bauernhansl, A new iterative method for solving the joint dynamic storage location assignment, order batching and picker routing problem in manual picker-to-parts warehouses, *Comput. Ind. Eng.*, **147** (2020), 106645. https://doi.org/10.1016/j.cie.2020.106645
34. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Publishing Company, Inc, 1989.
35. J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.
36. L. D. Whitley, T. Starkweather, D. Fuquay, Scheduling problems and traveling salesmen: The genetic edge recombination operator, in *Proceedings of the 3rd International Conference on Genetic Algorithms*, (1989), 133–140.
37. A. Wetzel, *Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial Optimization*, University of Pittsburgh, 1983.

AIMS Press