Metaheuristics for Order Picking Optimisation: A Comparison Among Three Swarm-Intelligence Algorithms

Jared Olmos, Rogelio Florencia, Vicente García, Martha Victoria González, Gilberto Rivera, and Patricia Sánchez-Solís

Abstract Nowadays, the Order Picking Problem (OPP) represents the most costly and time-consuming operation of warehouse management, with an average ranging from 50 to 75% of the total warehouse management cost. So, OPP is being analysed to improve logistics operations in companies. The OPP consists of dispatching a set of products, allocated in specific places in a warehouse, based in a group of customer orders. In most traditional warehouses, the optimisation methods of order picking operations are associated with time, whose model is based on the Traveling Salesperson Problem (TSP). The TSP is considered as an NP-Hard problem; thus, the development of metaheuristics approaches is justified. This chapter presents a comparison among three different optimisation metaheuristic approaches that solve the OPP. An analysis is used to evaluate and compare ant colony optimisation, elephant herding optimisation, and the bat algorithm. This study considers the number of picking aisles, the number of extra cross aisles, the number of items in the order, and the standard deviation in both the *x* and *y* axis of the product distribution in the warehouse.

Keywords Order picking problem · Ant colony optimisation · Bat algorithm · Elephant herding optimisation · Traveling salesperson problem · Swarm intelligence

1 Introduction

Nowadays, production and distribution companies are continually searching for innovative ways to generate the highest possible profit. So, the right management of a wide variety of processes throughout the product's lifecycle is essential. This study is focused on the Order Picking Problem (OPP), a phase of the product's lifecycle process, and its resolution using different optimisation algorithms. Order Picking is defined as the recollection of products stored in a warehouse, satisfying a set of

J. Olmos · R. Florencia · V. García · M. V. González · G. Rivera (✉) · P. Sánchez-Solís
Universidad Autónoma de Ciudad Juárez, 32310 Juárez, Chihuahua, Mexico
e-mail: gilberto.rivera@uacj.mx

A. Ochoa-Zezzatti et al. (eds.), *Technological and Industrial Applications Associated With Industry 4.0*, Studies in Systems, Decision and Control 347,
https://doi.org/10.1007/978-3-030-68663-5_13

customer orders, and it is considered as the most costly and time-consuming operation of warehouse management [1, 2].

Consequently, the OPP is a high priority area in warehouse logistics to improve. Optimisation techniques play an essential role when company managers search for minimising cost or maximising profit. Some of the most used optimisation techniques are the ones based on swarm intelligence (SI-based metaheuristics) [3]. These algorithms exploit the collective intelligence and behaviour of self-organised systems such as foraging of social insects or other animals.

The objective of this study is to run, evaluate and compare three SI-based algorithms: Ant Colony Optimisation (ACO), Elephant Herding Optimisation (EHO) and the Bat Algorithm (BA), facing an set of instances.

The remainder of this chapter is organised as follows: In Sect. 2, we present a literature review. Section 3 describes the instances. Section 4 presents the computational experiments and results. Finally, In Sect. 5, we discuss some conclusions and directions for future work.

2 Literature Review

This chapter compares three optimisation algorithms to solve the OPP. Hereafter, in this section, it is presented a literature review of order picking and the three SI-based metaheuristics evaluated.

2.1 Order Picking

In the OPP, there is a set of orders, consisting of a subset of items stored in the warehouse; each order must be supplied [4]. The "picker" is the element that collects the articles in the order, starting from the depot and finishing in the same spot. Beroule in [5] stated that optimisation methods of order picking operations are mostly associated with time and are based on the Traveling Salesperson Problem (TSP). The literature specialised in OPP assumes that the travelled distance is the element that mainly affects the performance; thus, optimising lengths is the main objective [5].

The TSP is defined by a salesman who must find an optimal route that passes through a set of costumers; the vendor must visit each of them only once, and (s)he starts and finishes at the same place. The aim is to find an optimal route in terms of cost, usually focusing on the distance, satisfying the restrictions described above. In our context of order picking, each location of a product is like a customer to visit. The TSP is classified as an NP-hard problem; so, metaheuristics are usually used to face it in a reasonable time [6].

The TSP is formalised as, given a positive integer n—number of cities—and a bidimensional distance matrix c, there exists a tour defined as a sequence of integers where each subsequent integer, taken from the set $[1, 2, 3, \ldots, n]$ appears at least once (the initial and final integers are identical). The tour can be represented as follows:

$$t = (i_1, i_2, i_3, \dots i_{n-1}, i_n, i_1), \tag{1}$$

where t represents the tour and i_j represents the jth costumer; t has a sequence of traversed edges x; so, the optimal solution is sought by minimising the objective function described in Eq. 2.

$$z(x) = \sum_{(i,j)\in x} c_{i,j} \tag{2}$$

Here, x is composed of the ordered pairs of t as follows:

$$x = \langle (i_1, i_2), (i_2, i_3), \dots, (i_{n-1}, i_n), (i_n, i_1) \rangle. \tag{3}$$

2.2 Ant Colony Optimisation

ACO is a well-known approach for solving combinatorial problems such as the OPP. Dorigo et al. [7] proposed this algorithm, and several variants have emerged from it [8–10]. This algorithm is inspired by the behaviour of ant colonies when searching for food supplies. Ants have the ability to produce and release a pheromone to communicate with each other. Ants start exploring the surface at random, emitting traces of pheromones while finding routes to the food supply [11]. Once the pheromone trail is on the surface, subsequent ants follow this trace. However, the pheromone trail tends to disappear over time, which causes shorter routes to be selected by the ants. The chosen pathways receive reinforcement of pheromones from each ant that walks on.

For this reason, more and more ants choose these shorter routes. The probability of selecting a path increases proportionally with the number of ants that walks by that route [12]. Figure 1 represents this behaviour.

In summary, the ACO algorithm contains two rules or phases [14]. The first one is the update of local pheromone while constructing the solutions. The second one occurs when the pheromone is globally updated once all ants have built the solution [15].

The following equation describes the probability of choosing a path by an ant.

$$P_{ij}^{k} = \frac{(\tau_{ij}^{\alpha})\left(n_{ij}^{\beta}\right)}{\sum_{l\in(N_i^k)}\left(\tau_{ij}^{\alpha}\right)\left(n_{ij}^{\beta}\right)}, \text{ if } j \in N_i^k \tag{4}$$

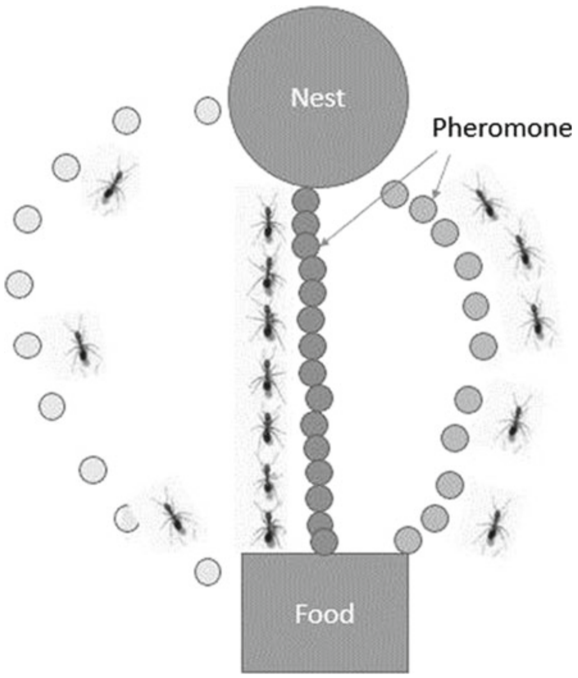


Fig. 1 Representation of the release of pheromone. *Source* [13]

Equation (4) calculates the probability of success of an ant k reaching a position j from a position i. In the first iteration, ants have the same likelihood of getting any node j because of the lack of pheromone trail on the routes. The term τ_{ij} represents the value of the amount of pheromone that exists between i and j, reflecting the attraction of moving from node i to node j [7]. The term n_{ij} is a heuristic value between nodes i and j, and it is inversely proportional to the distance between the two nodes. It is calculated as $n_{ij} = \frac{1}{d_{ij}}$. The relative importance of the pheromone trail is represented by the parameter α; and the parameter β is the relative importance of the heuristic information. Here, N_i^k is the set of nodes that the kth ant has not visited (while standing in position i); so, the end of the iterative process of visiting the nodes occurs when $N_i^k = \emptyset$. The pheromone update on the routes is affected at the end of the process [14].

The evaporation rate is represented by Eq. 3 and is bounded, so its value must be between 0 and 1. The term ρ is a parameter that avoids the unlimited accumulation of pheromone. This process causes the reduction of pheromones, where the ant traffic is lower.

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \forall(i, j) \in L, \quad 0 < \rho < 1 \tag{5}$$

Equation 6 and Eq. 7 are used to update the pheromone trace of the routes given ants' traffic.

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \forall (i, j) \in L \tag{6}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} {}^{1}\!/_{L^k} & \text{if } (i, j) \in T^k, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The pheromone value of the selected routes is increased, favouring the probability of choosing that route. The ants will prefer ways with intense pheromone, which positively reinforces the pheromone trails on the best routes [16]. The term L^k is used to represent the sum of the length of the edges that belongs to a route T^k. According to Eq. 4, the increase of pheromone is inversely proportional to the length of the path, using the $\lim_{L^k \to +\infty} \frac{1}{L^k} = 0$ to represent it. Finally, when this process is repeated—ants travel the graph, the evaporation is performed, and the pheromone is updated—it results in near-optimal solutions [10, 12, 17].

2.3 Elephant Herding Optimisation

Elephant Herds have the following behaviour when searching for food: they divide into two groups, one consisting of male elephants that forage for food in large distances, and the other one composed of female elephants that form groups, performing local searches near their matriarch [18–22]. Once the first group of male elephants finds a food source, the matriarch and the whole group moves toward it.

Wang in [18] lists four main assumptions used to model the elephant's biological behaviour; these assumptions are described below.

1. Each elephant has a particular visual range, which helps them to look out for other animals that might threaten their personal space. Male elephants have a wider visual range due to the instinct of protection they should have, allowing them to move randomly while searching for food. This range, in the algorithm, is calculated by the Euclidean distance.
2. When two elephants see each other, a contest begins to demonstrate who is the strongest one. This action is represented in the algorithm by comparing the fitness value of both elephants. The one with the highest value is considered the strongest one, and the loser must leave the area. Figure 2 illustrates that behaviour.
3. A clan is made up of only one female group, and they are always together.
4. There exists a maximum lifespan in each elephant. When an elephant dies, a new baby must be born. An impressive characteristic is that the gender of this new baby is inherited from the elephant that just died; this keeps balanced the clan's gender.
5. EHO splits elephants into k clans, considering that each jth member in the ith clan follows the movement of the clan's matriarch. The matriarch is the elephant

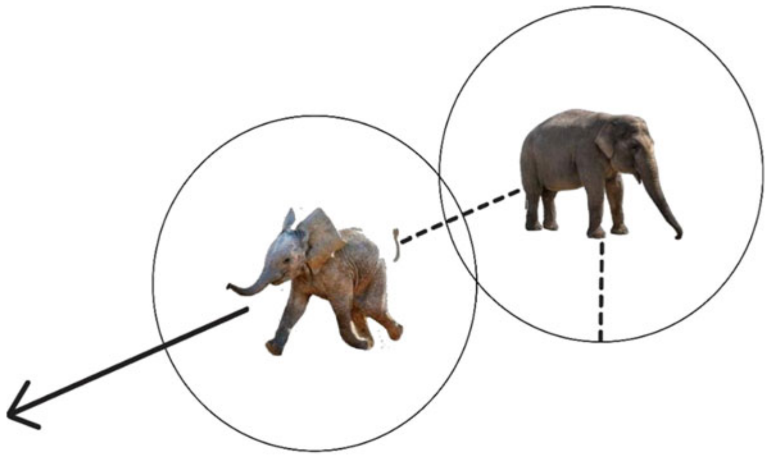


Fig. 2 Elephant with the lowest fitness flees from another elephant with better fitness. *Source* [23]

with the best value in each generation. The EHO algorithm identifies four types of elephants: the baby, the male, the female, and the matriarch.

Matriarch
The oldest living female elephant is considered the leader; this elephant—a.k.a. the matriarch—has the best fitness value, which enables her to dictate the direction that the whole clan must follow. Equation 8 describes the matriarch elephant movement [24, 25].

$$\mathcal{X}_{i,j}^{new} = (\beta)\mathcal{X}_{i}^{centre} \tag{8}$$

In Eq. 8, i is the index of the clan, $\mathcal{X}_{i}^{centre}$ represents the matriarch movement. The parameter β is a binary value that controls the influence of $\mathcal{X}_{i}^{centre} = \{\mathcal{X}_{i,1}^{centre}, \mathcal{X}_{i,2}^{centre}, \ldots, \mathcal{X}_{i,D}^{centre}\}$, defined by [24] as:

$$\mathcal{X}_{i,d}^{centre} = \frac{1}{n_i}\sum_{j=1}^{n_i} \mathcal{X}_{i,j,d}, \tag{9}$$

where $1 \leq d \leq D$ indicates the dth dimension, n_i represents the number of elephants in the clan i, $\mathcal{X}_{i,d}^{centre}$ is the centre, calculated by the average of the whole current solutions in the clan, and $\mathcal{X}_{i,j,d}$ represents the dth dimension of the point of elephant j belonging to the ith clan.

Female Elephants
The female elephants always follow the matriarch staying close to the clan. They perform a local search under the direction of the matriarch of the i clan. Their moving behaviour is represented by [24]:

$$x_{i,j}^{new} = x_{i,j} + \alpha \left(x_i^{best} - x_{i,j}\right) r, \tag{10}$$

where $x_{i,j}^{new}$ represents the new position for elephant j of the ith clan, $x_{i,j}$ represents the last position, x_i^{best} represents the best solution in the ith clan, the parameter $\alpha \in [0, 1]$ indicates the matriarch influence over the clan, and $r \in [0, 1]$ is a random number, used to diversification.

Male Elephants
Male elephants are responsible for searching for food in a more profound sense (randomly). They expand the search scope by exploring the area. The algorithm represents the elephants with the worst fitness movement, m_i, as follows [24]:

$$x_i^{worst} = x_{\min} \cdot \left(x_{\max} - x_{\min}\right) \cdot random() \tag{11}$$

The terms $x_{\max}$ and $x_{\min}$ describe the upper and lower bound, respectively, and a random-number function is used to represent randomness.

Baby Elephants
These baby elephants are born with the same gender of the most recently dead elephant, inheriting its fitness value. Usually, they stay with female elephants until reaching adulthood; then, they must leave the clan if they are male.

2.4 Bat Algorithm

Yang in [26] proposed the BA to solve combinatorial optimisation problems. This algorithm is based on the echolocation behaviour of bats while searching for prey. The author establishes three premises described below:

1. Bats use echolocation to trace their preys and hunt them.
2. There exists a fly randomly made by bats. This fly has a velocity v_i at a certain position x_i and a frequency $f_{\min}$ with a variation of the wavelength λ and loudness A_0 to search for their prey.
3. The parameter of the loudness varies reaching an $A_{\min}$.

With these premises, the author proposes the following equations to model the behaviour of bats

$$v_i^t = v_i^{t-1} + \left(x_i^t - x^*\right) f_i \tag{12}$$

$$f_i = f_{\min} + (f\max - f_{\min})\beta \tag{13}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{14}$$

$$x_{new} = x_{old} + \in A^t \tag{15}$$

Equation 12 helps to describe the random velocity at a specific time.

The term x_i represents a position at a fixed frequency $f_{\min}$. The wavelength varies and the volume A_0 is taken to find the prey. The loudness frequency can vary randomly in many ways. The position x_i, velocity v_i and the frequency f_i are initialised, and Eq. 10 is used to update and find the best solutions. The term β represents a random vector, and x^* represents the current best solution. This process is iterated, updating the loudness and the frequency of the impulse emission as follows [27]

$$A_i^{t+1} = \propto A_i^t \quad r_i^{t+1} = r_i^0\left[1 - \exp(-\gamma t)\right], \tag{16}$$

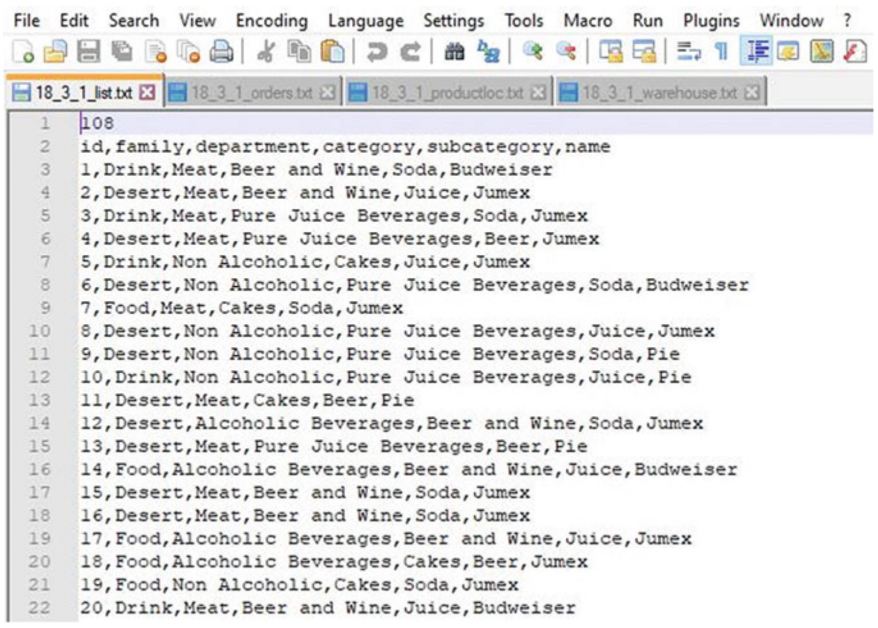
$$A_i^t = 0, r_i^t \to r_i^0, \text{ as } t \to \infty, \tag{17}$$

where $\propto$ and γ are constant.

3 Data Description

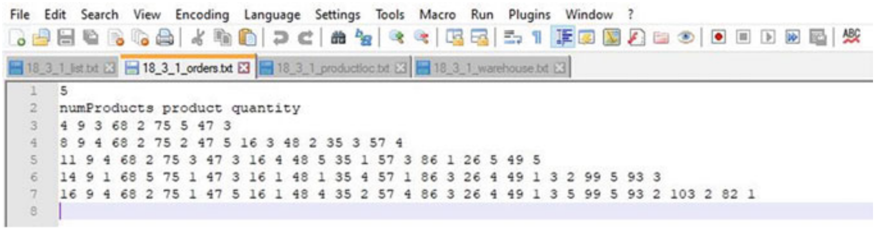
The set of instances used in this work has been adapted from [28]. There are four different files per test instance, which represent different features. The first one is the "list" file, which is a list of the products and their description; it is illustrated in Fig. 3. The second one is the "order" file, which is a set of customer orders specifying the number of products to be ordered, the identifier of the product, and the quantity needed; it is illustrated in Fig. 4. The third file, named "productloc", describes the location of the products in the warehouse, each product is allocated in a single location; it is illustrated in Fig. 5. The final file, named "warehouse", describes the warehouse configuration, and it's the most important one because it includes eight parameters to be set and five data variables that will guide our tests; it is illustrated in Fig. 6. The parameters and their ranges are the following:

1. The number of aisles: Represents the total of corridors in the warehouse.
2. The number of extra cross aisles: Represents the total of additional cross aisles in the warehouse (not considering both the front and the rear corridors).
3. The number of shelves: Represents the total shelves per rack position, the value doesn't vary.
4. Minimum of products required: Represents the number of products needed in the list archive to fill the warehouse.
5. Aisle width: This parameter doesn't vary and is set in three units.
6. Rack depth: This parameter doesn't vary and is set in one unit.
7. Location width: This parameter doesn't vary and is set in one unit.
8. Cross aisle width: This set doesn't vary and is set in one unit.


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
18_3_1_list.txt
18_3_1_orders.txt
18_3_1_productloc.txt
18_3_1_warehouse.txt

```
108
id,family,department,category,subcategory,name
1,Drink,Meat,Beer and Wine,Soda,Budweiser
2,Desert,Meat,Beer and Wine,Juice,Jumex
3,Drink,Meat,Pure Juice Beverages,Soda,Jumex
4,Desert,Meat,Pure Juice Beverages,Beer,Jumex
5,Drink,Non Alcoholic,Cakes,Juice,Jumex
6,Desert,Non Alcoholic,Pure Juice Beverages,Soda,Budweiser
7,Food,Meat,Cakes,Soda,Jumex
8,Desert,Non Alcoholic,Pure Juice Beverages,Juice,Jumex
9,Desert,Non Alcoholic,Pure Juice Beverages,Soda,Pie
10,Drink,Non Alcoholic,Pure Juice Beverages,Juice,Pie
11,Desert,Meat,Cakes,Beer,Pie
12,Desert,Alcoholic Beverages,Beer and Wine,Soda,Jumex
13,Desert,Meat,Pure Juice Beverages,Beer,Pie
14,Food,Alcoholic Beverages,Beer and Wine,Juice,Budweiser
15,Desert,Meat,Beer and Wine,Soda,Jumex
16,Desert,Meat,Beer and Wine,Soda,Jumex
17,Food,Alcoholic Beverages,Beer and Wine,Juice,Jumex
18,Food,Alcoholic Beverages,Cakes,Beer,Jumex
19,Food,Non Alcoholic,Cakes,Soda,Jumex
20,Drink,Meat,Beer and Wine,Juice,Budweiser
```

Fig. 3 List file. *Source* https://homepages.dcc.ufmg.br/~arbex/orderpicking.html adapted by authors

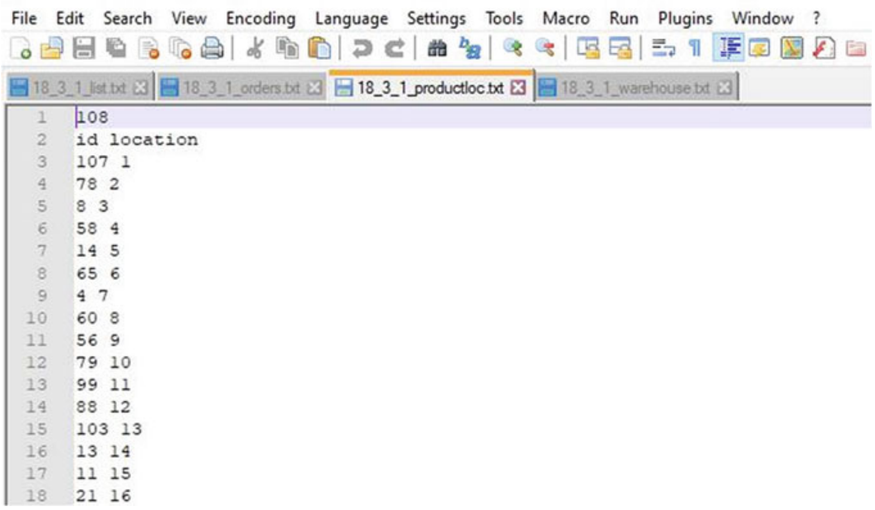

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
18_3_1_list.txt
18_3_1_orders.txt
18_3_1_productloc.txt
18_3_1_warehouse.txt

```
5
numProducts product quantity
4 9 3 68 2 75 5 47 3
8 9 4 68 2 75 2 47 5 16 3 48 2 35 3 57 4
11 9 4 68 2 75 3 47 3 16 4 48 5 35 1 57 3 86 1 26 5 49 5
14 9 1 68 5 75 1 47 3 16 1 48 1 35 4 57 1 86 3 26 4 49 1 3 2 99 5 93 3
16 9 4 68 2 75 1 47 5 16 1 48 4 35 2 57 4 86 3 26 4 49 1 3 5 99 5 93 2 103 2 82 1
```

Fig. 4 Orders file. *Source* https://homepages.dcc.ufmg.br/~arbex/orderpicking.html adapted by authors

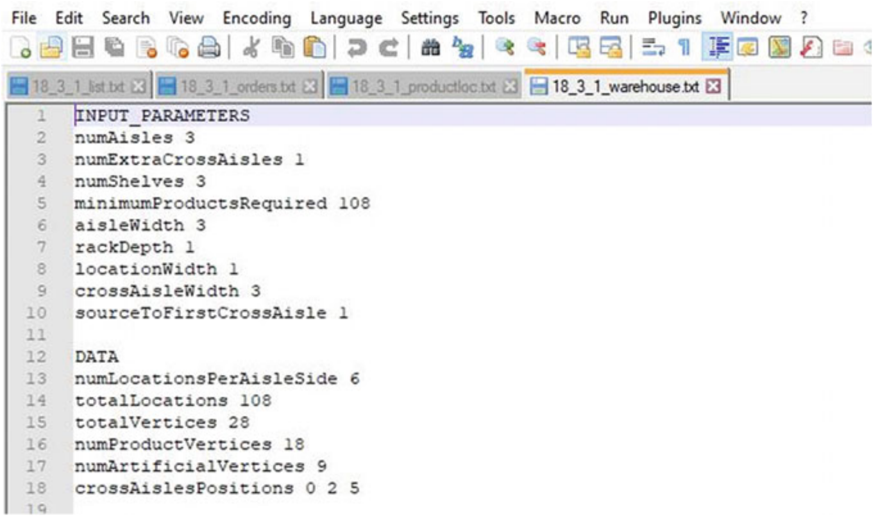
The data variables are the following:

1. The number of locations per aisle: Represents the total location per corridor in the warehouse.
2. Total number of locations: It is calculated by multiplying the number of locations per aisle side times the number of shelves times the double of the number of aisles.
3. The number of vertices: Represents the sum of product vertices—where the picker can get a product—and artificial vertices—the ones used to help build the graph.


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
18_3_1_list.txt
18_3_1_orders.txt
18_3_1_productloc.txt
18_3_1_warehouse.txt

```
108
id location
107 1
78 2
8 3
58 4
14 5
65 6
4 7
60 8
56 9
79 10
99 11
88 12
103 13
13 14
11 15
21 16
```

Fig. 5 Productloc file. *Source* https://homepages.dcc.ufmg.br/~arbex/orderpicking.html adapted by authors


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
18_3_1_list.txt
18_3_1_orders.txt
18_3_1_productloc.txt
18_3_1_warehouse.txt

```
INPUT_PARAMETERS
numAisles 3
numExtraCrossAisles 1
numShelves 3
minimumProductsRequired 108
aisleWidth 3
rackDepth 1
locationWidth 1
crossAisleWidth 3
sourceToFirstCrossAisle 1

DATA
numLocationsPerAisleSide 6
totalLocations 108
totalVertices 28
numProductVertices 18
numArtificialVertices 9
crossAislesPositions 0 2 5
```

Fig. 6 Warehouse file. *Source* https://homepages.dcc.ufmg.br/~arbex/orderpicking.html adapted by authors

4. The number of product vertices: Represents the vertices where the picker can pick a product.
5. The number of artificial vertices: Represents the new vertices added.

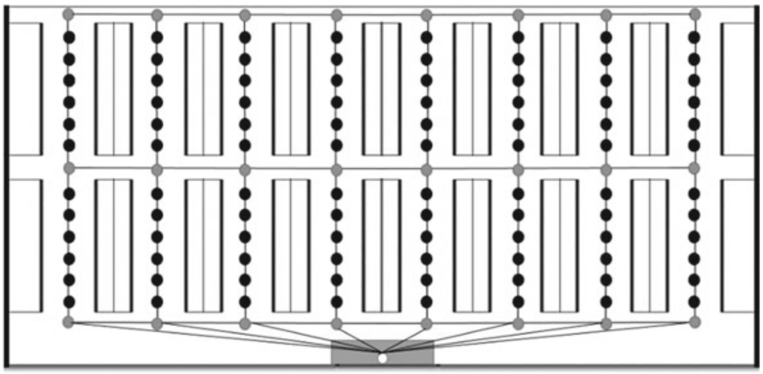


Fig. 7 Graph-oriented model configuration. *Source* https://homepages.dcc.ufmg.br/~arbex/orderpicking.html adapted by authors

6. Cross aisles positions: Represents the vertex number or position of the cross aisles.

The configuration described above is used and transformed into a graph-oriented model representing the pick points based on the locations of the warehouse. This transformation from an OPP to a TSP simplifies the problem-solution process. A Java program is used to transform the configuration. The graph-oriented model considers the possibility of picking up a product of either side of the aisle at every vertex. Figure 7 illustrates an example of this model.

In Fig. 7, it is described as a warehouse with eight aisles, one cross-aisle, and a centred depot. The black nodes represent the product picking vertices, the grey nodes represent the artificial vertices, and the white node represents the depot. With this model, we can quickly compute the distance travelled by a picker given a solution route. Once we have this graph model, a reduction process begins. This reduced graph is built based on the products of each order, eliminating the vertices that have no item in that specific order, and leaving the ones that have a product associated with them. An example of a reduced graph of Fig. 7, with eleven items in the order, is illustrated in Fig. 8. This reduced graph only indicates the product vertices needed to pick the items in the order.

This reduced graph simplifies the solution process of the problem; however, an extra reduction is needed to remove the artificial nodes that we do not need to use to solve the TSP. The elimination process uses a computation of the distance between the two types of nodes, the product, and the artificial ones. The length is evaluated, and then the shortest distance among every possible path from a node *i* to a node *j* is used. This resultant graph is used then to represent the instance as a TSP using the depot node as the starting and closing node of the solution. An adjacency matrix, as described in Fig. 9, is used to represent this graph in the different programs that run the algorithms.

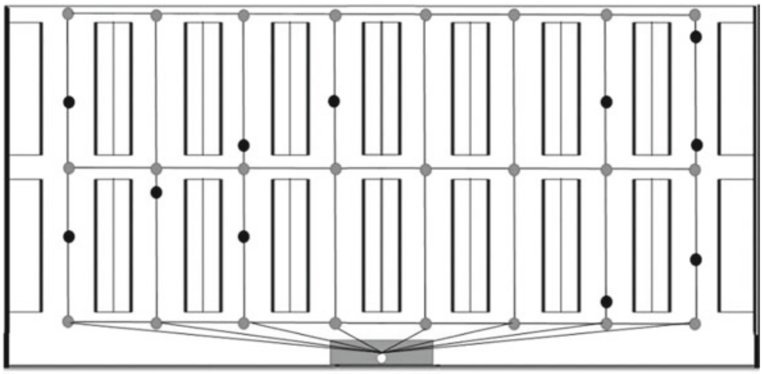
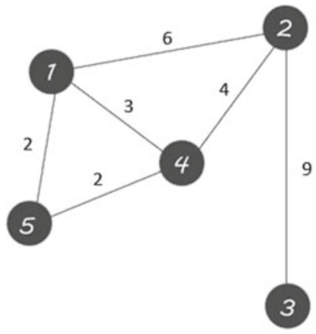
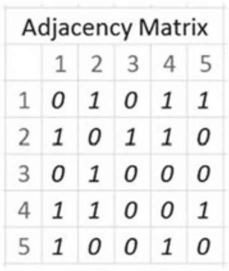


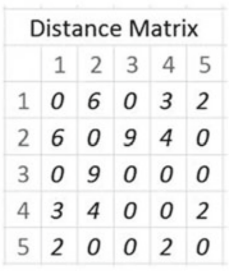
Fig. 8 Example of a simplified graph. *Source* https://homepages.dcc.ufmg.br/~arbex/orderpicking.html adapted by authors





Adjacency Matrix

	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	1	0
3	0	1	0	0	0
4	1	1	0	0	1
5	1	0	0	1	0



Distance Matrix

	1	2	3	4	5
1	0	6	0	3	2
2	6	0	9	4	0
3	0	9	0	0	0
4	3	4	0	0	2
5	2	0	0	2	0

Fig. 9 Example of the adjacency matrix and distance matrix. *Source* https://math.stackexchange.com/questions/1890620/finding-path-lengths-by-the-power-of-adjacency-matrix-of-an-undirected-graph adapted by authors

4 Computational Tests

A computational test evaluates the performance and the runtime of the algorithms to determine which one is the best given specific scenarios. Five parameters vary to generate the test instances. These features are:

1. The number of aisles, from 3 to 15.
2. The number of extra cross aisles, from 1 to 9.
3. The number of products in the order, from 16 to 405.
4. The standard deviation in the horizontal axis of the product's distribution, from 1.12 to 25.19.
5. The standard deviation in the vertical axis of the product's distribution, from 1.13 to 8.82.

The parameter's configuration of each algorithm remained static, using the best setting for each one reported in [6]. For the ACO algorithm, it was established a parameter setting with five ants, 500 iterations, and the evaporation rate of 0.5, $\alpha = 1$, and $\beta = 5$. For the EHO algorithm, it was established 100 elephants, 800 generations, and ten clans. For BA, it was designated a loudness of 0.8, a rate of 0.3, 50 bats, and 100 generations. Table 1 describes the results of the algorithms giving the best result after 30 runs.

The ACO and EHO algorithms were implemented in NetBeans IDE 8.2 running JAVA, while the BA was implemented in Matlab R2013b. It was used a platform with the following specifications: Intel Core i3 7th generation at 2.4 GHz, with 8 Gb of RAM and Windows 10 Home version.

Table 1 summarises the results obtained by each algorithm; Column 1 contains the instance ID, Columns 2–6 present the values of the five features for each instance, and Columns 7–12 includes two data on the performance for each algorithm: distance traversed by the picker in the proposed route and run time. ACO had the best results in every instance run, excluding the first one with 16 products, on which EHO performed better. The BA was only executed on three instances (1, 25, 31) because of the delay in run time (the program was stopped after running three days).

5 Conclusions and Further Research

In this work, it is presented a comparison between three SI metaheuristic for solving an OPP. The OPP was transformed into a TSP to simplify its resolution. An analysis of the performance and the runtime of these three algorithms were made using the best parameter configuration of each one of them. Five features were varied in the instances: the number of aisles, the number of extra cross aisles, the number of items in the order, and the standard deviation in both the x and y axis of the product distribution in the warehouse to evaluate and analyse the algorithms. Based on the results, we conclude that the ACO algorithm is the best in performance and execution time, given the proposed parameter variations. The contribution of this work is to suggest an SI-based metaheuristic to OPP, and this choice is backed by simulation and based on the results of three SI metaheuristics: ACO, BA, and EHO.

As future research, more metaheuristics used in the literature could be compared to the ACO algorithm to identify which implementation is better for this kind of warehouse layout and orders.

Table 1 Results of the different optimisation algorithms

						ACO		EHO		BA	
	Aisles	Cross aisles	Items in the order	Standard deviation on x	Standard deviation on y	Distance	Time of execution	Distance	Time of execution	Distance	Time of execution
1	3	1	16	1.71269768	1.58640054	11	2 s	4	5 s	5	16 m 30 s
2	3	3	32	3.13779638	1.60361235	28	3 s	47	8 s	n. e.	n. e.
3	3	5	49	5.38453315	1.69106294	67	2 s	106	22 s	n. e.	n. e.
4	3	7	65	6.86802093	1.61319678	65	5 s	122	26 s	n. e.	n. e.
5	3	9	81	8.94432566	1.62137169	104	9 s	121	34 s	n. e.	n. e.
6	6	1	32	1.76776695	3.73073006	53	1 s	66	7 s	n. e.	n. e.
7	6	3	65	3.68514429	3.24948221	67	4 s	139	21 s	n. e.	n. e.
8	6	5	97	5.04702968	3.23821548	125	13 s	176	39 s	n. e.	n. e.
9	6	7	130	7.30662453	3.47736526	140	21 s	233	1 m 7 s	n. e.	n. e.
10	6	9	162	8.67408019	3.36805018	165	45 s	421	1 m 52 s	n. e.	n. e.
11	9	1	49	1.74549956	5.14773248	55	2 s	84	13 s	n. e.	n. e.
12	9	3	97	3.53030571	5.32964058	96	11 s	214	47 s	n. e.	n. e.
13	9	5	146	4.74337168	5.09793089	169	29 s	285	1 m 26 s	n. e.	n. e.
14	9	7	194	7.11998551	5.4072342	250	1 m 5 s	550	2 m 24 s	n. e.	n. e.
15	9	9	243	8.85892178	5.13937965	253	2 m 1 s	707	4 m 49 s	n. e.	n. e.
16	12	1	65	1.76327383	6.65376556	103	4 s	140	21 s	n. e.	n. e.
17	12	3	130	3.40668255	6.95535212	135	19 s	290	1 m 9 s	n. e.	n. e.
18	12	5	194	5.14701343	6.86365572	262	1 m 3 s	570	2 m 21 s	n. e.	n. e.
19	12	7	259	7.09508402	7.00288767	321	2 m 24 s	775	6 m 20 s	n. e.	n. e.
20	12	9	324	8.71463059	6.84328023	418	5 m 15 s	1050	8 m 31 s	n. e.	n. e.

(continued)

Table 1 (continued)

						ACO		EHO		BA	
	Aisles	Cross aisles	Items in the order	Standard deviation on x	Standard deviation on y	Distance	Time of execution	Distance	Time of execution	Distance	Time of execution
21	15	1	81	1.72078793	8.12001034	121	9 s	162	30 s	n. e.	n. e.
22	15	3	162	3.29392974	8.69677884	226	43 s	413	1 m 44 s	n. e.	n. e.
23	15	5	243	5.37344979	8.6689457	304	1 m 58 s	701	5 m 17 s	n. e.	n. e.
24	15	7	324	6.77700729	8.81784443	403	4 m 44 s	954	7 m 31 s	n. e.	n. e.
25	15	9	405	8.35836248	8.66205639	532	9 m 42 s	1307	16 m 32 s	no results	8 h 30 m
26	3	1	7	3.35232684	1.13389342	34	1 s	25	2 s	n. e.	n. e.
27	6	3	29	4.96837288	3.51912439	139	1 s	163	7 s	n. e.	n. e.
28	9	5	65	8.38562759	4.95280612	328	9 s	590	24 s	n. e.	n. e.
29	12	3	58	15.4832228	2.85736755	370	5 s	416	19 s	n. e.	n. e.
30	3	5	30	12.9217361	1.59056124	155	2 s	202	7 s	n. e.	n. e.
31	6	1	20	3.99308613	2.55208893	88	1 s	94	5 s	89	33 m 2 s
32	9	7	121	15.5615502	5.06025675	643	28 s	1088	1 m 12 s	n. e.	n. e.
33	12	1	40	3.26441576	7.42759926	208	3 s	206	10 s	n. e.	n. e.
34	15	5	151	11.9861252	8.56440604	718	41 s	1310	1 m 39 s	n. e.	n. e.
35	3	3	26	10.1166276	1.90424627	152	1 s	153	6 s	n. e.	n. e.
36	6	9	130	27.0007884	3.63229403	635	32 s	1110	1 m 11 s	n. e.	n. e.
37	9	7	156	21.4880533	5.35944934	646	48 s	1236	1 m 40 s	n. e.	n. e.
38	12	3	104	10.6889931	7.13164382	492	19 s	806	48 s	n. e.	n. e.
39	15	7	259	20.8340185	8.33078229	1043	3 m 30 s	2230	7 m	n. e.	n. e.

(continued)

Table 1 (continued)

						ACO		EHO		BA	
	Aisles	Cross aisles	Items in the order	Standard deviation on x	Standard deviation on y	Distance	Time of execution	Distance	Time of execution	Distance	Time of execution
40	3	9	72	25.1932748	1.68371562	271	8 s	406	26 s	n. e.	n. e.
41	6	7	115	23.6088549	3.35121823	527	24 s	932	58 s	n. e.	n. e.
42	9	5	130	18.5746491	5.34552126	563	33 s	1034	1 m 17 s	n. e.	n. e.
43	12	3	115	11.7680805	7.04396134	478	24 s	854	58 s	n. e.	n. e.
44	15	1	72	5.26970184	8.51095906	320	9 s	398	26 s	n. e.	n. e.

References

1. Hadi, M.Z., Djatna, T.: Implementation of an ant colony approach to solve multi-objective order picking problem in beverage warehousing with drive-in rack system. In: International Conference on Advanced Computer Science and Information Systems, pp. 137–142. Balic (2017). https://doi.org/10.1109/icacsis.2017.8355024m
2. de Koster, R., Le-Duc, T., Jan-Roodbergen, K.: Design and control of warehouse order picking: a literature review. Eur. J. Oper. Res. **182**(2), 481–501 (2007). https://doi.org/10.1016/j.ejor.2006.07.009
3. Ab-Wahab, M.N., Nefti-Meziani, S., Atyabi, A.: A comprehensive review of swarm optimization algorithms. PLoS ONE **10**(5), 1–36 (2015). https://doi.org/10.1371/journal.pone.0122827
4. Rosenthal, H., Ratliff, D., Arnon, S.: Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper. Res. **31**(3), 507–521 (1983). https://doi.org/10.1287/opre.31.3.507
5. Beroule, B., Grunder, O., Barakat, O., Aujoulat, O.: Order picking problem in a warehouse hospital pharmacy. Sci. Direct IFAC Papers OnLine **50**(1), 5017–5022 (2017). https://doi.org/10.1016/j.ifacol.2017.08.910
6. Ortiz-Zezzatti, A.O., Rivera, G., Gómez-Santillán, C., Sánchez, B.: Handbook of Research on Metaheuristics for Order Picking Optimization in Warehouses to Smart Cities. IGI Global, Hershey (2019). https://doi.org/10.4018/978-1-5225-8131-4
7. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B Cybern. **26**(1), 29–41 (1996). https://doi.org/10.1109/3477.484436
8. Bastiani, S., Cruz-Reyes, L., Fernandez, E., Gómez, C., Rivera, G.: An ant colony algorithm for solving the selection portfolio problem, using a quality-assessment model for portfolios of projects expressed by a priority ranking. In: Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization, vol. 601, pp. 357–373. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17747-2_28
9. Gómez, C., Cruz, L., Schaeffer, E., Meza, E., Rivera, G.: Adaptive ant-colony algorithm for semantic query routing. J. Autom. Mob. Robot. Intell. Syst. **5**(1), 85–94 (2011)
10. Rivera, G., Gómez, C.G., Fernández, E.R., Cruz, L., Castillo, O., Bastiani, S.S.: Handling of synergy into an algorithm for project portfolio selection. In: Castillo, O., Melin, P., Kacprzyk, J. (eds). Recent Advances on Hybrid Intelligent Systems, vol. 451, pp. 417–430. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-33021-6_33
11. Gómez, C., Cruz, L., Schaeffer, E., Meza, E., Rivera, G.: Local Survival Rule for Steer an Adaptive Ant-Colony Algorithm in Complex Systems. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds) Soft Computing for Recognition Based on Biometrics, vol. 312, pp. 245–265. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15111-8_15
12. Fernandez, E., Gomez, C., Rivera, G., Cruz-Reyes, L.: Hybrid metaheuristic approach for handling many objectives and decisions on partial support in project portfolio opti-misation. Inf. Sci. **315**, 102–122 (2015). https://doi.org/10.1016/j.ins.2015.03.064
13. Olmos, J., Florencia, R., López-Ramos, F., Olmos-Sánchez, K.: Improvement of the optimization of an order picking model associated with the components of a classic volkswagen beetle using an ant colony approach. In: Handbook of Research on Metaheuristics for Order Picking Optimization in Warehouses to Smart Cities, pp. 189–210. IGI Global, Hershey (2019). https://doi.org/10.4018/978-1-5225-8131-4.ch010
14. Gomez, C.G., Cruz-Reyes, L., Rivera, G., Rangel-Valdez, N., Morales-Rodriguez, M.L., Perez-Villafuerte, M.: Interdependent Projects selection with preference incorporation. In: García-Alcaraz, J., Alor-Hernández, G., Maldonado-Macías, A., Sánchez-Ramírez, C. (eds) In New Perspectives on Applied Industrial tools and techniques, pp. 253–271. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-56871-3_13

15. De Santis, R., Montanari, R., Vignali, G., Bottani, E.: An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual ware-houses. Eur. J. Oper. Res. **267**(1), 120–137 (2018). https://doi.org/10.1016/j.ejor.2017.11.017
16. Cruz, L., Fernandez, E., Gomez, C., Rivera, G., Perez, F.: Many-objective portfolio optimization of interdependent projects with ‘a priori’ incorporation of decision-maker preferences. Appl. Math. Inf. Sci. **8**(4), 1517–1531 (2014). https://doi.org/10.12785/amis/080405
17. Nemhauger, M., Bellmore, G.: The traveling salesman problem: a survey. Oper. Res. **16**(3), 538–558 (1968). https://doi.org/10.1287/opre.16.3.538
18. Wang, G.-G., Deb, S., Coelho, L.D.S.: Elephant herding optimization. In: International Symposium on Computational and Business Intelligence, pp. 1–5. Bali (2015). https://doi.org/10.1109/iscbi.2015.8
19. Alihodzic, A., Tuba, E., Capor-Hrosik, R., Dolicanin, E., Tuba, M.: Unmanned aerial vehicle path planning problem by adjusted elephant herding optimization. In: 25th Telecommunication Forum (TELFOR), pp. 1–4. Belgrade (2017). https://doi.org/10.1109/telfor.2017.8249468
20. Bukhsh, R., Javaid, N., Iqbal, Z., Ahmed, U., Ahmad, Z., Nadeem-Iqbal, M.: Appliances scheduling using hybrid scheme of genetic algorithm and elephant herd optimization for residential demand response. In: International Conference on Advanced Information Networking and Applications Workshops, pp. 210–217. Cracow (2018). https://doi.org/10.1109/waina.2018.00089
21. Tuba, E., Alihodzic, A., Tuba, M.: Multilevel image thresholding using elephant herding optimization algorithm. In: International Conference on Engineering of Modern Electric Systems, pp. 240–243. Oradea (2017). https://doi.org/10.1109/emes.2017.7980424
22. Tuba, E., Capor-Hrosik, R., Alihodzic, A., Javanovic, R., Tuba, M.: Chaotic elephant herding optimization algorithm. In: World Symposium on Applied Machine Intelligence and Informatics, pp. 000213–000216. Košice (2018). https://doi.org/10.1109/sami.2018.8324842
23. Jimenez, R., Florencia, R., García, V., Lopez, A.: Use of elephant search algorithm to solve an order picking problem in a mobile atelier. In: Handbook of Research on Metaheuristics for Order Picking Optimization in Warehouses to Smart Cities, pp. 161–172. IGI Global, Hershey (2019). https://doi.org/10.4018/978-1-5225-8131-4.ch008
24. Wang, G.-G., Deb, S., Coelho, L., Gao, X.-Z.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. Int. J. Bio-Inspired Comput. **8**(6), 394–408 (2016). https://doi.org/10.1504/IJBIC.2016.10002274
25. Bentouati, B., Chettih, S., El-Sehiemy, R., Wang, G.-G.: Elephant herding optimization for solving non-convex optimal power flow problem. J. Electr. Electron. Eng. **10**(1), 31–36 (2017)
26. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds) In Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, vol 284. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12538-6_6
27. Florencia, R., Sanchez-Solis, J., Carvajal, I., Garcia, V.: Design of an Order Picking Reduce Module Using Bat Algorithm. In Handbook of Research on Metaheuristics for Order Picking Optimization in Warehouses to Smart Cities, pp. 211–225. IGI Global, Hershey (2019). https://doi.org/10.4018/978-1-5225-8131-4.ch011
28. Arbex-Valle, C., Beasley, J.E., da Cunha, A.S.: Modelling and solving the joint order batching and picker routing problem in inventories. In: Cerulli, R., Fujishige, S., Mahjoub, A. (eds) Combinatorial Optimization. ISCO 2016. Lecture Notes in Computer Science, vol 9849. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45587-7_8