Solving Order Batching/Picking Problems with an 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, Villa Regina, Argentina
fmiguel@unrn.edu.ar

[2] Departamento de Ingeniería, Universidad Nacional del Sur e IIESS UNS-CONICET, Bahía Blanca, Argentina
mfrutos@uns.edu.ar

[3] Instituto Universitario de Sistemas Inteligentes SIANI, Universidad de Las Palmas de Gran Canaria ULPGC, Las Palmas, Spain
maximo.mendez@ulpgc.es

[4] Departamento de Economía, Universidad Nacional del Sur e INMABB UNS-CONICET, Bahía Blanca, Argentina
ftohme@criba.edu.ar

Abstract. We present an evolutionary algorithm to solve a combination of the Order Batching and Order Picking problems. This integrated problem consists of selecting and picking up batches of various items requested by customers from a storage area, given a deadline for finishing each order according to a delivery plan. We seek to find the plan that minimizes the total cost of picking the goods, proportional to the time devoted to traverse the storage facility, grabbing the good and leaving it at the dispatch area. Earliness and tardiness induce inefficiency costs due to the excess use of space or breaching the delivery contracts. The results of running the algorithm compare favorably to those reported in the literature.

Keywords: Order Batching · Order Picking · Evolutionary algorithm

1 Introduction

A critical factor in the operational performance of the internal and external logistics of a firm involves the optimization of processes in distribution centers. Moving the items inside those centers, receiving, locating, selecting and collecting them are some of those processes [1, 2]. The selection and collection of items are the main activities carried out in storage facilities. They amount to picking up the right number of items requested by the customers and then taking them to the area in which orders are prepared [3]. The goods are classified, regrouping the units in each batch to prepare the individual orders [4, 5]. Most cases involve also marking, labelling and boxing up the goods into indivisible parcels. The process finishes once each of those parcels are checked out, loaded on the delivery trucks and finished the necessary documents. In this paper we focus on the optimization of the selection and collection of requested items.

2 Problem Description and Literature Review

We can distinguish three problems concerning the operations in storage facilities. The first one is the allocation of goods to different storage positions. The second involves the grouping of items in batches for their collection. The third problem is that of scheduling the sequence of pick-ups of goods and taking them to the dispatch area [6–8]. In this paper we focus on the integrated treatment of the second and third problems, critical for the efficiency of operations since they generate most of the costs of the operations on the floor of the storage facility, being intensive in the use of manpower [9, 10]. This combined problem starts with the arrival of the orders from different customers, detailing the amounts, specifications and availability dates at the merchandise dispatch area. The whole procedure is optimized by choosing the plan that minimizes the operational costs of the operations leading to satisfy the requests in due time of the batches, consisting of different goods for disparate customers [11–13]. Delays in complying with the plan create costs of breaching the contracts with customers. On the other hand, an early finishing of the plan creates costs of crowding the dispatch area, blocking flows of activity and increasing processing times for new orders [11]. Formally, the problem integrates the Order Batching Problem (OBP) and the Order Picking Problem (OPP) [14–16]. OBP consists in finding the amount and size of the batches of items, readying them for the pick-up team [13]. This requires to take into account the capacity of the team and the time at which each item must be available for finishing at the delivery area [17–19]. OPP consists in identifying optimal navigation plans around the sites where the items are stored [20–22]. From now on we denote this integrated problem as OBP+OPP.

A thorough review of the literature on OBP and OPP can be found in [2], while [1, 21, 23] review the heuristics for solving OPP. [8] presents two ways of solving OPP, using Ant Colony Optimization and Iterated Local Search. Other meta-heuristics applied to related routing problems can be found in [15]. [4] uses a clustering approach to solve OBP taking into account demand patterns instead of the distances covered by each sequence of visits. Other heuristics for OBP can be found in [7]. An integer programming approach to OBP is presented in [13] were the visit sequences are estimated and the problem is solved with a heuristic based on fuzzy logic. In turn [11] uses a multiple genetic algorithm for OBP+OPP. This latter contribution uses flexible time windows for the delivery time of each order. We adopt this methodology, but using an evolutionary algorithm with a specific chromosome covering different batches. This algorithm yields results that improve over those obtained at the instances and lay-out of [11].

3 Formulation of the Problem

Let $\mathcal{P} = \{1, \ldots, nIt\}$ be the set of items, where nIt is the amount of different goods. Each item has a unit weight, defining a corresponding class $\mathcal{W} = \left\{w_1, \ldots, w_p, \ldots, w_{nIt}\right\}$. Each customer i makes a single request, involving a list of items $\mathcal{P}_i$. Then, the number of customers is the same as the number of requests, $nReq$, being the set of customers $\mathcal{I} = \{1, \ldots, i, \ldots, nC\}$. Each request has a finishing time, defining a class $\mathcal{T} = \left\{t_1, \ldots, t_{\mathrm{i}}, \ldots, t_{nReq}\right\}$. Let $\mathcal{L} = \left\{\ell_0, \ell_1, \ldots, \ell_{\mathrm{p}}, \ldots, \ell_{nIt}\right\}$ be the positions on the floor of the store, where ℓ_0 is the dispatch area while the others correspond to the placements of the different items. So, for an item $p \in \mathcal{P}$, its position $\ell_{\mathrm{p}} = \left(x_p, y_p\right)$ corresponds to its coordinates in the floor. Given a pair of positions ℓ_{p} and $\ell_{\mathrm{p'}}$, if there is an open direct path between them, we define a distance $D_{lp,lp'}$. We denote with $\mathcal{P}_{\mathrm{r}}$ the class of items in a batch r (the goods in $\mathcal{P}_{\mathrm{r}}$ may or not correspond to the requests of a single customer). $\mathcal{R} = \{1, .., \mathrm{r}, \ldots, |\mathcal{R}|\}$ is the class of total batches to be picked-up. $\mathcal{S}_r = \mathrm{s}_1, \ldots, \mathrm{s}_\mathrm{u}, \ldots \mathrm{s}_{|\mathcal{S}_r|}$ is the sequence of positions to be visited to complete batch r, where s_u is the u-th position to be visited and $|\mathcal{S}_\mathrm{r}|$ is the amount of different items in batch r. $q_{i,p} \in Q$ denotes the total number of units of item p requested by customer i. Then $\mathcal{Q}_i = \sum_{p \in \mathcal{P}_i} q_{i,p}$ is the total number of units in the request of customer i and $\mathcal{Q}_p = \sum_{i \in \mathcal{I}} q_{i,p}$ the total number of units requested of item p. Analogously, we define $\mathcal{Q}_\mathrm{r}$ to be the total number of units in batch r. $\mathcal{K} = \{1, \ldots, |\mathcal{K}|\}$ denotes the class of pick-up teams. Each team as a maximum carrying capacity Cap. Then, the integrated problem OBP+OPP is defined on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ are nodes denoting the storage positions of the items in $\mathcal{P}$, plus two copies of the initial node (the position of the dispatch area). In turn, $\mathcal{A}$ represents all the feasible direct paths between nodes in $\mathcal{V}$. Each such edge $(h, l) \in \mathcal{A}$ has an associated time t_{hl} defined as the length of the distance between positions h and l divided by the speed of the pick-up team v (i.e. $t_{hl} = D_{h,l}/v$). Each edge has also an associated monetary cost per time unit ς. t_{pick} is the mean time to pick any item once reached its corresponding position. A Boolean variable x_{hlkr} is 1 if and only if item h is picked up right before item l by the picking team k in the sequence for batch r, where $h, l \in \mathcal{V}$, $k \in \mathcal{K}$ and $r \in \mathcal{R}$. Another Boolean variable is $y_{hkr} = 1$ if and only if the pick-up team k grabs item h for batch r, where $h \in \mathcal{V}$, $k \in \mathcal{K}$ and $r \in \mathcal{R}$. Then the formal presentation of OBP+OPP is as follows [3, 11]:[1]

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

s.t.:

$$\sum\nolimits_{p \in \mathcal{P}_r} \left(q_p \cdot w_p\right) \cdot y_{pkr} \leq Cap, \ \forall k \in \mathcal{K}, \ r \in \mathcal{R} \tag{2}$$

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

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

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

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

[1] We denote with $\{0, n+1\}$ the start and end at the dispatch area.

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

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

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

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

The goal (1) is the minimization of the total money cost of the time spent in collecting the batches plus a penalty for failing to meet the agreed-on deadlines for finishing the requests. The first term represents the cost of the time devoted to picking up the goods and taking them to the dispatch area. A penalization to earliness or tardiness in getting the items in time to the dispatch area is defined as follows. α is the earliness penalty per unit time while β is the corresponding unit time fine for tardiness. E_i is the time length of earliness in the fulfillment of the request of customer i and T_i corresponds to tardiness: $E_i = max\{0, t_i - c_i\}$ and $T_i = max\{0, c_i - t_i\}$, where c_i is the actual finishing time of the request of i (i.e. the time at which all the items in the request are finally prepared in the dispatch area). The list of constraints (2) indicates that the total weight carried by a team cannot exceed its capacity. The restrictions in (3) mean that no storage position should be visited more than once for the preparation of a given batch. (4) ensures that all pick-up teams start and end at the dispatch area. Restrictions (5) and (6) preserve the orderly sequence of pick-ups. (7) means that the requested amounts of each item p are picked-up. Analogously, (8) indicates that the amounts requested by customer i are picked-up. Finally, (9) and (10) indicate that variables x_{hlkr} and y_{hkr} are Boolean.

4 Solution Method

The problem presented in the previous section belongs to the NP-Hard complexity class. This means that, in practice, it can be solved analytically only in very small instances. Therefore, to find solutions in polynomial time we need to use heuristic methods. We propose here an evolutionary algorithm based on the usual integer representation used to solve combinatorial problems. In this representation we consider a chromosome consisting of two genomes. We can see how this works in a very simple instance, in which we codify the solution for three requests of four items. Table 1 shows that request 1 involves 3 units (one unit of item A and two units of C); request 2 asks for one unit of A, one of B, two of C and one of D while request 3 demands two units of B and two of D.

To codify this we consider two rows, one for items (identifying to which requests they belong) and another indicating cumulative amounts (Table 2).

A chromosome with two genomes captures the codification in Table 2 (see Table 3).

Chromosome: {[G1] [G2]}

Genomes:

[G1]: Sequence in which items will be picked up.

[G2]: Amount of items picked up in [G1] for each batch.

Number of entries:

Table 1. Example for OBP + OPP.

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

Table 2. Codification.

Items (corresponding request)											
A(1)	A(2)	B(2)	B(3)	B(3)	C(1)	C(1)	C(2)	C(2)	D(2)	D(3)	D(3)
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
1	2	3	4	5	6	7	8	9	10	11	12

Table 3. Chromosome = Genome 1 + Genome 2

Cumulative pick ups →	1	2	3	4	5	6	7	8	9	10	11	12
Genome 1												
[G1] →	6	7	9	10	11	1	3	2	12	4	8	5
	batch 1					batch 2		batch 3				
Genome 2												
[G2] →	5	7	12	–	–	–	–	–	–	–	–	–

[G1]: Total number of requested items $\sum_{i \in \mathcal{I}} \mathcal{Q}_i$.

[G2]: Total number of requested items $\sum_{i \in \mathcal{I}} \mathcal{Q}_i$ (non-null entries correspond to batches).

The sequence in which items of each batch are picked up is the following. Batch 1: $\{6 \rightarrow 7 \rightarrow 9 \rightarrow 10 \rightarrow 11\}$, Batch 2: $\{1 \rightarrow 3\}$ and Batch 3: $\{2 \rightarrow 12 \rightarrow 4 \rightarrow 8 \rightarrow 5\}$. This means that, for instance for Batch 2, that the pick-up team has to go to the first position on Table 2 (A(1)) and then to third position (B(2)) to finally take to the dispatch area the two picked up units of items A and B. Figure 1 shows the navigation path of the three pick-up teams.

This visual representation ensures to warrant the satisfaction of constraints (2)–(8). In the initial stage of the algorithm a population is generated at random. The procedure iterates, by selecting the fittest individuals, the winners in repeated tournaments among *k*

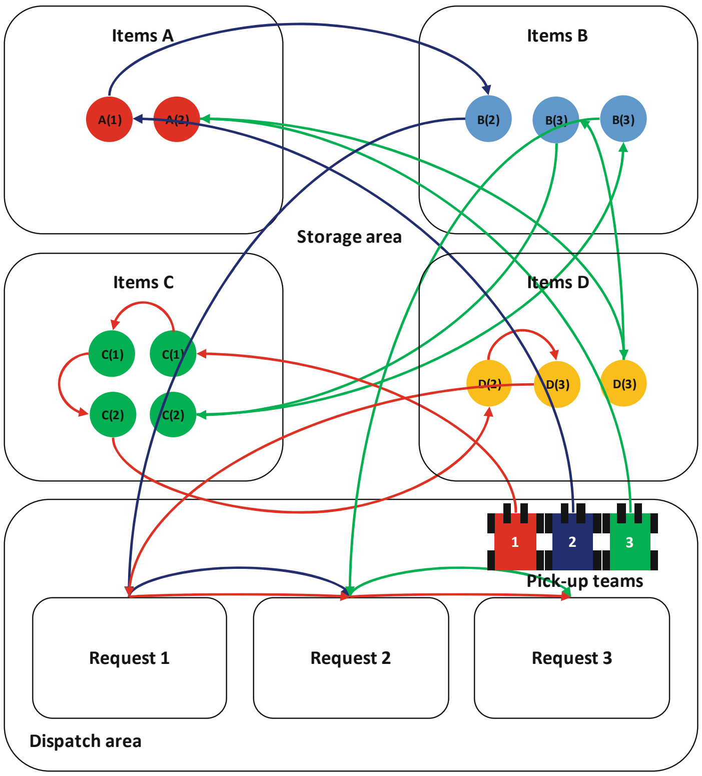

Fig. 1. Navigation paths of the pick-up teams

individuals chosen at random from the current population [24]. Then, the *edge recombination* Operator [25] and the *mutation of insertion* operator [26] are applied on Genome 1. Genome 2 is completed according to the capacity of pick-up teams. A cost criterion caps the number of iterations. Figure 2 presents the pseudo-code of the algorithm.

5 Computational Experiment

In order to evaluate the quality of the solutions and the performance of the algorithm we run it on the instances presented in [11]: a medium size instance (DS1/M1), a large one (DS2/M1) and a very large instance (DS3/M1) (Table 4).

We assume that the number of units of an item p requested by a customer i follows a uniform distribution between 1 and 10, i.e. $q_{i,p} \sim U(1, \ldots, 10)$.The number of different items requested by a customer i is assumed to obey to a normal distribution with mean 10 and standard deviation 5, that is $|\mathcal{P}_i| \sim N(10, 5)$. The finishing time of a request of customer i, follows a uniform distribution, on discrete periods of time measured in seconds between 10:00 am and 06:00 pm, that is, $t_i \sim U(36000, \ldots, 64800)$. The unit weight of each item p, obeys also to a uniform distribution ranging between 8 and 24 kg., i.e. $w_p \sim U(8, \ldots, 24)$. With respect to the pick-up teams we assume that their average

```
1: Load Input % information of requests, lay-out and parameters of the algorithm.
2: nLot ← nLotMin
3: while nLot < nLotMax
4:  t ← 0;
5:  P(t) ← InitPop(Entrada);
6:  FitP(t) ← EvalPop(P(t));
7:  For t ← 1 a MaxNumGen
8:      Q(t) ← SelecBreeders (P(t), FitP(t));
9:      Q(t) ← Crossover(Q(t));
10:         Q(t) ← Mutation(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: nLot ← nLot +1;
18: end
```

Fig. 2. Pseudo-code of the algorithm

Table 4. Features of the instances [11]

	DS1/M1	DS2/M1	DS3/M1
Number of requests	40	80	200
Number of different items	80	160	300
Total average weight (kg.)	13.704	37.152	158.784
Capacity of pick-up teams (kg.)	10.000	10.000	20.000

speed is $v = 2m/s$, will the mean grabbing time of items is $t_{pick} = 15$ s while the cost of displacement per unit of time is $\varsigma = \$0,05$ with Cap depending on the instance. For the penalty fees we consider $\alpha = \$0.5$ and $\beta = \$1$. We adopt also the strategy of bounding the search space presented in [11] as to compare our results with those reported in that article. This amounts to define lower and upper bounds on the number of batches: $|\mathcal{R}|_{min} \leq |\mathcal{R}| \leq |\mathcal{R}|_{max}$. These bounds are $|\mathcal{R}|_{min} = \left(\varphi_1 \cdot \sum_{p\in\mathcal{P}} w_p\right)/Cap$ y $|\mathcal{R}|_{max} = \left(\varphi_2 \cdot \sum_{p\in\mathcal{P}} w_p\right)/Cap$ where φ_1 and φ_2 are such that $\varphi_2 \geq \varphi_1$. The number of iterations is capped at 500, the size of the population is 150, the number of participants in the tournament is 2, while $\varphi_1 = 2$ and $\varphi_2 = 4$, with a crossover probability of 0.9, a mutation probability of 0.15, and a 5% of the population in the elite. We ran the experiment on a PC with an Intel Core i7 3.00 GHz processor and a RAM of 8 GB.

6 Results

We compare the results of running the algorithm on the three instances DS1/M1, DS2/M1 and DS3/M1 to those obtained in [11] (Table 5, D_{Total}: total distance in meters; n_{Batch}: optimal number of batches; $D_{Average}$: average distance in meters; D_{σ}: standard deviation in meters; T_{fail}: tardiness and earliness in seconds; C_{Total}: total cost in \$; $T_{CPU\ average}$: running time in seconds). We ran each instance 30 times.

Table 5. Results

	Results from [11]			Evolutionary algorithm		
	DS1/M1	DS2/M1	DS3/M1	DS1/M1	DS2/M1	DS3/M1
D_{Total}	1304.00	3569.00	16945.00	1266.00	3215.00	14548.00
n_{Batch}	8.00	11.00	27.00	8.00	12.00	27.00
$D_{Average}$	163.00	324.46	627.59	158.25	267.91	538.81
D_{σ}	3.70	25.17	18.69	2.90	23.41	21.75
T_{fail}	1181.00	4704.00	15481.00	1158.70	4801.91	15621.31
C_{Total}	1092.60	3104.83	9207.13	1090.78	2961.89	8898.01
$T_{CPUaverage}$	753.60	2629.90	5785.50	605.70	2121.90	4474.20

We can see that D_{Total} improves with our algorithm in all three instances (DS1/M1, DS2/M1 and DS3/M1, 2.91%, 9.92% and 14.15%, respectively). On instances DS1/M1 and DS3/M1, n_{Batch} was the same as in [11], while for DS2/M1 it was larger. For instance DS1/M1, T_{fail} improves 1.89%, while on DS2/M1 and DS3/M1 it worsens 2.08% and 0.91% respectively. Finally, $T_{CPU\ average}$ was lower in all instances. Figure 3 depicts the percentages of improvement or worsening with respect to the results in [11].

7 Conclusions

We presented an evolutionary algorithm to solve in an integrated way a combination of the Order Batching and the Order Picking problems. The algorithm operates on a novel way of representing the chromosome with two genomes, allowing incorporating directly specific knowledge of the problem, yielding a more flexible treatment of the search space while at the same time providing an explicit representation of the constraints. We ran the algorithm on simulated instances of different sizes. We found that the algorithm improved in general the results presented in [11]. Future work involves addressing the problem in a cross-dock platform and under a multi-objective perspective.

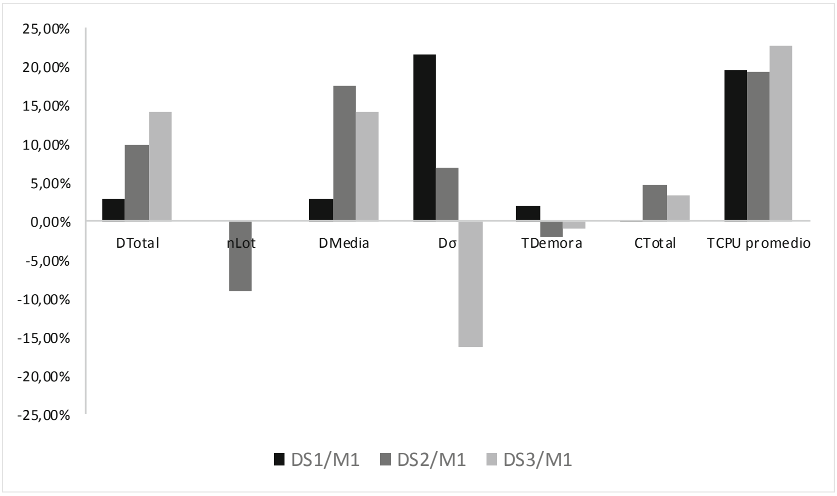

Fig. 3. Percentage of improvements over the results in [11].

References

1. De Koster, R., Van der Poort, E.S., Wolters, M.: Efficient order batching methods in warehouses. Int. J. Prod. Res. **37**(7), 1479–1504 (1999)
2. De Koster, R., Le-Duc, T., Roodbergen, K.J.: Design and control of warehouse order picking: a literature review. Eur. J. Oper. Res. **182**(2), 481–501 (2007)
3. Miguel, F., Frutos, M., Tohmé, F., Rossit, D.A.: A memetic algorithm for the integral OBP/OPP problem in a logistics distribution center. Uncertain Supply Chain Manag. **7**(2019), 203–214 (2019)
4. Chen, M.C., Wu, H.P.: An association-based clustering approach to order batching considering customer demand patterns. Omega **33**(4), 333–343 (2005)
5. Henn, S., Koch, S., Wäscher, G.: Order batching in order picking warehouses: a survey of solution approaches. In: Manzini, R. (ed.) Warehousing in the Global Supply Chain, pp. 105–137. Springer, London (2012). https://doi.org/10.1007/978-1-4471-2274-6_6
6. Henn, S., Schmid, V.: Metaheuristics for order batching and sequencing in manual order picking systems. Comput. Ind. Eng. **66**(2), 338–351 (2013)
7. Henn, S., Wäscher, G.: Tabu search heuristics for the order batching problem in manual order picking systems. Eur. J. Oper. Res. **222**(3), 484–494 (2012)
8. Henn, S., Koch, S., Doerner, K., Strauss, C., Wäscher, G.: Metaheuristics for the order batching problem in manual order picking systems. Bus. Res. **3**(1), 82–105 (2010)
9. Hwang, H., Kim, D.: Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. Eur. J. Oper. Res. **43**(17), 3657–3670 (2005)
10. Rana, K.: Order-picking in narrow-aisle warehouse. Int. J. Phys. Distrib. Logist. **20**(2), 9–15 (1991)
11. Tsai, C.Y., Liou, J.J., Huang, T.M.: Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. Int. J. Prod. Res. **46**(22), 6533–6555 (2008)
12. Grosse, E.H., Glock, C.H.: The effect of worker learning on manual order picking processes. Int. J. Prod. Econ. **170**(C), 882–890 (2015)

13. Lam, C.H., Choy, K.L., Ho, G.T., Lee, C.K.: An order-picking operations system for managing the batching activities in a warehouse. Int. J. Syst. Sci. **45**(6), 1283–1295 (2014)
14. Van Gils, T., Ramaekers, K., Braekers, K., Depaire, B., Carisa, A.: Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. Int. J. Prod. Econ. **197**, 243–261 (2018)
15. Ho, Y.C., Tseng, Y.Y.: A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. Int. J. Prod. Res. **44**(17), 3391–3471 (2006)
16. Scholz, A., Schubert, D., Wäscher, G.: 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**(2), 461–478 (2017)
17. Ardjmand, E., Shakeri, H., Singh, M., Bajgiran, O.S.: Minimizing order picking makespan with multiple pickers in a wave picking warehouse. Int. J. Prod. Econ. **206**, 169–183 (2018)
18. Öztürkoğlu, Ö., Hoser, D.: A discrete cross aisle design model for order-picking warehouses. Eur. J. Oper. Res. **275**(2), 411–430 (2018)
19. Tappia, E., Roy, D., Melacini, M., De Koster, R.: Integrated storage-order picking systems: Technology, performance models, and design insights. Eur. J. Oper. **274**(3), 947–965 (2018)
20. Lu, W., McFarlane, D., Giannikas, V., Zhang, Q.: An algorithm for dynamic order-picking in warehouse operations. Eur. J. Oper. Res. **248**(1), 107–122 (2016)
21. Petersen, C.G.: An evaluation of order picking routeing policies. Int. J. Oper. Prod. Manag. **17**(11), 1098–1111 (1997)
22. Žulj, I., Glock, C.H., Grosse, E.H., Schneider, M.: Picker routing and storage-assignment strategies for precedence-constrained order picking. Comput. Ind. Eng. **123**, 338–347 (2018)
23. Theys, C., Bräysy, O., Dullaert, W., Raa, B.: Using a TSP heuristic for routing order pickers in warehouses. Eur. J. Oper. Res. **200**, 755–763 (2010)
24. Wetzel, A.: Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial Optimization. University of Pittsburgh, Pittsburgh (1983)
25. Whitley, L.D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesmen: the genetic edge recombination operator. In: Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, pp. 133–140 (1989)
26. Fogel, D.B.: An evolutionary approach to the traveling salesman problem. Biol. Cybern. **60**(2), 139–144 (1988)