Scientific Research Publishing

American Journal of Operations Research, 2018, 8, 474-485
http://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

Genetic Algorithm for Scattered Storage Assignment in Kiva Mobile Fulfillment System

Mengcheng Guan*, Zhenping Li

School of Information, Beijing Wuzi University, Beijing, China
Email: *gmengcheng@163.com

How to cite this paper: Guan, M.C. and Li, Z.P. (2018) Genetic Algorithm for Scattered Storage Assignment in Kiva Mobile Fulfillment System. *American Journal of Operations Research*, **8**, 474-485.
https://doi.org/10.4236/ajor.2018.86027

Received: October 20, 2018
Accepted: November 25, 2018
Published: November 28, 2018

Abstract

Scattered storage means an item can be stored in multiple inventory bins. The scattered storage assignment problem based on association rules in Kiva mobile fulfillment system is investigated, which aims to decide the pods for each item to put on so as to minimize the number of pods to be moved when picking a batch of orders. This problem is formulated into an integer programming model. A genetic algorithm is developed to solve the large-sized problems. Computational experiments and comparison between the scattered storage strategy and random storage strategy are conducted to evaluate the performance of the model and algorithm.

Keywords

Scattered Storage Assignment, Kiva Mobile Fulfillment System, Association Rules, Genetic Algorithm

1. Introduction

Storage assignment strategy is a critical factor affecting warehouse operation efficiency. In the traditional picker-to-parts warehouse, pickers pick items according to the picking list, and the results of storage assignment determine the walking distance of the pickers and the order completion time. In recent years, Kiva mobile fulfillment system, in which pods are carried by robots to workstations, has been gradually applied [1], [2]. Storage assignment strategy in Kiva mobile fulfillment system is still critical. The problem is to determine the pods for each item to put on so as to minimize the number of pods to be moved when picking a batch of orders.

As in Figure 1, Kiva mobile fulfillment system consists of inventory pods, robots, and workstations. Items are stored on the inventory pods, and each pod has several inventory bins. When an order arrives, it is assigned to one of the

Figure 1. Kiva mobile fulfilment system layout.

workstations, and one or more pods is selected to complete the order. Target pods are moved by robots to workstations for workers to pick items from the pods. One robot can only carry one pod to a workstation at a time. After the items are picked up by workers in workstations, robots return the pods to the storage location.

There are many operation problems in Kiva mobile fulfillment system, such as storage assignment problem, order batching problem, robot path planning problem, and pods selecting problem. Storage assignment problem is an essential problem in Kiva mobile fulfillment system. Different from storage assignment problem in the traditional picker-to-parts warehouse, one item can be stored in multiple inventory bins in Kiva mobile fulfillment system, which is called scattered storage. The scattered storage assignment in Kiva mobile fulfillment system is one-to-many form while in the traditional picker-to-parts warehouse is one-to-one form.

In Kiva mobile fulfillment system, many items are frequently ordered in one order, and there are association rules among the items. If the items that are often ordered in one order are stored in the same pod, we can complete the orders

with fewer pods to be moved. With that in mind, this article studied the scattered storage assignment problem based on association rules in Kiva mobile fulfillment system. An integer programming model is formulated to solve this problem, and a genetic algorithm is proposed to solve the model. The validity of the proposed model and algorithm is validated by comparing with random storage strategy.

The rest of the article is organized as follows. The related work of storage assignment is reviewed in Section 2. The problem description is presented and the integer programming model is formulated in Section 3. A genetic algorithm is proposed in Section 4. The computational experiments are presented in Section 5. Lastly, Section 6 concludes the study.

2. Literature Review

Roodbergen and De Koster [3] presented four approaches to improve warehouse operation efficiency, one of which was storage assignment problem. Storage assignment problem in the traditional picker-to-parts warehouse has been intensively investigated [4] [5] [6] [7]. De Koster [8] described five frequently used types of storage assignment: random storage, closest open location storage, dedicated storage, full turnover storage and class-based storage. Random storage assignment assigns items to empty location randomly. The advantage of random storage is it results in a high space utilization, and the disadvantage is the increased travel distance of pickers [9]. Closest open location storage assigns items to the first empty location that is encountered by the worker. Hausman [10] argued that closest open location storage and random storage performed similarity if items are moved by full pallets only. Dedicated storage stores items in a fixed location. The advantage of this strategy is that pickers are familiar with item's location, and the disadvantage is it results in a low space utilization [11]. Full turnover assigns locations according to item's turnover, and the items with the highest turnover are assigned to the location nearest to the depot. Heskett [12] proposed the cube-per-order index (COI) rule to minimize the labour cost, which was an early storage policy of full turnover storage. Later, Malmborg and Bhaskaran [13] provided an optimal solution based on COI. Class-based storage groups items into classes and divides the locations into classes. Then assign items to the locations of the corresponding class.

Plenty of research has been done on the storage assignment with considering the item's relationship. Several items that are frequently ordered in one order are suggested to be stored nearby to reduce the order completion time [14]. Ming-Huang Chiang *et al.* [15] proposed a heuristic method based on association rules mining for storage assignment problem. Bindi *et al.* [16] compared different storage assignment rules in a warehouse with correlated storage policy and presented a heuristic algorithm. Xiao and Zheng [17] studied the correlated storage assignment with bill of material information (BOM). Li *et al.* [18] proposed a genetic algorithm for the dynamic storage assignment problem (DSAP) with the consideration of the item affinity and ABC classification. Chiang [19]

proposed an adaptive approach for storage assignment based on data mining. [20] studied the storage assignment and order batching problem in Kiva mobile fulfillment systems, they proposed a model for storage assignment problem and solved by CPLEX. However, CPLEX cannot solve the large-sized storage assignment problems within a reasonable time. There are few studies on the scattered storage assignment, and [21] introduced the scattered storage strategy and studied the scattered storage assignment problem.

Previous studies almost considered the storage assignment problem in the traditional picker-to-parts warehouse, but there were relatively few studies devoted to the large-sized storage assignment problem in Kiva mobile fulfillment system. Therefore, this article focuses on the scattered storage assignment problem based on association rules in Kiva mobile fulfillment system and proposes an algorithm for large-sized storage assignment problems.

3. Problem Description and Integer Programming Model

3.1. Problem Description

Scattered storage assignment problem based on association rules in Kiva mobile fulfilment system is considered in this article. We assume that all pods are empty at first. One pod has Q inventory bins to store items. The number of item types and the number of inventory bins that each item requires are given. Historical orders are given so that we can obtain item's similarity and verify the proposed model and algorithm. We assume that the amount of an item in one inventory bin can satisfy the demand of the item in an order. The scattered storage assignment aims to maximize the item's similarity in all pods so that we can minimize the number of pods to be moved when picking the orders.

3.2. Integer Programming Model

This section proposes a model for the scattered storage assignment problem based on association rules in Kiva mobile fulfillment system.

The parameters are defined in **Table 1**.

Association rules can be described as follows: let $I = \{I_1, I_2, ..., I_m\}$ be an itemset, let D, the task-relevant data, be a set of database transactions where each transaction T is a nonempty itemset such that $T \subseteq I$. We say that a transaction T contains A, a set of some items in I, if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$. Support count is the count of transactions in D that contains A and B. Association rules are considered if they satisfy both a minimum support threshold and a minimum confidence threshold [22]. Thus, we can obtain item similarity matrix S after mining association rules.

Given items which are frequently ordered in one order are stored in the same pod results in fewer pods to be moved to complete the picking operation. The objective function of the integer programming model is set to maximize the total item similarity in all pods.

The integer programming model is similar to the model proposed by [20], it is

Table 1. Parameters of the model.

	Indices
i, j	index of items, $i, j = 1, ..., N$
o	index of orders, $o = 1, ..., O$
m	index of pods, $m = 1, ..., M$
	Parameters
N	total number of items
O	total number of orders
M	total number of pods
Q	number of inventory bins in one pod; Q=8 in this article
B_i	number of inventory bins for the item i needed to be assigned
S_{ij}	similarity value between the item i and j, it is equal to the support count between item i and j
S	item similarity matrix
	Decision variables
x_{im}	$x_{im} = 1$ if the item i is put on the pod m; $x_{im} = 0$, otherwise
y_{im}	number of inventory bins that item i occupies on the pod m

unreasonable for all items to be stored on all inventory bins, which means that no bin is vacant. In our model, we allow empty inventory bins to exist. The integer programming model is as follows:

$$\max \sum_{i} \sum_{j>i} \sum_{m} s_{ij} x_{im} x_{jm} \tag{1}$$

Subject to:

$$\sum_{i} y_{im} \le Q, \quad \forall m \tag{2}$$

$$\sum_{m} y_{im} = B_i, \quad \forall i \tag{3}$$

$$y_{im} \le Q x_{im}, \quad \forall i, m \tag{4}$$

$$y_{im} \ge x_{im}, \quad \forall i, m \tag{5}$$

$$x_{im} \in \{0,1\}, y_{im} \ge 0, \quad \forall i, m \tag{6}$$

Equation (1) is the objective function, it is to maximize the total item similarity in all pods. Constraint (2) ensures that inventory bins that a pod is assigned not exceed Q. Constraint (3) ensures that the number of inventory bins that item i needed can be satisfied. Constraint (4) indicates that the number of bins that item i occupies in the pod m is no more than Q only if it is selected to store item i. Constraint (5) indicates that item i can be stored in pod m only if it is selected to store item i. Constraint (6) is a basic constraint.

4. Algorithm for the Scattered Storage Assignment Problem

The problem can be decomposed into two stages. First, association rules of items can be obtained according to the historical orders. Second, storage assignment results are optimized by the proposed genetic algorithm.

4.1. Mining Association Rules

There are many existing algorithms to obtain association rules, such as Apriori, FP-growth. In this article, we choose Apriori algorithm to obtain association rules without setting thresholds. So, we can obtain the similarity matrix S.

$$S = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{pmatrix}$$

We set the support count between the item i and j to be the similarity between the item i and j and set $s_{ii} = 0$.

4.2. Genetic Algorithm

Genetic algorithm is a family of computational models inspired by evolution, which was first used by [23]. In this article, we propose a single-parent genetic algorithm and design brand-new operators with considering the characteristic of the scattered storage assignment problem.

4.2.1. Genetic Coding

We need to present the relationship of items and pods so the individual can be described as a matrix in the following:

$$\begin{bmatrix} 2 & 3 & 1 & 2 & 4 \\ 4 & 2 & 5 & 1 & 3 \\ 3 & 5 & 4 & 5 & 1 \\ 2 & 3 & 3 & 4 & 5 \end{bmatrix}$$

The rows of the individual stand for the pods and the columns stand for the inventory bins. The elements stand for item types. This individual represents there are five items need to be assigned to four pods. Each pod has five inventory bins. The first row of this individual represents that item 2, item 3, item 1 and item 4 are stored in the first pod while item 2 occupies two inventory bins.

4.2.2. Generate Initial Population

The initial population is generated randomly according to the inventory bins that each item requires. It ensures the diversity of the population.

4.2.3. Fitness Function

The genetic algorithm is always used for maximization problem, and the objective function of the integer programming model is also a maximization function. So, the fitness function in our algorithm is the objective function of the integer programming model:

$$F(x) = \sum_{i} \sum_{j>i} \sum_{m} s_{ij} x_{im} x_{jm}$$

4.2.4. Selection Method

We adopt the roulette-wheel method, which uses a probability distribution for selection in which the selection probability of a given string is proportional to its fitness. Moreover, elitism preserving strategy is applied to improve the algorithm's efficiency.

4.2.5. Crossover Operator

A brand-new crossover operator is designed to solve the problem for the reason that it can make the objective function as large as possible when a variety of item types are stored on one pod. So, we choose two pods to exchange the duplicate items on them. For each individual, a single-parent genetic crossover operation is performed.

For example, item 2 appears two times on the first pod, item 5 appears two times on the third pod, and item 3 appeared two times on the fourth pod, then randomly select two duplicate items to exchange. We might as well exchange item 2 on the first pod and item 5 on the third pod. Equation (7) shows the example of the crossover operator.

$$\begin{bmatrix} 2 & 3 & 1 & 2 & 4 \\ 4 & 2 & 5 & 1 & 3 \\ 3 & 5 & 4 & 5 & 1 \\ 2 & 3 & 3 & 4 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 5 & 3 & 1 & 2 & 4 \\ 4 & 2 & 5 & 1 & 3 \\ 3 & 2 & 4 & 5 & 1 \\ 2 & 3 & 3 & 4 & 5 \end{bmatrix} \tag{7}$$

If the two pods do not have duplicate items, we conduct the crossover operation by the method of mutation operator which is described in the following.

4.2.6. Mutation Operator

Performing the mutation operator ensures that new search space is reached, which crossover operator cannot do. Because the less fit individuals are discarded, mutation operator can avoid the algorithm falling into a local optimal solution. We randomly exchange two items on different pods as the mutation operator. Equation (8) shows the example of the mutation operator. We perform mutation operation by exchanging item 5 on the first pod and item 3 on the second pod.

$$\begin{bmatrix} 2 & 3 & 1 & 2 & 4 \\ 4 & 2 & 5 & 1 & 3 \\ 3 & 5 & 4 & 5 & 1 \\ 2 & 3 & 3 & 4 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 5 & 1 & 2 & 4 \\ 4 & 2 & 3 & 1 & 3 \\ 3 & 5 & 4 & 5 & 1 \\ 2 & 3 & 3 & 4 & 5 \end{bmatrix} \tag{8}$$

The specific steps of the genetic algorithm are as follows:

Step 1: Set population size P, the number of iteration G and mutation probability.

Step 2: Generate initial population t.

Step 3: Calculate the fitness of each individual.

Step 4: Perform selection, crossover operation and mutation operation.

Step 5: Obtain the population $t+1$.

Step 6: Determine whether the iteration reaches G and returns Step 3 if it not reaches.

5. Computational Experiments

5.1. Examples Generation

We generated random examples according to the characteristic of the scattered storage in Kiva mobile fulfilment system. The setting was as follows:

- Set the number of orders as O.
- Set the average number of item types in order as l and set maximum of it as H.
- The number of item types in order obeys $P(l)$, Poisson distribution with parameter l.
- Set the number of item types as N.
- The inventory bins that each item requires obey $U(a, b)$, uniform distribution with parameter a and b.
- Set the number of inventory bins of each pod as 8, $Q = 8$.
- Calculate the number of pods as M.

Table 2 shows the different parameter settings of different cases.

5.2. Comparison between Proposed Algorithm and CPLEX

Experiments were conducted to compare the results obtained by proposed algorithm and global optimal objective value solved by CPLEX. Item similarity matrix was obtained by 30% of the historical orders. The values obtained by CPLEX with a time limit of 1800 s. The integer programming model was implemented by CPLEX 12.8. The comparison between the proposed algorithm and CPLEX is shown in Table 3.

The results showed that the objective values of the proposed algorithm were very close to the global optimal values since the largest gap was 1.34% in the first three examples. It implies that the proposed algorithm is an effective method for solving the scattered storage assignment problem. Moreover, the time that proposed algorithm used was shorter than CPLEX used. The CPLEX method

Table 2. Experiment parameter settings.

o	□	H	N	a	b	Q	M
1000	4	10	10	2	5	8	5
1000	4	10	20	2	5	8	10
1000	4	10	30	2	5	8	13
1000	4	10	50	2	5	8	21
1000	4	10	100	2	5	8	46
1000	4	10	150	2	5	8	70

Table 3. Comparison between the proposed algorithm and CPLEX.

Parameters						GA		CPLEX		Gap[a]
M	N	$[a,b]$	Q	P	G	Obj1	Time/s	Obj2	Time/s	
5	10	[2,5]	8	5	50	5473	0.0625	5473	0.1658	0.00%
10	20	[2,5]	8	50	700	4945	10.8906	4945	548.5877	0.00%
13	30	[2,5]	8	100	1000	4569	42.3906	4631	1483.6850	−1.34%
21	50	[2,5]	8	50	3000	3674	116.7500	3379	1800.0000	8.73%
46	100	[2,5]	8	50	3000	3883	711.3334	2722	1800.0000	42.65%
70	150	[2,5]	8	50	3000	3459	1080.3070	949	1800.0000	264.49%

[a](Obj1 − Obj2)/Obj2 * 100%.

cannot obtain an optimal solution within 1800s when the number of item types increases to 50. When the number of item types increases to 150, the objective value of proposed genetic algorithm is 264.49% better than the value of CPLEX and spent less time than CPLEX.

5.3. Comparison between the Scattered Storage Assignment and Random Storage Assignment

To evaluate the validity of the scattered storage assignment, some experiments are conducted to calculate the number of pods to be moved to complete 70% of the historical orders, which are compared with random storage strategy.

Table 4 showed that the scattered storage assignment strategy based on association rules can decrease the number of pods to be moved by an average 34.66%.

5.4. Sensitivity Analysis

We consider that the number of inventory bins of one pod (Q) and item's distribution may affect the number of pods to be moved when picking orders, so the sensitivity analyses are conducted to verify the performance of the proposed model with respect to $N = 10$.

From Figure 2(a) and Figure 2(b), it can be found that the objective value of the proposed model tends to increase while the number of pods to be moved tends to decrease as Q increase. As shown in Figure 2(c) and Figure 2(d), the objective value of the proposed model tends to increase while the number of pods to be moved tends to decrease as parameters of uniform distribution increase.

6. Conclusion

The scattered storage assignment problem based on association rules in Kiva mobile fulfillment system was studied in this article that determined which item to store on which pods, with the objective of maximizing item similarity in our proposed integer programming model. For solving the problem, a genetic

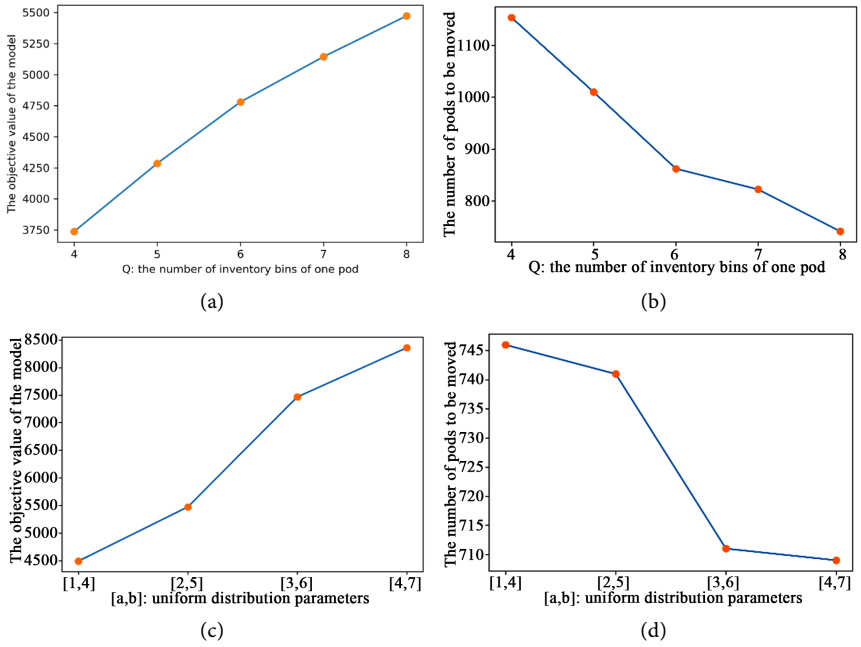

Figure 2. Sensitivity analyses with respect to $N = 10$.

Table 4. The number of pods to be moved of our strategy and random storage assignment strategy.

Parameters				Our strategy	Random strategy	Decrease percentage
M	N	$[a,b]$	Q	The number of pods to be moved	The number of pods to be moved	
5	10	[2,5]	8	741	1084	31.64%
10	20	[2,5]	8	1000	1423	29.73%
13	30	[2,5]	8	990	1524	35.04%
21	50	[2,5]	8	1180	1874	37.03%
46	100	[2,5]	8	1244	1958	36.47%
70	150	[2,5]	8	1299	2102	38.02%
Average				1076	1661	34.66%

algorithm was proposed. Computational experiments, comparison between our algorithm and CPLEX and comparison between the scattered storage strategy and random storage strategy are conducted to obtain the following results: 1) The proposed algorithm has a good performance both on solution quality and time, compared with CPLEX. 2) When the items types are increasing, the scattered storage assignment cannot be solved by CPLEX but by our proposed algorithm. 3) The scattered storage assignment based on association rules can significantly reduce the number of pods to be moved with the comparison of random storage strategy.

Acknowledgements

This work is supported by the National Natural Science Foundation of China

[grant number 71771028], Beijing Municipal University's High-level Innovation Team Construction Project [grant number IDHT20180510] and Beijing Intelligent Logistics Cooperative Innovation Centre.

Conflicts of Interest

No potential conflict of interest was reported by the authors.

References

[1] Wurman, P.R., D'Andrea, R. and Mountz, M. (2008) Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *Ai Magazine*, **29**, 9-20.

[2] Enright, J. and Wurman, P.R. (2011) Optimization and Coordinated Autonomy in Mobile Fulfillment Systems. *Autom. Action Plan. Automated Action Planning for Autonomous Mobile Robots from* 2011 *AAAI Workshop*, San Francisco, California, USA, 7 August 2011, 1-1.

[3] Roodbergen, K.J. and De Koster, R. (2001) Routing Order Pickers in a Warehouse with a Middle Aisle. *European Journal of Operational Research*, **133**, 32-43. https://doi.org/10.1016/S0377-2217(00)00177-6

[4] Accorsi, R., Baruffaldi, G. and Manzini, R. (2018) Picking Efficiency and Stock Safety: A Bi-Objective Storage Assignment Policy for Temperature-Sensitive Products. *Computers & Industrial Engineering*, **115**, 240-252. https://doi.org/10.1016/j.cie.2017.11.009

[5] Quader, S. and Castillo-Villar, K.K. (2018) Design of an Enhanced Multi-Aisle Order-Picking System Considering Storage Assignments and Routing Heuristics. *Robotics and Computer-Integrated Manufacturing*, **50**, 13-29. https://doi.org/10.1016/j.rcim.2015.12.009

[6] Davis, C.J. (2017) Using Self-Organizing Maps to Cluster Products for Storage Assignment in a Distribution Center.

[7] Chang, Y., Ma, W. and Wu, Y. (2017) Principles of Storage Location Assignment in Multi-Tier Shuttle Warehouse System. *Chinese Automation Congress* (*CAC*), Jinan, 20-22 October 2017, 5658-5662. https://doi.org/10.1109/CAC.2017.8243792

[8] de Koster, R., Le-Duc, T. and Roodbergen, K.J. (2007) Design and Control of Warehouse Order Picking: A Literature Review. *European Journal of Operational Research*, **182**, 481-501. https://doi.org/10.1016/j.ejor.2006.07.009

[9] Il-Choe, K. and Sharp, G. (2014) Small Parts Order Picking: Design and Operation.

[10] Hausman, W.H., Schwarz, L.B. and Graves, S.C. (1976) Optimal Storage Assignment in Automatic Warehousing Systems. *Management Science*, **22**, 629-638. https://doi.org/10.1287/mnsc.22.6.629

[11] De Koster, M.B.M. and Neuteboom, A.J. (2001) The Logistics of Supermarket Chains. Elsevier, Doetinchem, The Netherlands.

[12] Heskett, J.L. (1963) Cube-Per-Order Index: A Key to Warehouse Stock Location. *Transportation and Distribution Management*, **3**, 27-31.

[13] Malmborg, C.J. and Bhaskaran, K. (1990) A Revised Proof of Optimality for the Cube-Per-Order Index Rule for Stored Item Location. *Applied Mathematical Modelling*, **14**, 87-95. https://doi.org/10.1016/0307-904X(90)90076-H

[14] Frazele, E.A. and Sharp, G.P. (1989) Correlated Assignment Strategy Can Improve Any Order-Picking Operation. *Industrial Engineering*, **21**, 33-37.

[15] Ming-Huang Chiang, D., Lin, C.P. and Chen, M.C. (2014) Data Mining Based Storage Assignment Heuristics for Travel Distance Reduction. *Expert Systems*, **31**, 81-90. https://doi.org/10.1111/exsy.12006

[16] Bindi, F., Manzini, R., Pareschi, A. and Regattieri, A. (2009) Similarity-Based Storage Allocation Rules in an Order Picking System: An Application to the Food Service Industry. *International Journal of Logistics Research and Applications*, **12**, 233-247. https://doi.org/10.1080/13675560903075943

[17] Xiao, J. and Zheng, L. (2010) A Correlated Storage Location Assignment Problem in a Single-Block-Multi-Aisles Warehouse Considering BOM Information. *International Journal of Production Research*, **48**, 1321-1338. https://doi.org/10.1080/00207540802555736

[18] Li, J., Moghaddam, M. and Nof, S.Y. (2016) Dynamic Storage Assignment with Product Affinity and ABC Classification—A Case Study. *The International Journal of Advanced Manufacturing Technology*, **84**, 2179-2194. https://doi.org/10.1007/s00170-015-7806-7

[19] Chiang, D.M.-H., Lin, C.-P. and Chen, M.-C. (2011) The Adaptive Approach for Storage Assignment by Mining Data of Warehouse Management System for Distribution Centres. *Enterprise Information Systems*, **5**, 219-234. https://doi.org/10.1080/17517575.2010.537784

[20] Xiang, X., Liu, C. and Miao, L. (2018) Storage Assignment and Order Batching Problem in Kiva Mobile Fulfilment System. *Engineering Optimization*, **50**, 1941-1962. https://doi.org/10.1080/0305215X.2017.1419346

[21] Weidinger, F. (2018) A Precious Mess: On the Scattered Storage Assignment Problem. In: *Operations Research Proceedings* 2016. Springer, Cham, 31-36. https://doi.org/10.1007/978-3-319-55702-1_5

[22] Han, J.W., Kamber, M., Han, J., Kamber, M. and Pei, J. (2012) Data Mining: Concepts and Techniques.

[23] Holland, J.H. (1992) Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Vol. 69.