Evaluation of the Joint Impact of the Storage Assignment and Order Batching in Mobile-Pod Warehouse Systems

Ning Yang

School of Economics and Management, University of Science and Technology, Beijing, China

Correspondence should be addressed to Ning Yang; lizzyelizabeth2021@163.com

Received 11 January 2022; Revised 6 March 2022; Accepted 19 March 2022; Published 11 April 2022

1. Introduction

Robotic systems are an inevitable trend of highly developed smart logistics, and mobile-pod warehouse systems are representative of this trend. Traditional warehouse systems mostly use an inefficient picker-to-parts order-picking mode by which a lot of picker moves are wasted to get to the inventory slots. To improve the operational efficiency and save labor, automated warehouses like the AS/RS (Automated Storage and Retrieval System) were introduced. These systems often use a parts-to-picker picking mode, in which their space utility and operational efficiency can be improved [1]. However, the AS/RS warehouses constitute a large fixed investment, require a long installation period, and have a poor flexibility. Mobile-pod warehouse systems were developed in response to these limitations. A typical representative of these systems is the robotic mobile fulfillment system (RMFS, the pioneer of which is called the Kiva system [1] and now Amazon Robotics). Figure 1 shows such an application scenario which has a concise structure that is based on mobile robots. The robots are components of the parts-to-picker system and deliver products directly to the pickers. Therefore, this can save the picker’s travel time along aisles and improve the warehouse productivity [2]. Because the system is operational flexible, RMFS has the virtue of incremental scalability, as well as ease of built-in redundancy [3]. Such systems have other advantages, including saving on the cost of labor and improvements in the operational efficiency and order-picking accuracy. This makes them well suited for the tight deadlines of e-commerce [4, 5]. For relevant research about RMFS, the reader is referred to the survey work of da Costa Barros and Nascimento [6].

Decreasing the number of round trips of total robots is of great significance. The features of the mobile-pod warehouse
system are its flexible mobile pods (also called racks), swift robots, and great central computer control. During operation, a group (can be even thousands) of robots travel on the floor. They park underneath, lift the pods off the ground, and move along the landmarks on the floor to the workers. They then return each pod to an available position in the storage area after items on it are picked. The central computer commands all of the robots, allowing them to move pods accurately while preventing collisions and deadlocks [7]. In this operating environment, most of the power is consumed during the robots’ moving process. Due to dynamically changing pod positions, it is difficult to account for the real mileage of the robots. Therefore, instead of minimizing the mileage, most studies aim to decrease the number of round trips by all robots without loss of accuracy.

Order picking is the process of retrieving products from storage or buffer areas in a warehouse to fulfill the customer orders, typically, in a strict time window. Many factors must be taken into consideration to improve the picking efficiency, such as storage assignment, order batching, and picker routing [6, 8, 9]. A storage assignment method is the rules which can be used to assign products to storage locations. Order batching is the process of clustering a set of orders into several groups, and orders in each group are then aggregated and picked simultaneously (i.e., handled as super orders). In a mobile-pod warehouse system, the storage assignment obviously affects the order-picking workload. Figure 2 illustrates a toy example in which 5 orders are pending for picking out from a mobile-pod warehouse (a more detailed description of the order-picking process will be elaborated in Section 3.1). The item types in each order are as follows: \( O_1 = \{1, 2, 3, 4\} \), \( O_2 = \{2, 3, 5\} \), \( O_3 = \{1, 2, 3\} \), \( O_4 = \{2, 3\} \), and \( O_5 = \{4, 5\} \). Two storage allocation policies are compared: the single storage and mixed storage which represent storage of single type of items and multiple types of items (e.g., according to the correlation of items). We assume that the number of items storage in each pod is always sufficient without considering the replenishment issues. In Figure 2(a), each order is handled separately; therefore, the number of round trips by robots is 14 for single storage, but it is only 7 for mixed storage, since one pod can provide multiple kinds of items. In Figure 2(b), the five orders are batched into two groups (by assuming that the capacity for each batch is 7 items type). Therefore, the number of round trips by robots is 10 for single storage and only 4 for mixed storage. We conclude from this example that order batching and storage assignment policies significantly affect the efficiency of order picking in mobile-pod warehouses.

Although storage assignment and order batching have been extensively investigated by previous studies (as elaborated in the next section), the dedicated research for mobile-pod warehouse systems is still insufficient. Therefore, we fill in this gap in this study. Our contribution can be threefold:

1. We analyze the combined effects of storage assignment and order-packing methods in an emerging robotic warehousing system, which were not well addressed before.
2. Based on the characteristics of mobile storage and parts-to-picker processing mode, we recast the two decision problems as a unified clustering model by using item similarity and order similarity as distance metrics, respectively.
3. We compared six policy combinations of order batching and storage assignment by numeric experiments, and the conclusions can shed light on industrial applications.

This paper is organized as follows. In the next section, we review the literature on the storage assignment and order batching problems. In Section 3, we describe the problem we are studying, including its settings and basic assumptions and develop the similarity coefficients used in this paper. In Section 4, clustering models are built to solve the storage assignment and order batching problems, respectively. An evaluation model was built to compute the amount of time saved using our model. Section 5 analyzes the saving results of the different policies. Finally, in Section 6, we draw conclusions and provide insights on further research.

### 2. Literature Review

In the RMFS, storage assignment and order batching ultimately affect the efficiency of order picking. Each order may have more than one order line. Petersen and Aase [9] compared various methods to measure their performance, and their results showed that order batching performs better when the order capacity is small. They used the sensitive analysis method to verify the different factors that influence the performance. Based on this, we consider two main factors, the order batching and the storage assignment, in the order-picking process to find a more effective picking method.

#### 2.1. Order Batching

Recent trends show that customer orders have changed from few-and-large orders to many-and-small orders, so it is important to discuss how to handle order combinations and batching. Generally, when the order size is large, a picker tends to finish a single order picking in one route; otherwise, the picker will need to finish more than one order to reduce the total travel distance and gain increased efficiency [10, 11]. Order batching is the process for grouping a set of orders into a small number of subsets so each subset can be retrieved by a single picking tour.

It is not realistic to solve a large order batching problem optimally due to the enormous number of computations...
involved, which is why many researchers tend to choose the heuristic algorithm to solve it. de Koster et al. [12] evaluated two groups of heuristic algorithms, seed algorithms and time savings algorithms, and their performance was evaluated by two picker routing strategies (S-shape and Largest Gap). The researchers compared the travel distance using these picking strategies. The seed algorithm proposed by Elsayed and Stern [13] was based on the total number of goods at a specific location in two orders in an automatic warehouse system. They designed heuristic rules and then proposed and evaluated 24 kinds of order batching methods. Ho et al. [14] extended the work on the seed algorithm and evaluated 11 kinds of seed order selection methods and 14 kinds of seed order adding methods. These were compared with the previous methods.

Some more sophisticated methods are also used in this field, for example, the metaheuristic algorithms and exact solution approaches (for simplified scenarios). Hsu et al. [15] used the genetic algorithm to solve the order batching problem in the traditional warehouse to find the shortest travel route. Tsai et al. [16] proposed two other genetic algorithms to solve the order batching and picker routing problems simultaneously. Gademann and Velde [17] proposed the branch-and-price optimization algorithm for the order batching problem. They proved that it was an NP-hard problem, but it was solvable using polynomials if no batch contained more than two orders. Boysen et al. [18] for the first time pointed out that both the order processing sequence and the pod arrival sequence at the picking station notably affect the order-picking efficiency in a robotic part-to-picker system. They provided a mathematical formulation of the joint problem and proposed a decomposition-based solution approach that incorporates two dynamic programming (DP) algorithms. Füßler and Boysen [19] treated the processing sequence of orders in a single picking station to reduce the number of storage bins transferred from the storage system to fulfill orders. They dealt with the problem with a general heuristic method. Lam et al. [20] studied an order-picking operations system to assist in formulating an order-picking plan and batch-handling sequence in a manpower warehouse. The study integrates a mathematical model and a fuzzy logic technique to divide the receiving orders into batches and prioritize the batch-handling sequence for picking. Wu et al. [21] considered an automated stereoscopic warehouse system and studied the optimization of order picking; specifically, they used two intelligent search algorithms, called fruit fly optimization and particle swarming optimization, to minimize the order-picking time.

Some order batching methods are based on order associations. Hwang et al. [22, 23] proposed heuristic algorithms based on cluster analysis to solve the order batching problems in the AS/RS, and the efficiency and validity of these algorithms were illustrated through computer simulations. Later, Hwang and Kim [24] developed an efficient order batching algorithm based on cluster analysis for the low-level picker-to-part warehousing system. Chen and Wu [25] described the order batching problem based on a priori data mining to develop the association rule and then established an order clustering model based on 0-1 integer programming to maximize the total associations between the orders within each batch. Chen et al. [26] also developed the same approach to obtain the order and then adopted the seed algorithm to develop the order clustering approach. They then compared the computational results from this approach with those of the other five kinds of heuristic algorithms and drew some useful conclusions. This paper follows that idea: we developed a clustering model with 0-1 integer programming based on the correlation of orders to maximize the correlation of all the batches to obtain the optimal batching.

2.2. Storage Assignment. The storage problem is a research focus in the field of warehouse operations. de Koster [10] classified storage assignment policies into three categories (experience storage, class-based storage, and family
grouping). Experience storage includes radon storage, closest open location storage, and dedicated storage that rely on the long-term accumulation of experience instead of cargo information. On this basis, Gu et al. [27] added the furthest available storage policy and the longest idle storage policy. The class-based storage policy groups products into classes based on the turnover rate of the fastest moving class that is closest to the depot. Family grouping entails possible relations between products, which means similar products are located in the same region of the storage area. Family grouping strategy originates from Frazele and Sharp [28] who proposed the concept of correlated assignments (CA). They employed the statistical method to measure the correlation between pairs of SKUs and then sorted the correlation values in decreasing order to assign the storage locations. Family grouping strategy is also adopted in this study.

How to measure the correlation of items is vital for the family grouping-based storage assignment. Shafer and Rogers [29] summarized the methods of measuring the goods’ correlation and discussed the evolution of these methods. van Oudheusden et al. [30] proposed a pairwise interchange procedure to improve the closeness relationships of the items in a manufacturing plant. Liu [31] proposed measuring the item similarity by the number of common items in orders and also considered the correlation between customers.

Some studies consider more factors, other than item similarities, for a better storage assignment in specific applications. Wutthisirisart et al. [32] took into consideration the frequency and size of the orders. They found that orders with more kinds of items should be located far away from the depot, since they typically need more time to handle, while orders with higher inbound/outbound frequency should be closer to the depot. Chiang et al. [33] developed a goods’ correlation index using the data mining approach, in which the turnover rate was involved. Later, they proposed another heuristic algorithm on the basis of the goods’ correlation [34]. Lee [35] clustered the correlated items into a group and then allocated goods based on each group’s COI (cube-per-order index). Rosenwein [36] solved the storage assignment issue using the cluster analysis method. The items in the same group should be as close to each other as possible in order to form an effective pick route. Chuang et al. [37] proposed a two-stage “clustering-assignment” model for the storage assignment problem. The first stage is responsible for clustering items based on the between-item support extracted from customers’ orders, and the second stage aims to minimize the picking distance. Tu et al. [38] considered a storage assignment problem in a pick-to-pass order picking system in which a picker picks part in one zone and hands it over to another picker until all items in the order are fulfilled. A genetic algorithm was proposed to solve the storage assignment problem, additionally considering the workload balance between picking lines.

There are limited research works dedicated to mobile-pod warehouse systems that are related to the storage assignment. Weidinger et al. [39] considered the problem of assigning pods to storage positions when a pod returns from a picking station to the storage area. They minimized the total travel distance of the mobile robots to complete all the tasks. However, they only considered the pod location but not the item location. Similarly, Yuan et al. [40] also focused on the pod location problem in a parts-to-picker system, where the velocity-based storage policies were analyzed. Li et al. [41] also studied the pod location policy in a high-density mobile-pod system. When applying multiclass closed queueing network models, Roy et al. [42] analyzed both order-picking and replenishment processes in a mobile fulfillment system. Valle and Beasley [43] extended the work of Boysen et al. [18] by additionally considering the order and pod allocation to picking station (or pickers as in the paper). The (first-stage) order and pod allocation solution was then submitted to a (second-stage) pod sequencing problem for each picker where the order sequence was also determined. This study formulated and solved the two subproblems separately: for the order and pod allocation problem, two heuristics were proposed; meanwhile, for the pod sequencing problem, the proposed formulation explicitly considered the pod inventory positions which were ignored in the work of Boysen et al. [18], and the CPLEX solver was used to generate a feasible (not optimal) solution.

To summarize, most related studies focus on traditional warehousing systems, such as manpower warehouses and AS/RS; however, only limited studies addressed the mobile-pod system. The fundamental difference is that a robot can carry only one pod (but with multiple items) per trip to a picking station, while a picker needs to visit multiple storage slots per trip. Therefore, measuring the storage assignment and order batching methods is quite different. Even though a few research were dedicated to mobile-pod systems, they did not address allocation items to pods. Our work aims to fill in such a gap.

3. Problem Description

In this section, we introduce the system conditions, warehouse layout, and basic assumptions. We also discuss the coefficients of similarity used in the storage assignment and order batching.

3.1. Mobile-Pod Warehouse System. A mobile-pod warehouse system conducts a profound reform in the picking mode on the basis of the traditional warehouse system, and the mobile robots and pods constitute a highly dynamic system. The system can be simplified into five parts (without loss of generality, we neglect the replenishment and shipping operations). They are the inbound, the storage area, the sorting service area, and the outbound. The layout is shown in Figure 3.

A blue square represents a mobile pod, a grey square means the pod in that position has been removed, and the blank areas between the pods are the aisles where the robots move. The orange squares represent robots without a pod. A blue square embedded in a grey square represents a robot carrying a pod and the arrow indicates its walking direction. There are four picking stations in the order sorting service
area in this example, and a smiley face represents a picker. Robots in the service area need to wait until the picking tasks are finished by the operators to leave. Figure 4 shows more details of the pod (Figure 4(a)) and the picking station (Figure 4(b)).

The main process of order picking in a mobile-pod warehouse that we consider can be summarized as follows:

1. **Storage.** Items are replenished to the warehouse and stored in pods. Specifically, each pod can accommodate multiple kinds of items; that is, mixed storage policy is considered (as Figure 3(a) shows). To allocate storage locations (pods) to each item, the storage assignment problem must be tackled, which answers the question of which items should be stored in the same pods. This is equivalent to dividing the item SKUs (Stock Keeping Units) into several nonoverlapping groups. The grouping of items is based on the item similarity, which will be formally defined in Section 3.3.

2. **Batching.** A set of orders are divided into batches and each batch will be processed in a picking station simultaneously. The batching of orders is based on the order similarity (see Section 3.4); the logic behind this is that orders with common items are more likely to be fulfilled by one trip of pod moving to the picking station.

3. **Picking.** For a batch of orders, one or multiple pods are moved from their storage slots to the associated picking station (as shown in Figure 3(b)), to meet the desire of items by the orders. Once all the orders in this batch have been picked, another batch will be processed next.

Our aim of this study is to identify effective storage assignment and order batching methods to minimize the number of moves by all robots. To do so, we turn to maximize the overall order similarity in the order batching problem and to maximize the overall item similarity in the storage assignment problem (see Section 4.1). Then, the number of robot moves is estimated by a saving-based method which will be given in Section 4.2.

3.2. Assumptions. In this study, we made the following assumptions:

1. Each item can only be stored in one mobile pod, and the number of items that can be stored in each pod is limited.

2. The maximum number of orders in each batch is given due to the limited capacity in the picking station.

3. Each picking station can accept the next batch only if the current batch is finished; there are no priorities concerning the picking station sequences.

4. Orders that need picking are known and static instead of arriving dynamically.

5. Order-picking efficiency is measured by the number of robot trips to and from the picking station, despite the velocity of robots and the amount and the routes of robots.

3.3. Item Similarity. Item similarity represents the possibility that two items appear in the same order, which can be evaluated by the information in the historical transaction data. The higher this similarity is, the stronger the association between the two items is, and the more likely the two items should be stored in the same pod. We adopt two types of item similarities as given below.
3.3.1. Jaccard Index. Developed by the Swiss mathematician Paul Jaccard, the Jaccard Index is used to compare the similarities and differences between sample sets. Frazele and Sharp [28] also used the Jaccard Index to measure the association of goods in their study of storage assignments. Given \( N \) orders, \( O = \{ O_1, O_2, \ldots, O_N \} \), containing \( P \) items in total. For any item \( A \) and item \( B \), denote \( a \) as the number of orders that contain both \( A \) and \( B \):

\[
a = \sum_{i=1}^{N} CAB_i, \tag{1}
\]

where \( CAB_i \) is a 0-1 variable:

\[
CAB_i = \begin{cases} 
1, & \text{if } A \in O_i \cap B \in O_i, \\
0, & \text{else}.
\end{cases} \tag{2}
\]

Define \( b \) as the number of orders that only contain item \( A \):

\[
b = \sum_{i=1}^{N} CA_i, \tag{3}
\]

where \( CA_i \) is a 0-1 variable:

\[
CA_i = \begin{cases} 
1, & \text{if } A \in O_i \cap B \notin O_i, \\
0, & \text{else}.
\end{cases} \tag{4}
\]

Denote \( c \) as the number of orders that only contain item \( B \):

\[
c = \sum_{i=1}^{N} CB_i, \tag{5}
\]

where \( CB_i \) is a 0-1 variable:

\[
CB_i = \begin{cases} 
1, & \text{if } A \notin O_i \cap B \in O_i, \\
0, & \text{else}.
\end{cases} \tag{6}
\]

Then, \( S_{AB} \) is defined in the Jaccard Index as

\[
S_{AB} = \frac{a}{a + b + c}, \tag{7}
\]

which is in the range of \([0, 1]\).

3.3.2. Weighted Support Count (WSC). Developed by Ming-Huang Chiang et al. [34], the WSC represents the association between any pair of goods and it involves the concepts of the \textit{support} and \textit{lift}, which are originally from data mining. Given \( N \) orders, \( O = \{ O_1, O_2, \ldots, O_N \} \), containing \( P \) items in total. For any item \( A \) and item \( B \), the following rules apply:

1. \textit{support}, defined as the percentage of the orders that have both \( A \) and \( B \), represented by \( P(A \cup B) \) here, is expressed as

\[
P(A \cup B) = \frac{a}{N}. \tag{8}
\]

2. \textit{lift}, defined as the ratio of the probability of having \( B \) under the condition of having \( A \) and the probability of having \( B \), illustrates the type of relationship between the items. \( \text{Lift}_{AB} \) is defined as

\[
\text{Lift}_{AB} = \frac{P(B|A)}{P(B)} = \frac{P(A \cup B)}{P(A)P(B)}, \tag{13}
\]

If the lift value is greater than 1.0, the actual probability of \( A \) and \( B \) appearing in the same order is greater than the theoretical probability (\( A \) and \( B \) are complementary to each other). If the lift value is less than 1.0, \( A \) and \( B \) are substitutive for each other. If the lift value equals 1.0, \( A \) and \( B \) are independent of each other. \( LW_{AB} \) is the relationship of two items:

\[
LW_{AB} = \begin{cases} 
1, & \text{if } \text{Lift}_{AB} > 1 \text{ complementary}, \\
0, & \text{if } \text{Lift}_{AB} = 1 \text{ independent}, \\
-1, & \text{if } \text{Lift}_{AB} < 1 \text{ substitutive}.
\end{cases} \tag{14}
\]

The WSC coefficient is then defined as

\[
S_{AB} = LW_{AB} \times P(A \cup B). \tag{15}
\]

As equations (14) and (15) suggest, unlike the Jaccard index, the WSC coefficient can be a negative value that varies between \([-1, 1]\). A negative WSC coefficient between two items will lead to less probability of assigning them to the same pod.

3.4. Order Similarity. The association between a pair of orders is measured by their similarities, which can be represented by the number of items in common regardless of the quantity of each kind of item. Orders with a high similarity should be allocated to the same batch. More specifically, the order similarity is the product of the proportions of items in common in each of the two orders being measured, within the range of \([0, 1]\). For orders \( A \) and \( B \), containing \( P \) kinds of items in total, the following variables were utilized:

\[
P(A) = \sum_{i=1}^{N} A_i, \tag{9}
\]

where \( A_i \) is a 0-1 variable:

\[
A_i = \begin{cases} 
1, & \text{if } A \in O_i, \\
0, & \text{else}.
\end{cases} \tag{10}
\]

\[
P(B) = \sum_{i=1}^{N} B_i, \tag{11}
\]

where \( B_i \) is a 0-1 variable:

\[
B_i = \begin{cases} 
1, & \text{if } B \in O_i, \\
0, & \text{else}.
\end{cases} \tag{12}
\]
\( U_A \), set of goods order A contains.
\( U_B \), set of goods order B contains.
\( M \), number of goods order A contains.
\( N \), number of goods order B contains.
\( C \), number of goods orders A and B contain in common, expressed as
\[
C = \sum_{i=1}^{P} \sum_{j=1}^{P} X_{ij}.
\] (16)

\( X_{ij} \) is the 0-1 variable:
\[
X_{ij} = \begin{cases} 
1, & \text{if } i \in U_A \cap j \in U_B, \\
0, & \text{else}.
\end{cases}
\] (17)

Simi_{AB}, the similarity of orders A and B, is expressed as
\[
\text{Simi}_{AB} = \frac{C}{M} \times \frac{C}{N}.
\] (18)

For example, if orders A and B have no items in common, \( C = 0 \), Simi_{AB} = 0; if orders A and B are exactly the same, then \( C/M = 1, C/N = 1 \), and Simi_{AB} = 1; if orders A and B are partly the same, then \( 0 < \text{Simi}_{AB} < 1 \).

4. Models

4.1. The Clustering Model. The main ideas of order batching and storage assignment problems are consistent, since they are both based on associations. Therefore, a common method can be used to solve the two problems simultaneously: clustering analysis.

We use the storage assignment problem as an example. As assumed in the problem description, one kind of item can only be allocated to one cluster, and the items in one cluster are stored in the same pod. We first calculate the item similarities according to the measures defined in Section 3 and then cluster all items with the objective to maximize the sum of all the item similarities in each pod. The order batching problem is approached in a similar way, that is, to maximize the sum of all the order similarities in each batch. We can then build the clustering model for the storage assignment problem using the parameters shown as follows:

\( P \), number of the kinds of items in the storage area
\( K \), number of pods in the storage area
\( C \), maximum number of items that can be stored in one pod
Define the 0-1 variable \( x_{ij} \), \( x_{ij} = 1 \) if item \( i \) is assigned to cluster \( j \); otherwise \( x_{ij} = 0 \)
Define the 0-1 variable \( y_{ij} \), \( y_{ij} = 1 \) if item \( j \) is selected as a cluster center; otherwise \( y_{ij} = 0 \)

We can then use the Jaccard Index and WSC coefficient as the cost coefficients in the objective function separately. The objective function maximizes the sum of all the similarities between the items and their cluster centers and is expressed as
\[
\max \sum_{i=1}^{P} \sum_{j=1}^{P} S_{ij} x_{ij}.
\] (19)

To make sure that every item has been assigned to one (and only one) cluster, we must build the constraint:
\[
\sum_{j=1}^{P} x_{ij} = 1, \quad i = 1, \ldots, P.
\] (20)

The number of clusters is constrained to \( K \), the number of pods, which leads to constraints:
\[
\sum_{j=1}^{P} y_{ij} = K,
\] (21)

Constraint (23) represents the capacity limit of the pod:
\[
\sum_{i=1}^{P} x_{ij} = C, \quad j = 1, \ldots, P.
\] (23)

With variable-type constraints (24) and (25), the model is complete:
\[
x_{ij} = 0,1, \quad i, j = 1, \ldots, P,
\] (24)
\[
y_{ij} = 0,1, \quad j = 1, \ldots, P.
\] (25)

The order batching problem can also use this model (19)–(25), with only minor alterations in the meaning of the parameters. Specifically, in the order batching problem, \( C \) is the maximum number of orders simultaneously handled in the picking station, \( P \) is the number of orders, \( K \) is the number of batches which can be estimated as up rounding \( P/C \).

4.2. The Policy Evaluation Model. We need to evaluate the effect of the storage assignment and order batching policies. During the order-picking process, order batching was conducted first so that orders in each batch are aggregated in terms of items. Then, items were fetched per batch based on the given storage assignment policy which means picking multiple items needed from one pod containing such items. In this study, we propose evaluating the system performance by the saved number of round trips by the robots, which consists of savings from the order batching policy (BatchSave) and from the storage assignment policy (StorageSave).

\[
\text{Save} = \text{Batch Save} + \text{Storage Save}.
\] (26)

Suppose that \( g \in G \) item set, set of storage items;
\( o \in O \) order set, set of original customer orders;
\( b \in B \) batch set, set of multiple batches gained from dividing
original orders; and \( k \in K \) pod set, set of pods (a pod can store multiple items).

\[
X_{go} = \begin{cases} 
1 & \text{order } o \text{ contains item } g \\
0 & \text{order } o \text{ does not contain item } g 
\end{cases}, \\
Y_{ob} = \begin{cases} 
1 & \text{batch } b \text{ contains order } o \\
0 & \text{batch } b \text{ does not contain order } o 
\end{cases}, \\
Z_{gb} = \begin{cases} 
1 & \text{batch } b \text{ contains item } g \\
0 & \text{batch } b \text{ does not contain item } g 
\end{cases}, \\
S_{gk} = \begin{cases} 
1 & \text{pod } k \text{ contains item } g \\
0 & \text{pod } k \text{ does not contain item } g .
\end{cases}
\]

We here give an example to show the policy evaluation method.

4.2.1. Order Batching Process. Let

\[
Z_{gb} = \begin{cases} 
1, & \sum_o X_{go} \times Y_{ob} \geq 1, \\
0, & \sum_o X_{go} \times Y_{ob} = 0, 
\end{cases} \tag{27}
\]

\[
Z_b = \begin{cases} 
1, & \sum_g Z_{gb} \geq 1, \\
0, & \sum_g Z_{gb} = 0, 
\end{cases}
\] \tag{28}

and then the saved number of round trips from any batch is

\[
\text{batch save}_b = \sum_g \sum_o X_{go} \times Y_{ob} - Z_b, \forall b \in B. \tag{29}
\]

The saved number from the entire order batching process is

\[
\text{Batch Save} = \sum_b \text{batch save}_b. \tag{30}
\]

For example, suppose that there are nine orders waiting to be picked, and they are \( O_1 = \{1, 2, 3, 4\}, O_2 = \{2, 3, 5\}, O_3 = \{1, 2, 3\}, O_4 = \{2, 3\}, O_5 = \{4, 5\}, O_6 = \{1, 2, 3, 4\}, O_7 = \{2, 3, 5\}, O_8 = \{1, 2, 3\}, \text{and } dO_9 = \{2, 3\}.

Suppose that the batching rules obtained after solving the model are \( B_1 = \{O_1, O_3, O_5, O_6\}, B_2 = \{O_2, O_4, O_5, O_8\}, \) and \( B_3 = \{O_3\} \). Then the three batches contain different items as \( B_1 = \{1, 2, 3, 4\}, B_2 = \{2, 3, 5\}, \) and \( B_3 = \{4, 5\} \). This process reduced the number of round trips the robots made from \( 4 + 3 + 3 + 2 + 2 + 4 + 3 + 3 + 2 = 26 \) to \( 4 + 3 + 2 = 9 \), which shows the advantage of order batching.

4.2.2. Storage Assignment Process. Define

\[
T_{kb} = \begin{cases} 
\sum_g S_{gk} \times Z_{gb} - 1, & \sum_g S_{gk} \times Z_{gb} > 1, \\
0, & \sum_g S_{gk} \times Z_{gb} = 1, 
\end{cases}
\] \tag{31}

as the number of the saved round trips by batch \( b \) when pod \( k \) arrives. Then, the saved number from the entire storage assignment process is

\[
\text{Storage Save} = \sum_b \sum_k T_{kb}. \tag{32}
\]

For the above example, if the items are located based on the storage assignment model, the location assignments are \( \{1, 2, 3\} \) and \( \{4, 5\} \). Under this situation, for the batch orders \( B_1 = \{1, 2, 3, 4\}, B_2 = \{2, 3, 5\}, \) and \( B_3 = \{4, 5\} \), the robot only needs to take \( 2 + 2 + 1 = 5 \) round trips to finish the orders. But if one pod only contains one kind item, the round trips of the robot will be \( 4 + 3 + 2 = 9 \).

To summarize, the number of round trips the robots saved is \( \text{Save} = (26 - 9) + (9 - 5) = 21 \).

5. Experiments and Data Analysis

We solve the models built in Section 4 using the commercial solver Cplex and then simulate the order batching and storage assignment process and gather the experimental data. Based on the data, we evaluate the performance of the different policies through both intuitive and ANOVA analysis.

5.1. Experimental Settings. The experimental instances are generated randomly. Specifically, Orders with capacities limited to 1–4 item lines are randomly generated, with each line corresponding to a unique item. In order to simulate real demands patterns, items are classified into three types: best sellers, secondary best sellers, and ordinary goods, which held 10%, 25%, and 65% storage capacity, respectively. They appear with a frequency ratio of 6:3:1 as shown in Table 1.

We determine a certain combination of the order-picking policies for comparison. For order batching, we consider three policies: no batching, random batching (orders are randomly grouped), and batching by the correlation rule (i.e., by the proposed order similarity). For storage assignment, we consider two policies: random storage and storage by the correlation rule (i.e., by the Jaccard index or the WSC coefficient). By crossing over the batching and storage policies, we form \( 3 \times 2 = 6 \) order-picking policies, as shown in Table 2. For example, Policy 2 means that the goods are stored and batched both by correlation policies.

We then design 4 instance types for the numerical test to evaluate the effects of these policies under different problem scales. The instance types are created by changing the parameters, like the number of orders, items, and pods. We first use a large number of orders and items to obtain the association rules and then use these rules as inputs to feed the clustering model and the evaluation model. Instance type 1 had the smallest scale and instance type 4 had the largest scale. Table 3 lists their details.

For each of the four instance types, we randomly conduct ten experiments under each of the above six policies and each of the two item similarity measures, resulting in
10 × 4 × 6 × 2 = 480 experiments. The results of these experiments are then used to do the following numeric studies.

5.2. Comparison of Policies. We compare the performances of the six order-picking policies with the generated instances. Tables 4 and 5 show the results by using the Jaccard Index and WSC coefficient, respectively. The percentage values represent that the ratio of the number of round trips can be saved (saving degree). The best policies are highlighted by bold fonts.

The comparison of Tables 4 and 5 shows that the saving results are better when we use the WSC coefficient. This is because there may exist complementary, substitutive, or independent relationships between items, and mining these associations when assigning storage can improve the system performance. We further analyze the data gained by using the WSC coefficient.

Through a horizontal comparison of Table 5, Policies 2 and 3 have the best saving effects (both using the correlation rule for storage allocation), and the larger-scale system shows a better performance. The saving results were over 80%, which means that every 100 round trips of the robots can be reduced to 20. Policy 6 performs the worst (and performs poorly on a larger scale), which directly indicates the advantage of the joint storage assignment and order batching.

We further compare the results of instance types 1 and 2 and instance types 3 and 4. We find that all six policies performed worse in type 1 than in type 2, and the same happens in types 3 and 4. The difference is the number of orders used to develop the rules: types 2 and 4 are ten times types 1 and 3. This means that if the system provides enough order data for association rule mining, the advantage of the storage assignment will be more evident, which may result from more accurate association rules (like customer preference) being obtained from more order data.

In the following, we observe the influence of different order batching policies when we fix the correlative storage policy (WSC). An intuitive chart is drawn in Figure 5, where X-axis and Y-axis stand for the index of the experiment number and the saving degree, respectively. Note that numbers 1–10, 11–20, 21–30, and 31–40 represent the random instances with types 1, 2, 3, and 4, respectively. The saving degree is the percentage of saving compared with the worst cases under each instance.

Notice that the saving results of the correlative storage policy are greatly affected by the order batching policy. The average saving result is only 28.25% with the no-batching policy, whereas it can reach 80.00% with the random or correlative batching policy.

For more detailed comparison analysis, we report the result of a pairwise comparison in Table 6.

From Table 6, we conclude the following insights:

(1) Without order batching, the use of correlative storage performs much better than the use of random storage, which can be thousands of percentage points. Larger test scales can result in more obvious advantages (1669.57%–5829.74%).

(2) With random order batching, correlative storage can still perform an average of 10% better than a random storage policy. A larger test scale and an increasing number of items can both improve the advantage
(11.97%–17.83%) of the correlative storage policy. This proves that the accuracy of the correlative rules is related to the number of orders, and these rules perform better with more items.

(3) Under the correlative storage policy, the advantage of the correlative order batching policy increases from 10.85% to 21.24% as the numbers of orders and items increased. This indicates that the accuracy of the correlative rule is directly affected by the size of order data volume.

(4) With the random storage assignment, the performance of correlative batching is better than that of random batching when the test scale is small. No big difference exists when the test scale is large. The correlative batching has a considerable advantage over the policy of no batching at any test scale and a larger advantage with a larger test scale.

(5) With the correlative storage assignment, the correlative order batching results are similar to those with a random policy. The correlative order batching has less advantage (about 1-2 times) than the no batching policy (compared with the random storage policy).

5.3. ANOVA Analysis. We also perform ANOVA (shown in this section) due to the sample randomness. We intend to

Table 4: Average saving degree of the different policies (using the Jaccard index).

<table>
<thead>
<tr>
<th>Policy 1 (%)</th>
<th>Policy 2 (%)</th>
<th>Policy 3 (%)</th>
<th>Policy 4 (%)</th>
<th>Policy 5 (%)</th>
<th>Policy 6 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>13.23</td>
<td>80.29</td>
<td>72.57</td>
<td>75.47</td>
<td>69.97</td>
</tr>
<tr>
<td>Instance 2</td>
<td>24.56</td>
<td>83.57</td>
<td>77.90</td>
<td>75.15</td>
<td>69.83</td>
</tr>
<tr>
<td>Instance 3</td>
<td>18.88</td>
<td>79.05</td>
<td>78.13</td>
<td>68.53</td>
<td>70.68</td>
</tr>
<tr>
<td>Instance 4</td>
<td>24.92</td>
<td>80.99</td>
<td>80.36</td>
<td>67.60</td>
<td>69.71</td>
</tr>
</tbody>
</table>

Table 5: Average saving degree of the different policies (using the WSC coefficient).

<table>
<thead>
<tr>
<th>Policy 1 (%)</th>
<th>Policy 2 (%)</th>
<th>Policy 3 (%)</th>
<th>Policy 4 (%)</th>
<th>Policy 5 (%)</th>
<th>Policy 6 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>24.81</td>
<td>83.18</td>
<td>78.30</td>
<td>75.04</td>
<td>69.94</td>
</tr>
<tr>
<td>Instance 2</td>
<td>26.67</td>
<td>84.51</td>
<td>79.45</td>
<td>73.58</td>
<td>70.61</td>
</tr>
<tr>
<td>Instance 3</td>
<td>30.62</td>
<td>82.21</td>
<td>82.43</td>
<td>68.17</td>
<td>70.21</td>
</tr>
<tr>
<td>Instance 4</td>
<td>30.91</td>
<td>82.67</td>
<td>82.78</td>
<td>68.21</td>
<td>70.26</td>
</tr>
</tbody>
</table>

Table 6: Average values of the performance comparison between policies (WSC coefficient).

<table>
<thead>
<tr>
<th>Condition</th>
<th>Comparison with the storage assignment policy</th>
<th>Comparison with the order batching policy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. Random Correlation</td>
<td>Random Correlation</td>
</tr>
<tr>
<td>Policies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance 1</td>
<td>1:6</td>
<td>3:5</td>
</tr>
<tr>
<td>Instance 2</td>
<td>1669.57%</td>
<td>11.97%</td>
</tr>
<tr>
<td></td>
<td>10.85%</td>
<td>7.33%</td>
</tr>
<tr>
<td>Instance 3</td>
<td>1378.54%</td>
<td>12.53%</td>
</tr>
<tr>
<td></td>
<td>11.82%</td>
<td>7.05%</td>
</tr>
<tr>
<td></td>
<td>-2.90%</td>
<td>4081.80%</td>
</tr>
<tr>
<td>Instance 4</td>
<td>4192.62%</td>
<td>17.41%</td>
</tr>
<tr>
<td></td>
<td>20.66%</td>
<td>5323.83%</td>
</tr>
<tr>
<td></td>
<td>4:5</td>
<td>6.37%</td>
</tr>
<tr>
<td>Instance 3</td>
<td>5829.74%</td>
<td>17.83%</td>
</tr>
<tr>
<td></td>
<td>21.24%</td>
<td>-2.91%</td>
</tr>
<tr>
<td></td>
<td>12953.20%</td>
<td>168.96%</td>
</tr>
<tr>
<td></td>
<td>2:3</td>
<td>-0.12%</td>
</tr>
<tr>
<td></td>
<td>168.35%</td>
<td>240.72%</td>
</tr>
</tbody>
</table>

Figure 5: Comparison of the order batching policies on the system performance.
test the significance of the performance differences. ANOVA is a statistical method developed in the 1920s. It studies experimental data to interpret the relationship between experimental variables. Hong et al. [42] also used ANOVA to test the significance level of the effect of different factors in a large-scale order batching algorithm and used the t-test to compare different algorithms.

We apply double factor variance analysis. Suppose that the two factors, the order-picking policy and the numerical instance, influencing system performance are not independent of each other (i.e., their interaction needs consideration).

We first propose the hypothesis.

Hypothesis of row factors:

$H_0$: Row factor (example) has no significant influence on the degree of system performance.
$H_1$: Row factor (example) has significant influence on the degree of system performance.

Hypothesis of column factors:

$H_0$: Column factors (order-picking strategy) have no significant influence on the degree of system performance.
$H_1$: Column factors (order-picking strategy) have a significant influence on the degree of system performance.

Hypothesis of interaction:

$H_0$: There is no interaction between the two factors.
$H_1$: There is an interaction between the two factors.

We then construct test statistics. Because of the article space limitation, we omitted the mechanical steps and went straight to the analysis of variance.

We use a significance level of $\alpha = 0.05$. The analysis is derived by Excel and the results are shown in Table 7 (we skip the calculating processes). The results are all significant.

SS: sum of the squares, df: degree of freedom, MS: mean square, $F$: F statistics, $P$ value: $P$ value used for the test, and $F$ crit: critical value given level $\alpha$.

We utilized the $F$ value to make decisions and got three conclusions from Table 7. First, for the “Sample” row factor, the $F$ test value is $4.602848 > F_{0.05}(3, 216) = 2.646398$. So, we reject the original hypothesis $H_0$ of the row factor, which reflects that the influence of the row factor is significant. Second, for the “Column” (order selection policy) factor, the $F$ value $19846.64 > F_{0.05}(5, 216) = 2.255861$. Thus, we reject the original hypothesis $H_0$ of the column factor. This means that the order selection policy significantly influences the degree of system performance. Last but not least, we analyze the test value of interaction factor: $F = 27.43351 > F_{0.05}(15, 216) = 1.712905$. We should reject the original hypothesis $H_0$ of the interaction factor, which demonstrates that interaction of instances and selection policy significantly influences system performance. Therefore, the analysis confirms the conclusions drawn in Section 5.2.

6. Conclusions

This work studies how the order-picking policy in a mobile-pod warehouse differs from traditional manual or automated warehouse (e.g., AS/RS) systems. Firstly, we focus on the joint impact of storage assignments and order batching policies on order picking process, which is still not well addressed in the mobile-pod warehouses. Secondly, we propose a unified clustering models to handle the storage assignment and order batching problems, followed by an effective policy evaluation model. Last but not least, we conduct numerical experiments to test the six crossover policies and draw some practical conclusions, which can shed light on industrial applications.

The conclusions are summarized as follows:

1. Among the six order-picking policies, the combination of the correlative storage assignment policy and the correlative order batching policy performs the best
2. For the two item similarity measures, the WSC coefficient performs better than the Jaccard Index
3. More historical data helps obtain accurate correlation information and make better storage and batching decisions, which can significantly improve the order-picking efficiency

There are some extensions that can be studied in future research. First, we assumed that one kind of item could only be stored in one mobile pod, but items stored in several pods are also common in practice. Since single and dedicated storage locations may result in low efficiency in replenishment and order picking, this is an area that requires further study. Second, we did not consider the specific quantity of each kind of items or parameters like volume. These extensions can be made by adding quantity and space constraints.

Data Availability

The data are generated randomly based on the experiments.

Conflicts of Interest

The author declares that there are no conflicts of interest.
References


